

rtgwg
Internet-Draft
Intended status: Informational
Expires: November 10, 2022

S. Bryant
University of Surrey 5GIC
A. Clemm
Futurewei Technologies, Inc.
May 09, 2022

Token Cell Routing Data Plane Concepts
draft-bcx-rtgwg-tcr-02

Abstract

Token Cell Routing is a powerful yet hardware friendly method of constructing data plane packets to meet the needs of new applications. It is based on the use of token cells (special kinds of lightly structured tokens) to provide pointers to procedures pre-positioned in the forwarding layer together with the parameters needed to provide the required processing context. A packet can be composed from multiple token cells as needed to result in new network processing and forwarding semantics.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 10, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	4
2. The TCR Concept	4
3. Relationship to prior work	5
4. TCR Packet Structure	6
5. Token Cell types and categories	7
6. Token Cell Structure	9
7. Token Cell Processing Model	10
8. Token Cell Processing Order	12
8.1. Serial Token Cell Processing	13
8.2. Parallel Token Cell Processing	14
8.3. Combined Serial and Parallel Token Cell Processing	16
9. Token Cell Pushing and Token Cell Popping	18
10. Selected Token Cell Type Categories	19
10.1. Disposition Token Cells	19
10.2. Scratchpad and Metadata Token Cells	19
10.3. Conditional and Directive Token Cells	21
10.4. Security Token Cells	21
11. Example applications of TCR	23
11.1. Basic Tunneling of Payload	24
11.2. Latency-Based Forwarding	25
11.3. Forwarding with Flexible Addressing	26
11.4. Forwarding with iOAM analytics	27
11.5. FRR with Latency-Based Forwarding	30
11.6. Segment Routing with Latency-Based Forwarding	32
11.7. Enhanced Segment Routing with Latency-Based Forwarding	32
11.8. Enhanced Segment Routing with Differentiated iOAM	33
12. Items for further discussion	35
13. Security Considerations	36
14. IANA Considerations	38
15. References	38
15.1. Normative References	38
15.2. Informative References	38
Authors' Addresses	39

1. Introduction

Advances in data plane protocols are needed to address new network requirements that stretch existing protocols (including MPLS, IPv6 and Segment Routing) to their limits. Token Cell Routing (TCR) is a new network layer data plane technology that provides the ability to

program the data plane to meet the needs of many new operational scenarios. TCR is based on token cells which provide a flexible method describing the required packet action to the forwarder as well as carrying any parameters and other data necessary to correctly execute the required action.

Packet actions are not limited to forwarding actions, and it is possible perform multiple packet actions at any given node. For example, packet actions can include application of special QoS algorithms, collection of telemetry, even assessment of dynamic conditions before the performing of other actions. Token Cells can be differentiated by type depending on the type of packet action that they represent.

TCR is thus analogous to each packet carrying a stack of pointers to procedures together with the data needed by those procedures. The structure used allows token cells to be sequenced and parallelized in groups, and permits the use of pointers to information in other token cells. This results in a powerful method constructing advanced new packet types from token cells to meet network needs of new applications.

TCR therefore supports customizable semantics with which packets are to be processed by nodes encountered along a path, it accommodates flexible addressing semantics that do not necessarily depend on a single addressing format. TCR accommodates custom "guidance" beyond forwarding (such as the definition of QoS treatments that are to be applied, the ability to differentiate behavior depending on dynamic context encountered at a node, and the ability to collect and pre-process telemetry in support of manageability applications). TCR enables the direct processing via a "scaffold" that explicitly indicates serialization, parallelization, disposition rules. It that enables a "lego-esque" composition of packet processing behavior / features while at the same time being hardware-friendly, and easy to optimize for performance at line rate general in nature and easy to extend.

A new TCR features is added by introducing a new procedure in the forwarder and then including in the packet a pointer to that procedure. The procedure knows how to interpret the other information carried in the token cell. In many ways this is similar to the way new FECs are introduced into MPLS and new instructions introduced to segment routing.

TCR thus provides a general, highly extensible data plane that supports custom semantics which is hardware friendly. It thus takes the programmability of both MPLS and Segment Routing to a new level of capability.

A key differentiator from earlier protocols is the ability to process a variable number of processing actions at each hop, at directed by the token cell structure. Furthermore token cells do not need to be processed in the order in which they are placed in the packet, and may be explicitly programmed for a flow.

We would like to note that at the time of writing the current -00 version of the draft, this represents a sketch of an idea that neither we or anyone else has built. Thus, it is likely to have bugs and certainly has many aspects that can be improved on. We would be delighted to work with others who are interested in exploring this idea and developing it further. A starter set of discussion items is included in Section 12 towards the end of the document.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The TCR Concept

The foundation of TCR is the construction of the packet from a set of token cells. A token cell is an extended length, type, value construct. The type and part of the value is processed by a longest match engine, which operates much like an IP address lookup engine, but operates on arbitrary constructs rather than being confined to address lookup. As part of its value, the token cell may also carry parameters specific to the token cell type that are needed to process the token cell. Depending on the token cell type, different code points can be invoked that process the token cell. The token cell type determines the semantics, i.e. the function to be applied. It also defines any structure that may be contained in the token cell value.

Token Cell processing can have generalizable packet processing semantics: forwarding is a common semantic, but other semantics can be applied, in a similar manner to the way in which an MPLS label has generalized semantics. The processing of a token cell is: Input - Match - Effect, where "effect" is one of forwarding, token cell disposition, or something else (such as, conditional directives or QoS treatment). Packet processing based on the first n bits of the token cell at a known position, which is matchable on prefix, in which case less significant bits serve as input.

The TCR approach to packet design is differentiated from common protocol designs in that it allow for processing of variable number of token cells per hop, as directed by token cell structure. Which token cells to process, and whether token cells have interdependencies that require them to be processed in serial order or whether they allow for parallelization, is indicated by the token cells themselves, as the token cell structure includes length and serialization indicators. Processing can be serialized per "next token cell" indicator. Processing can be parallelized per "manifest token cell" that refers to parallel token cells in order to allow for optimization. Note that there is no requirement to "must" parallelize. Instead, parallelization is an optimization that nodes with support for parallel packet processing stages may take advantage of to reduce packet latency due to packet forwarding time.

Token Cells do not necessarily need to be processed in stack order but can be located wherever is most efficient. Some token cells may not be processed at all but can be used to carry meta-data (read-only or writable) that can be referred to from other token cells.

The TCR approach allow for extensibility and programmability through:

- o Level 1: Combining different token cells
- o Level 2: Parameterizing token cells
- o Level 3: Introducing new types of token cells

3. Relationship to prior work

In general data plane packets have been designed with a fixed structure plus some variable number of TLVs to provide additional instructions/advice to the forwarder. In general any address (for example IP address) or instruction (MPLS label or SR SID) is of a fixed length although may be structured into a prefix and suffix arrangement to support aggregation and is processed using a longest match lookup. Where TLVs are used there is their inclusion in the packet is normally free form and thus it is necessary search the packet for the set of TLVs that need to be processed. MPLS has a very primitive parameter system in which one label may be used to provide context for the label that follows. Examples of this are the use of the context label and the use of the ELI/EL pair.

In deigning TCR we noted the generality and simplicity of the MPLS label stack model and the effectiveness of the longest match technique used in IP lookups. Putting these two together we concluded that an LTV approach allowed the creation of a simple, powerful, extensible, hardware friendly packet design.

As we will see later in the document, concatenating the T(type) and the V(value) allows a single longest match look-up to resolve which table to look up the action in, and then to absorb as much of the V as is necessary to determine the specific action to take on the T. The stack and pointer structure means that it is not necessary to search for the applicable TLVs, they the forwarder is led to them through the token cell structure pushed onto the packet.

The general method of processing is thus input - match - effect via a match processor. The normal effect is to process one token cell at a time in series, but an effect might be to process multiple other token cells in parallel where the forwarder supports multiple concurrent operations (where parallization is not supported serial processing results in the same effect). Token Cells can have interdependencies and they can allow for cross-referencing (e.g. meta-data, scratch-pad)

4. TCR Packet Structure

A TCR packet Figure 1 is a series of token cells, each token cell carrying a component of the packet delivery system or the payload itself.

```
<Preamble>
<Token Cell>
<Token Cell>
<Token Cell>
...
<Token Cell>
```

Figure 1: Structure of a TCR Packet

Token Cells are a unit of packet processing that may include parameters and/or data. The semantics, structure and processing of the token cell is determined by the token cell type. A Token Cell can be thought of as a type of "stem cell" that can be morphed into any packet component, including components not yet designed, resulting in a highly programmable packet design.

The preamble contains a small number of packet elements that are always present in all packets, such as version identification and TTL. It has yet to be decided if the preamble should be carried conventionally or be carried within a token cell.

There is no separate payload portion of the packet. Instead, the payload is carried within a token cell, generally located at the tail of the packet. This allows for different payload delivery semantics at the destination, including simply stripping the payload off or

applying a special type of codec. It also allows for the possibility of payload-less packets that can be used for signaling and control purposes.

A token cell MAY contain a complete TCR packet permitting hierarchical encapsulation.

5. Token Cell types and categories

Token Cells have a type. Types themselves can be categorized, depending on the purpose which token cells of that type serve.

The following token cell categories are provided as part of the TCR design:

- o Forwarding: A forwarding token cell specifies the destination address and method of delivery of the packet. It may also include the source address as a parameter, but this could also be specified in a separate token cell. Different types within this category may differentiate between address types such as IPv6 or IPv4.
- o SLO: A Service Level Objective (SLO) token cell specifies the target quality of delivery, such as latency, delivery time, required discard properties etc., each differentiated by corresponding IDs as separate types. This allows intermediate nodes on the path to apply special treatment to the packet, such as scheduling algorithms, resource reservation, or prioritization, in order meet SLOs as requirement.
- o Metadata: These token cells carry metadata that can be referenced and accessed as other token cells are being processed. Metadata can thus be decoupled from token cells that access it, allowing for their independent disposal, not interfering with pushing and popping of other token cells.
- o Scratchpad: Scratchpad token cell are in effect writeable metadata token cells, a category of token cell in which the network takes "notes" during the packet transit. Examples of this include recording the route, adding proofs-of-transit, telemetry data, or packet transit time of particular nodes that were traversed.
- o Security: A security token cell signs parts of the packet with an agreed cryptographic signature. It includes a signature mask that specifies which token cells and/or portions thereof are covered by the signature. This allows for the possibility of not only the sender, but also nodes in transit being able to sign portions of the packet. One example use would be for telemetry data that is

added to a scratchpad by a node being traversed while leaving other parts of the scratchpad open to modification by other nodes.

- o **Conditional:** A conditional token cell is able to test one of more conditions and make invocation of the next token cell (NT) dependent on the conditions evaluating to true. This allows to define more complex behavior, such as the invocation of a particular function depending on a dynamic condition encountered at a node.
- o **Directive:** A directive token cell specifies some type of action that should be performed. An example would be a directive to collect telemetry or OAM data.
- o **Manifest:** A manifest token cell provides a method of specifying which token cells may be processed in parallel. Parallel processing is optional, and the token cells can also be correctly processed serially. It is up to the entity that specifies the manifest to ensure that the parallelism is safe.
- o **Rendezvous:** A rendezvous token cell is a token cell used to ensure that all parallel operations have completed and that it is hence safe to resume serial operation of the forwarder. A rendezvous token cell may specify the first serial operation to execute after the rendezvous, or it may simply hand off to a new token cell.
- o **Disposition:** A disposition token cell describes what is to be done when the packet leaves the TCR domain. Such a token cell might, for example specify a pseudowire [RFC3985] action (strip the TCR header and send the payload to interface X), or a VPN action (lookup the payload IP address in VRF Y). However, the mechanism introduces the opportunity to attach a more sophisticated disposition action, for example "if the packet arrives before time T, forward using VRF V, otherwise drop the packet".
- o **Payload:** A payload token cell simply carries the payload as its value.
- o **Other:** This is a catch-all category to allow for token cell types that do not fit any of the other categories.

Some of the mentioned categories will be described in greater detail in Section 10. In addition to the mentioned categories, it is expected that other categories would be introduced as needed.

The grouping of types into categories may prove useful for various purposes. For example, it will allow for the articulation of packet grammars and "best packet practices", such as mandating that a packet

contain at least one token cell of the forwarding category, or that the first token cell in a TCR packet must not be a token cell of categories metadata, scratchpad, or security. Types allow to further differentiate token cells within a given category.

It should be noted that some of the token categories should be considered experimental. Which token types and even which token categories to support will in depend on the needs of actual deployments.

6. Token Cell Structure

The structure of a token cell is shown in Figure 2:

```

<Length>
<Next Token>
<Token Cell Type>  +-
    <Cat/Purpose>      |
    <ID>              +- Match Zone
<Token Cell Blob>  |
    <Prefix>         +-
    <Suffix>

```

Figure 2: Structure of a TCR Token Cell

Token Cells will vary in length depending on the token cell type and the contents of the token cell blob, although in practice a given token cell type MAY be a fixed size.

Next token cell is a relative pointer (offset) to the next token cell to be processed as part of the group of token cells to be sequentially processed as a part of the packet action at the node. If there is no next token cell to process, its value is null. Processing of a packet begins with processing the first token cell. Whether or not any additional token cells are processed depends on the guidance provided via the Next Token field.

The token cell type is used to identify the category or purpose of the token cell (Cat/Purpose) and the sub-type (ID) within that class. An example of a purpose might be "Forwarding", indicating that the token cell represents an instruction to forward the packet closer to the destination. An example of a sub-type within that category might be "IPv6", indicating tht the destination is identified by the following IPv6 address.

The set of IDs consists of a set of well known IDs and a set of user specified IDs. This provides both an extensible, and a programmable mechanism for enhancing the protocol over time and within

deployments. Token cell type, category, and sub-type ID are not fixed-length fields but represent conventions.

The token cell blob carries the information needed to process the explicit packet that carries it, and/or is a place to record information about the packet for later use. Within the token cell blob there is a prefix and a suffix, which may themselves each be structured. The structure of the token cell blob depends on the token cell type, some of which may themselves be subjected to standardization.

The token cell blob prefix, which is neither fixed nor a fixed length field is used to qualify the type in the lookup. For example, if the sub-type was IPv6 destination address the prefix would be an IPv6 address. However this is a more general device than just a destination address and allows for token cell type categorization. This may prove useful for various purposes (e.g. packet grammars and Best Packet Practices).

The match zone is the portion of the token cell that is subject to look up (see processing model section). The model is that the lookup will be a longest match of the whole match zone. The token cell design does not specify the length of the match zone or the length of either the ID or the prefix. It is a property of longest match lookup that it will either consume all the bits it needs, or reject the lookup. If a result is found the result can specify the structure of the token cell blob. Note that this is a model, and there is much scope for implementor optimization without sacrificing the generality of the design.

The number of bytes sent to the lookup engine is implementation specific. If the attempted match is longer than needed, the longest match will ignore the overspill. If more bytes are needed, it is a property of longest match that the lookup can be restarted from where it left off.

Within the blob, and in particular within the suffix, any structure can be carried such as subfields, parameters. The definition of what the suffix contains and how it is structured is part of the token cell type definition.

7. Token Cell Processing Model

The TCR processing model is shown in Figure 3.

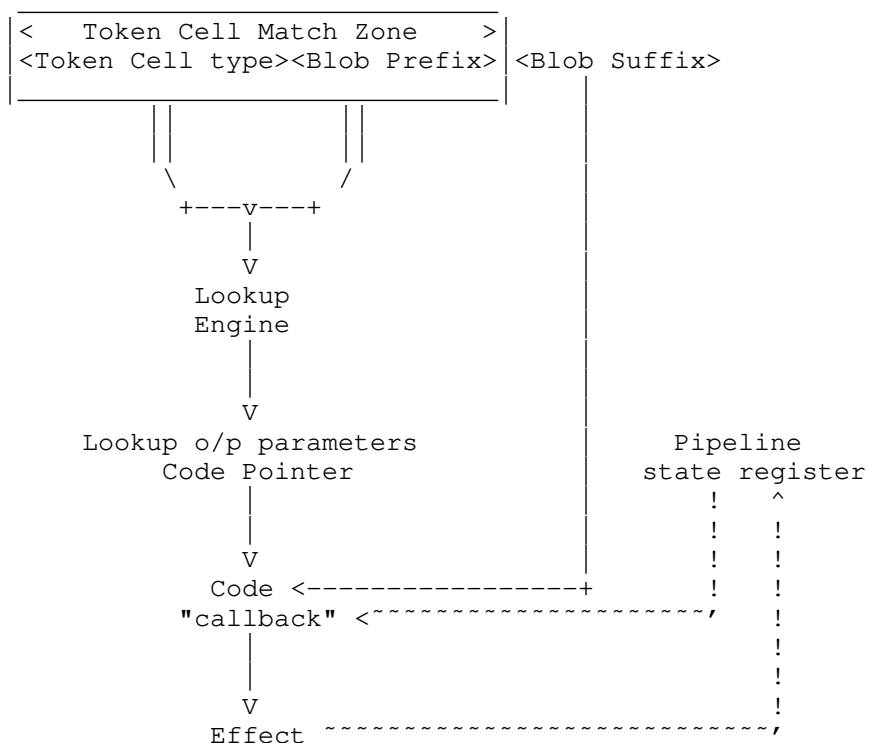


Figure 3: TCR Processing Model

This operates as follows. First the token cell match zone is fed into the lookup engine. The lookup engine performs a longest match and returns a parameter block which includes the address of the code to be executed on the token cell by the forwarder. That code knows how to interpret the token cell. Readers will recognize the genesis of this is a hybrid between an IP address lookup which performs a longest match lookup and returns a parameter block to the IP forwarding code, and an MPLS label lookup which performs a fixed size look-up logically returns a pointer to the executable code (MPLS forward packet, pseudowire, VPN etc) and a parameter block.

Where it cannot be clear that what the length of the blob prefix is from the lookup result, for example where the address is an IPv6 address, that length needs to be encoded in the prefix in some convenient way, such as as a prefix to the prefix.

From the above, it is clear that there is no constraint on the type and structure of the prefix and thus any address type or other construct may be submitted to the lookup engine. Token Cell types

and prefix sets may be added to the forwarder by adding appropriate data to the database queried by the lookup engine, and providing the corresponding callback code to the network processor, in a manner similar to that used in MPLS and in Network Programming. In this regard the approach is compatible with proven hardware forwarding models.

The callback code has access to any other element of the token cell that it needs, and indeed to other elements of the packet as required. Implementations MAY impose a reasonable limitation on the number of processing cycles within which callback code needs to complete.

As part of the effect of processing a token cell, a pipeline state register can be written which can serve as input for the processing of subsequent token cells of the same packet. This allows to compose a pipeline of token cells being processed in which the output of one function serve as input to the next function. In the special case of rendezvous token cells that have multiple predecessors, their respective outputs are merged as part of the specific rendezvous semantics.

8. Token Cell Processing Order

It is entirely possible to require several token cells to be processed at any given node. For example, one token cell may contain a forwarding directive, whereas another token cell might contain a directive related to QoS treatment of the packet for meeting a Service Level Objective (SLO), while a third token cell indicates that certain telemetry data from traversed nodes should be collected.

In the simple case that token cells can or should be serially processed, the processing of token cells will simply be chained, as directed by the token cells' respective NT fields. After processing of a token cell concludes, the reference in the NT field is resolved and processing continues with that token cell. If the NT contains no reference (or in the special case of a conditional token cell evaluating to false), the processing of the packet concludes. In that case, the processing of a packet will require n stages, n being the number of chained token cells.

A packet processing pipeline needs to support a depth that is equivalent to the maximum number of token cells that can be chained.

In some cases, optimization is possible by exploiting parallelization. In the earlier example, it may be possible to perform some tasks in parallel, such as the application of QoS treatment and collection of telemetry data, while other tasks may

still need to be serialized, such as determining which outgoing interface to use as a forwarding decision before performing the QoS actions against that interface. The fact that certain token cells may be processed concurrently can be indicated through a special manifest token cell that references the token cells that follow. Each of those token cells can in turn have their own successors, in effect resulting in separate packet processing "threads" (all processing different token cells of the same packet). While the processing of the manifest adds an additional cycle, depending on the complexity of the workflow this may be more than offset by the parallelization that ensues.

While full exploitation of the optimization potential may require advances in hardware pipeline design, it should be emphasized any such optimization is optional and not required. Also, to maintain packet ordering, the packet will generally still be required to pass through all n pipeline stages. That said, the option of parallelization does allow for hardware pipeline designs able to exploit concurrency of multiple threads and accommodating a larger number of token cells than would otherwise be supported by pipelines of a given depth, respectively reduce the pipeline depth that needs to be supported.

8.1. Serial Token Cell Processing

All packets have an element of serial processing in that the preamble and then the token cell that follows are always processed.

A serial group of token cells is constructed by using the next pointer to point to the token cell to be processed on completion of the token cell. The end of a serial token cell group is logically indicated using a NULL next token cell pointer, although none of the foregoing should be taken as dictating the wire format which will be the subject of another text.

```
Token Cell T1 <Length>
               <Next Token> T2
               <Token Cell Type>
               <Token Cell Blob>
Token Cell T2 <Length>
               <Next Token> T4
               <Token Cell Type>
               <Token Cell Blob>
Token Cell T3 <Length>
               <Next Token> -
               <Token Cell Type>
               <Token Cell Blob>
Token Cell T4 <Length>
               <Next Token> -
               <Token Cell Type>
               <Token Cell Blob>
```

Figure 4: TCR Serial Token Cell Processing of Token Cells

Figure 4 shows four token cells. The serial processing instructed by this construct is that token cell 1 is to be processed, followed by token cell 2 and then followed by token cell 4. There are many reasons why this construct is interesting and these are discussed later in this document.

In order to simplify the graphical annotation, references to token cell cells are in the following simply indicated by a numeric identifier instead of being depicted by arrows.

8.2. Parallel Token Cell Processing

In some cases it is desirable to introduce parallel processing to the packet where there are token cells have no result interdependencies. We do this through the introduction of a manifest token cell that contains a set of pointers or offsets to other token cells.

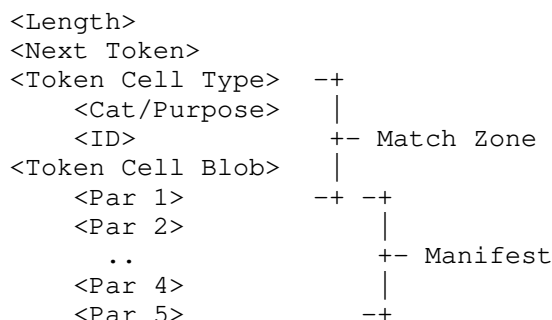


Figure 5: TCR Parallel Processing of Token Cells

The structure of the manifest token cell is shown in Figure 5. The initial part of the token cell is standard to all token cells. There follows a list of token cell pointers to the set of token cells to be processed in parallel. The end of the set of token cells to be processed in parallel is determined when the end of the token cell is reached as indicated by the token cell length. When all the child token cells have completed execution the token cell pointed to by the next token cell field is executed.

Note that the match zone overlaps the manifest. The token cells in the manifest will however be ignored by the longest match which will complete with the ID.

Also note that, as previously mentioned, parallelism is an efficiency issue, not a correctness issue. A forwarder that does not support the parallel dispatch of token cells or that supports less parallelism than specified in the manifest can choose to execute individual token cells (respectively groups of token cells) serially. It is up to the sender to construct the packet as needed and make any token cell interdependencies explicit, without requiring the network to second-guess whether or not there are any such interdependencies. In other words, when the order in which token cells are processed might result in different behavior, it is the responsibility of the sender to specify any required serialization as needed.

In most cases where it is used, a manifest token cell will typically be the first token cell after the preamble, to exploit the possibility of concurrent processing threads for multiple token cells in the same packet from the onset. However, this is not an architectural requirement and manifest token cells could also occur later in the packet.

It is possible to remerge concurrent token cell threads using a special rendezvous token cell. A rendezvous token cell awaits for

each of its predecessors to have completed before resuming processing. In addition, output from predecessors (i.e. pipeline state register content) can be aggregated as needed.

8.3. Combined Serial and Parallel Token Cell Processing

Figure 6 illustrates the concept of parallel and serialized processing further. Please note that this is an admittedly complex scenario and most scenarios in practice will be simpler.

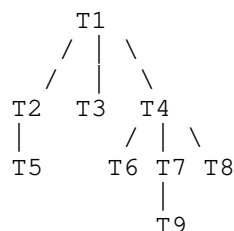


Figure 6: TCR Combined Serial and Parallel Processing of Token Cells

After processing T1, three tokens (T2, T3, T4) can be processed concurrently. T5 has a serialization dependency on T2. T6, T7, T8 can be processed concurrently once T4 has been processed. Finally, T9 has a serialization dependency on T7.

A corresponding packet is shown in Figure 7. Manifest token cells are introduced to represent the fact that T2, T3, T4 respectively T6, T7, T8 can be processed in parallel. A Next Token field of "-" indicates there is no next token.

```

Token Cell T1 <Length>
               <Next Token> M1
               <Token Cell Type>
               <Token Cell Blob>
Token Cell M1 <Length>
               <Next Token> -
               <Token Cell Type = Manifest>
               <Token Cell Blob>
                 <Par 1> T2
                 <Par 2> T3
                 <Par 3> T4
Token Cell T2 <Length>
               <Next Token> T5
               <Token Cell Type>
               <Token Cell Blob>
Token Cell T3 <Length>
               <Next Token> -

```



```

                                <Token Cell Type>
                                <Token Cell Blob>
Token Cell T4 <Length>
                                <Next Token> M2
                                <Token Cell Type>
                                <Token Cell Blob>
Token Cell M2 <Length>
                                <Next Token> -
                                <Token Cell Type = Manifest>
                                    <Par 1> T6
                                    <Par 2> T7
                                    <Par 3> T8
Token Cell T5 <Length>
                                <Next Token> -
                                <Token Cell Type>
                                <Token Cell Blob>
Token Cell T6 <Length>
                                <Next Token> -
                                <Token Cell Type>
                                <Token Cell Blob>
Token Cell T7 <Length>
                                <Next Token> T9
                                <Token Cell Type>
                                <Token Cell Blob>
Token Cell T8 <Length>
                                <Next Token> -
                                <Token Cell Type>
                                <Token Cell Blob>
Token Cell T9 <Length>
                                <Next Token> -
                                <Token Cell Type>
                                <Token Cell Blob>

```

Figure 7: TCR Combined Serial and Parallel Processing of Token Cells

Token Cell M1 is a manifest that indicates three children (at the protocol level, the number of potential children is subject only to packet size constraints). From a protocol perspective all three children: token cell 2, token cell 3 and token cell 4 can execute immediately and concurrently. When token cell 2 completes token cell 5 runs, when token cell 5 completes that that processing branch is completed. Token Cell 3 runs and when it completes that token cell branch is completed. Token Cell M2 is also a manifest with three children: token cell 6, token cell 7 and token cell 8. When token cell 6 completes that processing branch is completed. When token cell 7 is completed token cell 9 runs. When token cell 9 completes that processing branch is completed. When token cell 8 is completed

that processing branch is completed. When token cells 5, 3, 6, 9 and 8 are completed the token cell group is completed.

It should be noted that the current design of the manifest token cell type includes all pointers to subsequent token cells as part of the token cell blob. Alternative designs are conceivable in which (for example) the next token field would be populated with the first of the concurrent successors, with only additional token cells (beyond the first) to be referenced from the token cell blob.

It should also be noted packet permutations can be accommodated, i.e. the order of the token cells in the token cell group can be any order convenient to the network application designer. For example, the same token cells from Figure 7 could have been arranged also in the following sequence: T1 - M1 - T2 - T5 - T4 - T3 - M2 - T6 - T7 - T9 - T8. The token cells do not need to be arranged in a stack in the way that MPLS or SRv6 arrange their labels or SIDs respectively; the order in which to process token cells is always resolved by the NT reference (respectively any manifest token cell references). However it is RECOMMENDED that backward references are avoided as a packet with no backward references will not form a processing loop.

9. Token Cell Pushing and Token Cell Popping

Token Cell groups can be stacked by pushing a group of token cells onto a packet to encode a hierarchical set of operations to be executed on the packet as it journeys to its destination. This is similar to IP tunneling or the pushing and popping of MPLS labels.

In a manner similar to the pipe model described in [RFC3270] it is a matter for the protocol designer and the operator whether the pushed token cells are able to understand and interact with existing the tokens cells. This will be discussed further in a future version of this document.

Similarly when a token cell group has accomplished its purpose on the packet journey it is popped so that the forwarded can gain access to the next element of processing.

Again considerations similar to [RFC3270] may apply.

For the purposes of this discussion a Token Cell group may consist of a single token cell.

Considerations regarding penultimate hop popping will be included in a future version of this text.

10. Selected Token Cell Type Categories

An overview of token cell type categories was given in Section 5. In this section, some of these categories will be explained in further detail.

10.1. Disposition Token Cells

A disposition token cell describes to the network what actions are to be performed as a packet egresses the TCR domain. A forwarding token cell may for this purpose include a reference to a disposition token cell as a parameter.

All existing IETF network protocols include a disposition semantic although sometimes this is implicit. For example when IPv6 has a next header of "TCP" the disposition instruction is to dispose of the IPv6 header and hand the packet to the TCP handler. Similarly pseudowires have a disposition instruction in the PW label instructing the forwarder how to dispose of the MPLS header and to reconstruct the packet in its original format.

However as the metadata carried in the packet become more sophisticated there is a requirement for the disposition to become more sophisticated.

Disposition token cells thus formalize the description of the egress behavior of the network on the packet and allow a richer egress semantic to be described.

It is conceivable to define TCR such that in the absence of a reference to a disposition token cell, TCR will revert to implicit disposition behavior when the destination address of a forwarding token cell is reached. That will involve popping the token cells up to the forwarding token cell and resuming processing with the subsequent token cell. In other words, a disposition token cell is in that case used to specify any special semantics that would go beyond vanilla implicit disposition.

10.2. Scratchpad and Metadata Token Cells

Metadata, provided by a sender of a packet or by an ingress node, can provide important guidance to nodes along a path to guide processing of the packet. Examples include SLOs that should be taken into account for QoS treatment, profiles to apply towards processing a packet, and more. Other uses include the carrying of security material as well as the tagging of packets for classification purposes.

Scratchpads refer to writeable metadata, i.e., metadata that can not only be accessed but also modified or added by nodes "in flight", i.e. during transit. Example uses include collection of telemetry and iOAM data [I-D.ietf-ippm-ioam-data], auxiliary data used for measurements such as intermediate time stamps, or proof-of-transit data and data verifying certain properties of nodes being traversed (such as whether from a trusted vendor or located in a certain region).

Metadata and scratchpad token cells allow to carry such metadata as part of a packet independent of the token cells that access it. This decoupling allows to dispose of metadata and scratchpads independent of token cells that process it. For example, it makes it conceivable to collect different iOAM data along different segments of a path, as directed by different token cells, and to export the data only once an exporter or egress node is reached. This way, scratchpad token cells enable applications that rely on sharing of node-specific data along a path to do so without the complications of having to introduce piggyback extensions to the underlying protocol. In addition to adding and updating data items in scratchpad token cells along a path, also additional scratchpads can be added. It also avoids the need for the same metadata to be copied across token cells, for example in cases where the same SLOs are applicable across different segments, each governed by their own respective token cells.

Metadata and scratchpad items can be referenced by other token cells. The specific format of those references is to be determined by the rules governing the content of the respective token cell blobs for the token cell type; in general the reference format will involve an offset from the referring token cell.

Disposition semantics need to include defining what is to happen with metadata or scratchpad carried in corresponding token cells. Possible disposition actions include:

- o Discard
- o Export (possibly parameterized to specificity export mechanism as well as export target)
- o Log (again, possibly parametrized with a logging target)

Metadata and scratchpad data can also be independently authenticated and secured. This allows, for example, to ensure that scratchpad data that is added or modified by intermediate nodes cannot be tampered with. It also allows for the implementation of operations

that authenticate both metadata and scratchpad data before processing it further.

Metadata, let alone scratchpad data, is a concept that is not directly supported by token-based protocols such as SR or MPLS today. While it would be possible to encode such as part of a token, its processing would require the token itself be processed (e.g. as part of a forwarding operation), tied to its being pushed or popped on a stack. This would make it harder to e.g. update scratchpad data (that should be protected from popping and may be buried underneath a token or label stack), to access data without effectively copying it across tokens, and generally to accommodate metadata and scratchpad processing where the lifecycle of data items does not directly correspond to that of segments. With TCR, there is no need to process metadata in "stacked order" nor need for complex token cell rewrite rules in order to preserve data.

10.3. Conditional and Directive Token Cells

Directive Token Cells allow the specification of some type of action or function that should be performed as a result of the token cell being processed. An example of a directive would be a request to collect telemetry or in-situ OAM data and record it as part of scratchpad. It is expected that there can potentially be a large number of possible directives, each distinguished by its own ID respectively token cell type. Different token cell types may impose their own respective structure on the token cell blob to represent different parameters.

Conditional Token Cells are similar to Directive Token Cells in that they allow the specification of an operation to be performed. However, it is different from other categories of tokens in that the outcome of the operation determines whether the NT field will be processed, i.e. whether processing will subsequently resume with the token cell referenced by the NT field or whether it should terminate. This allows for the definition of functionality that should be performed depending on certain conditions that are being encountered. For example, a conditional token cell might allow for a different paths to be selected depending on dynamic circumstances such as load conditions. Similarly, the collection of certain telemetry data might be made dependent on certain conditions being encountered.

10.4. Security Token Cells

Security token cells enable a security mechanism that allows to sign the invariant elements of the packet whilst avoiding signing the packet elements that are modified during the passage of the packet through the network, such as scratchpad token cells.

Likewise, they allow for differentiated signing of different parts of the packet by different signers, such as in cases where nodes add data items to a scratchpad that should be independently signed.

A security token cell allows to carry signature material pertaining to elements of the packet. It includes the following items:

- o An identification of the signer
- o A mask indicating the parts of the packet respectively token cell(s) being signed
- o The signature material itself

The general structure of the security token cell is as follows

```

<Length>
<Next Token>
<Token Cell Type>  -+
    <Cat/Purpose>    | - Match Zone
    <ID>           -+
<Token Cell Blob>
    <Key ID>
    <HMAC>
    <N>
    <Token Cell Mask>
        <Token_Protected>
            <NP>
            <Full_Token>
            <Byte_Protected>
        <Bytes>

```

Figure 8: Structure of a TCR Security Token Cell

The token purpose and ID specify that this is a security token, the exact structure of the security token, and the type of signature that has been used. The structure used here is illustrative and used to explain the concept.

The token cell blob contains the security material and the mask. A possible structure is as follows:

As described in [RFC8754] the Key ID is used to identify the preshared key and the algorithm, though the algorithm may be indicated by the token ID (this is a matter for the token designer).

The Hashed Message Authentication Code (HMAC) is the hash of the complete TCR security token and the N TCR tokens, and within those tokens the token elements specified by the token mask.

For efficiency the mask operates at a number of levels:

- o The next N token cells
- o The marked tokens within the next N token cells
- o Octets or words within a specific token cell

N specifies the number of tokens covered by this security token in addition to itself. If present, Token Cell Mask is a structure that specifies which tokens are actually protected. Token_Protected.NP is the number of tokens within the token mask.

Token_Protected.Full-Token is a bit mask of Token_Protected.NP bits indicating which complete tokens are to be included in the signature. Token_Protected.Byte_Protected is a bit mask of Token_Protected.NP bits indicating which complete tokens are to be included in the signature.

11. Example applications of TCR

This section contains a number of examples illustrating how the TCR token system is used to create or "program" packets. The examples are illustrative and not exhaustive. Only the essential features of the packet are shown.

In order to simplify depiction of TCR packets, specifically the order in which tokens are processed and the cross-dependencies between tokens, we will introduce a syntax which identifies individual tokens by a numeric token identifier, and that allows to reference tokens using this identifier as opposed to packet offsets depicted using ASCII art pointers. This is merely done for convenience of textually representing TCR packets in this draft. It is independent of the actual TCR packet and token structure, in which tokens do not have separate identifiers and are simply referenced by offset. This applies to Next Token fields, as well as for disposition. A disposition token cell is referenced through a field in token cells whose type is of the forwarding category.

A packet is simply represented by a sequence of tokens. Each token is represented by a set of fields and their values. In addition, a "pseudo field" is introduced for the numeric token identifier. A reference to another token (from the <Next Token> field, from a manifest token, or from a rendezvous token) contains as value the respective token identifier.

11.1. Basic Tunneling of Payload

In this example we consider a the case where a packet is to be tunneled across a network as one might do in a sub-IP network.

For illustrative purposes we show in Figure 9 an IPv6 address being used as a destination address, but any other address family can be used with the corresponding forwarding token type.

```

                                <Preamble>
Token Cell T1 <Length>
                                <Next Token> -
                                <Type = IPv6fwd>
                                <Prefix = Dest Add>
                                <Disp = T2>
Token Cell T2 <Length>
                                <Next Token>
                                <Type = Disp>
                                <Disp Parameters>
Token Cell T3 <Length>
                                <Next Token>
                                <Type = Payload>
                                <Payload>

```

Figure 9: Basic Tunnelling Over TCR

The packet starts with a preamble followed by token T1 which is IPv6 forwarding token, a token with the semantic that it is to deliver the packet as close to the target address as it can reach. There is no next token cell to be processed until arrival at the destination and so Next Token is NULL.

When the packet arrives at its destination the token that is pointed to by the disposition pointer is noted and all tokens up to the disposition token are popped (in this case only token 1 is popped).

The disposition token T2 is processed and the disposition parameters say how the payload is to be processed. This may be as simple as providing the protocol type of the payload, but it may also provide information other information such as the identity of a VRF table to use, or an interface to dispatch the packet to in the case of a pseudowire. It also provides a pointer to the payload. When the parser knows how to dispose of the packet, it pops Token 2 and processes token 3. Token 3, in the case of containing a tunneled IP packet, just requires the length to be noted and corrected for the token overhead, the type confirmed as payload and payload to be forwarded as an IP packet.

11.2. Latency-Based Forwarding

In this example we consider the previous case (basic tunneling of payload) Section 11.1 and extend it to provide support for on-time delivery according to an end-to-end latency objective. The support is provided using a Latency-Based Forwarding (LBF). LBF bases the decision on when to sent to packet on how urgently or at what exact time it needs to arrive. To do so, LBF involves an algorithm that determines at any given node whether the packet is "on track" to meet its latency objective, and matches the QoS treatment and scheduling of the packet against a latency "budget" that is determined from latency objective, the latency that was already incurred, and the expected remaining latency towards the destination. For further details, please refer to [DOI.10.1109_NOMS47738.2020.9110431].

To indicate that a packet should undergo LBF treatment, a corresponding token cell type of category "SLO" is introduced. When it is processed, the packet is subject to LBF. Parameters for LBF include the end-to-end latency objectives and a helper parameter to assess the latency being incurred. An additional input is the egress interface that is obtained from processing of the forwarding token cell that precedes it and that can be passed using TCR's pipeline state register.

A packet that enables LBF in conjunction with the previous case is depicted in Figure 10. It adds one additional token cell over the previous use case, in essence adding the LBF functionality as a "module" that is combined with the earlier functionality. The ability to compose functionality by simply combining corresponding token cells into a packet is part of what makes TCR quite powerful.

Processing of the packet is similar to the processing of the basic tunneling case except that Token 1 the IPv6fwd token contains a pointer to a further token that is to be sequentially processed at every hop, including arrival at the destination. The Next Token pointer in Token 1 points to token 2 which specifies that the packet must arrive at a specified time, and contains information needed to record the time taken and hence the time at which it should optimally leave its current node.

When the packet arrives at its destination a final check is made to see if the arrival time requirement was met and it is processed according to the failure to arrive on time instructions in either the LBF_ontime token or the disposition token. All further processing is as per the above basic tunneling case.

```

                                <Preamble>
Token Cell T1 <Length>
                                <Next Token> T2
                                <Type = IPv6fwd>
                                <Prefix = Dest Add>
                                <Disp = T3>
Token Cell T2 <Length>
                                <Next Token> -
                                <Type = LBF_ontime>
                                <Blob = LBF parameters>
Token Cell T3 <Length>
                                <Next Token>
                                <Type = Disp>
                                <Disp Parameters>
Token Cell T4 <Length>
                                <Next Token>
                                <Type = Payload>
                                <Payload>

```

Figure 10: LBF Using TCR

11.3. Forwarding with Flexible Addressing

There is an emerging need to support multiple address technologies. There are two cases in point, firstly where an operator wants to use a short address to address the infrastructure nodes in their network. This is particularly the case in segment routing where there is competition amongst the vendors to introduce address compression due to the extended size of SRv6 data plane headers. There is a secondary benefit to this in that if an address system other than IP is used within the provider network there are security benefits as has been found in operating MPLS. Finally it has been speculated that some application environments would prefer to use their native addresses rather than manage the mapping between those addresses and IP addresses.

This is achieved by creating a new token cell type, populating the FIB with the corresponding addresses and installing in the forwarder the necessary forwarding code. That forwarding code may be generic since the action will almost certainly be address family independent.

In the example shown in Figure 11, the address family is encoded in the address itself. In an alternative model, the token cell type would be specific to the address family.

```

                                <Preamble>
Token Cell T1 <Length>
                                <Next Token> -
                                <Type = FlexAddr>
                                <Prefix = AddressFamily + Address>
                                <Disp> T2
Token Cell T2 <Length>
                                <Next Token> -
                                <Type = Disp>
                                <Disp Parameters>
Token Cell T3 <Length>
                                <Next Token>
                                <Type = Payload>
                                <Payload>

```

Figure 11: Indirect LBF Using Flexible Addresses in TCR

11.4. Forwarding with iOAM analytics

In many cases, there is desire to collect in-situ OAM data [I-D.ietf-ippm-ioam-data] as a packet traverses a path. There are multiple applications for this, including but not limited to diagnosing performance, identification of bottlenecks, and generation of data to feed machine-learning algorithms for service optimization.

Using TCR, it is possible to indicate that iOAM data should be collected using a corresponding token cell. The token cell can contain in-situ parameters, such as which data items to collect. In-situ data itself can be added to a scratchpad, allowing for its export using a variety of means upon disposition of packet.

One scenario is depicted in Figure 12. In this particular example, we assume that the iOAM data can be collected per T3 in parallel with the forwarding decision being made per T2 in order to show also use of a manifest (T1). iOAM Data items are written to the scratchpad in T5. T4 indicates disposition, T6 contains the payload.

It should be noted that rather than simply collecting iOAM data, other operations could be applied to aggregate that data and result in more refined behavior. In the interest of brevity, the example does not feature security tokens used by intermediate nodes to sign the scratchpad data items that they add.

```

    <Preamble>
Token Cell T1 <Length>
    <Next Token> -
    <Type = Manifest>
        <Par 1> T2
        <Par 2> T3
Token Cell T2 <Length>
    <Next Token> -
    <Type = IPv6fwd>
    <Prefix = Dest Add>
    <Disp> T4
Token Cell T3 <Length>
    <Next Token> -
    <Type = Directive-iOAM>
        <Ioam-parameters - data items to collect>
        <Scratchpad> T5
Token Cell T4 <Length>
    <Next Token> -
    <Type = Disp>
    <Disp Parameters>
Token Cell T5 <Length>
    <Next Token> -
    <Type = Scratchpad>
    <Blob-Scratchpad>
Token Cell T6 <Length>
    <Next Token>
    <Type = Payload>
    <Payload>
```

Figure 12: iOAM in TCR

If iOAM data to be collected includes telemetry about the egress interface, the manifest cell can be omitted as the forwarding decision needs to be made prior to collecting the iOAM data. The resulting packet becomes even simpler, as depicted in Figure 13.

```
Token Cell T1 <Preamble>
               <Length>
               <Next Token> T2
               <Type = IPv6fwd>
               <Prefix = Dest Add>
               <Disp> T3
Token Cell T2 <Length>
               <Next Token> -
               <Type = Directive-iOAM>
               <Ioam-parameters - data items to collect>
               <Scratchpad> T4
Token Cell T3 <Length>
               <Next Token> -
               <Type = Disp>
               <Disp Parameters>
Token Cell T4 <Length>
               <Next Token> -
               <Type = Scratchpad>
               <Blob-Scratchpad>
Token Cell T5 <Length>
               <Next Token>
               <Type = Payload>
               <Payload>
```

Figure 13: iOAM in TCR - serialized

To show the modularity that TCR enables, the third scenario shows iOAM data to be collected while at the same time LBF is being applied to the same packet (Figure 14). Note that in this case, LBF and iOAM could have also been applied in parallel, in which case a manifest token cell would be added. T1 would then point to the manifest token cell, which in turn would point to T2 and T3 as successors.

```

Token Cell T1 <Preamble>
               <Length>
               <Next Token> T2
               <Type = IPv6fwd>
               <Prefix = Dest Add>
               <Disp> T4
Token Cell T2 <Length>
               <Next Token> T3
               <Type = LBF_ontime>
               <Blob = LBF parameters>
Token Cell T3 <Length>
               <Next Token> -
               <Type = Directive-iOAM>
               <Ioam-parameters - data items to collect>
               <Scratchpad> T6
Token Cell T4 <Length>
               <Next Token> -
               <Type = Disp>
               <Disp Parameters>
Token Cell T5 <Length>
               <Next Token> -
               <Type = Scratchpad>
               <Blob-Scratchpad>
Token Cell T6 <Length>
               <Next Token> -
               <Type = Payload>
               <Payload>

```

Figure 14: iOAM and LBF in TCR

11.5. FRR with Latency-Based Forwarding

This example depicted in Figure 15 extends the earlier LBF example to include a fast re-route diversion. We show only a single token cell (T0) added at the point of local repair (PLR), but of course the repair might be more complex and need multiple intermediate staging counts to successfully be repaired.

```

                                <Preamble>
Token Cell T0 <Length>
                                <Next Token> T2
                                <Type = IPv6fwd>
                                <Prefix = Reroute Add>
Token Cell T1 <Length>
                                <Next Token> T2
                                <Type = IPv6fwd>
                                <Prefix = Dest Add>
                                <Disp = T3>
Token Cell T2 <Length>
                                <Next Token> -
                                <Type = LBF_ontime>
                                <Blob = LBF parameters>
Token Cell T3 <Length>
                                <Next Token>
                                <Type = Disp>
                                <Disp Parameters>
Token Cell T4 <Length>
                                <Next Token>
                                <Type = Payload>
                                <Payload>

```

Figure 15: FRR with LBF Using TCR

The PLR was expecting to forward the packet normally and so is aware that the packet is latency sensitive and understands the semantics and hence importance of token 2. To maintain the expected path quality the PLR MUST use an FRR path while also ensuring the SLO is still being adhered to per the LBF token cell. The FRR path therefore needs to feature nodes able to support LBF token cells, and not incur a latency penalty that would physically prohibit being able to meet the SLO. This path can be selected by the SDN controller, or locally through the use of path attributes applied to the normal IP-FRR path selection process.

On detecting a local repairable failure of the next hop or the link to the next hop the PLR pushes one or more tokens as is necessary to deliver the packet to the destination. In each case the next token pointer points to Token 2 the LBF token. When the packet arrives at the intermediate node chosen by the PLR for the next stage of the repair, the token (in this case Token 0) at the top of the token stack is popped and forwarding proceeds as dictated by token 1. The above operation can clearly be carried out as many times as necessary on the packet to provide a repair, by pushing as many tokens as are needed.

Note that the scheme in this example uses an implicit disposition operation by intermediate nodes as described above in Section 10.1.

11.6. Segment Routing with Latency-Based Forwarding

The operation described in Section 11.5 in which repair tokens are pushed onto the packet is identical to the operation that happens when a packet is configured for segment routing (SR). Technically the packet depicted in Figure 15 IS a segment routed packet.

It therefore follows that implementing segment routing and segment routing enhanced by features such as enhanced QoS is trivial in TCR. As we shall see in the next sections, TCR is capable of enhancing SR significantly beyond that.

11.7. Enhanced Segment Routing with Latency-Based Forwarding

This example illustrates how TCR can be used to add an enhanced QoS capability such as latency based forwarding to segment routing. We saw previously Section 11.6 how all segments can be made to execute a common policy but it might be desirable to execute a different policy in different segments. For example, some segments might underly their own latency objectives (just for that segment), without affecting the overall end-to-end objective.

A packet that achieves this is depicted below in Figure 16. The packet in the example has three segments. Segments 1 and 3 apply LBF for the end-to-end latency objective, per T5. Segment 2 applies a "sub- SLO" just for that particular segment, per T4. Disposition of T1 and T2 is implicit (popping the token on reaching the segment destination), whereas disposition of T3 is explicit per T6.


```

Token Cell T1 <Preamble>
               <Length>
               <Next Token> T5
               <Type = IPv6fwd>
               <Prefix = Seg 1 Add>
Token Cell T2 <Length>
               <Next Token> T4
               <Type = IPv6fwd>
               <Prefix = Seg 2 Add>
Token Cell T3 <Length>
               <Next Token> T5
               <Type = IPv6fwd>
               <Prefix = Seg 3 Add>
               <Disp = T6>
Token Cell T4 <Length>
               <Next Token> -
               <Type = LBF_ontime (for segment)>
               <Blob = LBF parameters>
Token Cell T5 <Length>
               <Next Token> -
               <Type = LBF_ontime (for end-to-end)>
               <Blob = LBF parameters>
Token Cell T6 <Length>
               <Next Token>
               <Type = Disp>
               <Disp Parameters>
Token Cell T7 <Length>
               <Next Token>
               <Type = Payload>
               <Payload>

```

Figure 16: Enhanced Segment Routing Using TCR

11.8. Enhanced Segment Routing with Differentiated iOAM

The final example shows a variation of the previous example. Instead of applying a specific latency objectives for particular segments, it is possible to also invoke other functionality, such as collecting certain iOAM data only for particular segments, or collecting additional iOAM data for one of the segments, or even for collecting different sets of iOAM data along different segments. The particular example depicted (Figure 17) shows a packet with three segments. One set of parameters is collected for segments 1 and 3 using scratchpad T6, another set of parameters is collected for segment 2 using scratchpad T7.

If instead, segment 2 should collect additional parameters beyond those collected for segments 1 and 2, this could be easily

accommodated by simply setting "Next Token" of T4 to T5 instead of null. (This does not include a possible optimization for parallelization, but attests to the flexibility of the approach.)

```

                                <Preamble>
Token Cell T1 <Length>
                                <Next Token> T4
                                <Type = IPv6fwd>
                                <Prefix = Seg 1 Add>
Token Cell T2 <Length>
                                <Next Token> T5
                                <Type = IPv6fwd>
                                <Prefix = Seg 2 Add>
Token Cell T3 <Length>
                                <Next Token> T4
                                <Type = IPv6fwd>
                                <Prefix = Seg 3 Add>
                                <Disp = T8)
Token Cell T4 <Length>
                                <Next Token> -
                                <Type = Directive-iOAM>
                                <Ioam-parameters - data items to collect>
                                <Scratchpad> T6
Token Cell T5 <Length>
                                <Next Token> -
                                <Type = Directive-iOAM>
                                <Ioam-parameters - data items to collect>
                                <Scratchpad> T7
Token Cell T6 <Length>
                                <Next Token> -
                                <Type = Scratchpad>
                                <Blob-Scratchpad>
Token Cell T7 <Length>
                                <Next Token> -
                                <Type = Scratchpad>
                                <Blob-Scratchpad>
Token Cell T8 <Length>
                                <Next Token>
                                <Type = Disp>
                                <Disp Parameters>
Token Cell T9 <Length>
                                <Next Token>
                                <Type = Payload>
                                <Payload>

```

Figure 17: Enhanced Segment Routing Using TCR

12. Items for further discussion

This document does not constitute a finalized design and there are many design decisions that will require further discussion. The following is a partial list:

- o Preamble. The preamble needs further study. The goal is that it contains only the minimum of information as it needs to be popped and pushed when a token cell is popped or pushed. We need to understand if there is a need for a number of token cell's indicator, and/or a last child indicator.
- o Token Cell identification. As an alternative to referencing other token cells by offset, it is conceivable to introduce token cell identifiers and refer to token cells by their ID.
- o Manifest token cells. Rather to require processing of a full token cell, it is conceivable to express manifests as part of a token cell preamble, at least as long as the number of token cells to process in parallel are limited. This could save on stage in a token cell processing pipeline. However, it would result in a slightly longer or more complex preamble.
- o Manifest token cells (cont'd.) It should be noted that the current design of the manifest token cell type includes all pointers to subsequent token cells as part of the token cell blob. Alternative designs are conceivable. For example, the design might be altered to allow for the next token to be populated and only require any additional token cells to be referenced from the token cell blob.
- o Rendezvous token cells. As an optimization, it is conceivable to combine manifests and rendezvous points into a single token cell.
- o Security token cells. Further investigation is needed to determine the most effective structure, specifically compact yet efficient encodings for the token cell mask that is used to identify the portions of the packet being signed.
- o Disposition token cells. There are different ways that disposition token cells could be referred to for processing, including through a DT parameter (Disposition Token Cell) as part of forwarding token cells, through a separate field as part of the token cell structure, or indirectly by virtue of popping a forwarding token cell when the destination is reached and processing the token cell behind it.

- o Disposition token cells for processing by intermediate nodes. The example in Section 11.5 uses an implicit disposition operation of forwarding token cells by intermediate nodes, in which a forwarding token cell is simply popped and processing simply resumes with the subsequent token cell. It is conceivable to apply explicit disposition operations instead for the sake of more consistent semantics. Explicit semantics of "pop and resume processing with subsequent token cell" could be incorporated into the semantics of a forwarding token cell itself, or potentially indicated by a corresponding flag in the prefix.
- o Implementation profiles. Implementations may impose a reasonable limit on the number of token cells that can be serially processed at any one node. This will facilitate mapping to packet processing pipelines, which then support a predefined number of token cell processing stages per packet at line rate while maintaining constant packet processing delay at any given node. Determination of reasonable constraints is for further study. Analogous limitations may apply to the number of processing cycles that can be performed by callback functions, within which the processing of any given token cell needs to complete.
- o Implementation profiles (contd.). For further study are any mechanisms for the discovery of constraints as mentioned in the previous list item, as well as any capabilities for negotiation of corresponding profiles and the definition of node behavior when such limitations were to be breached (in all likelihood resulting in aborting the processing of the packet, dropping it with corresponding error code).
- o The design only requires forward token cell pointers and this prevents processing loops. We need to understand if this is too significant a restriction. If this restriction is removed some form of maximum tokens to be processed limit, similar in concept to TTL may be needed.

13. Security Considerations

This section concerns itself with the security of the TCR dataplane.

The security of the control and management plane is a matter for the designers of those aspects of the solution. However, it is not anticipated that the securing of those components will be any more onerous than securing the control and management plane of other IETF designed dataplanes.

Security of entry to the dataplane will depend on what entities have access to the dataplane. If TCR is used as a single domain sub-IP

layer in the way that MPLS is used, then it will have the same security properties as MPLS in that it is extremely difficult for an unauthorized party to inject traffic into such a network because with TCR such traffic is easily recognized at network ingress and dropped. If the traffic is a TCR packet that is to be carried across the TCR network it will be encapsulated and so, in the absence of a TCR network error, will not be able to escape the encapsulation and cause harm. Only if a TCR network (or node) were to peer with another TCR network would there be a security concern with that third party having the ability to control the actions taken on the packet. Such a case is for further study. It should be noted that similar situations have been satisfactorily addressed in MPLS.

We now need to consider the security of the contents of the packet. Clearly we could craft a token that signed the payload, and where the payload was a token, we have the option of including that signature in the token itself. However, securing the tokens themselves is more interesting because we need to authenticate selected components of the packet header, such as single tokens, groups of tokens or even components/portions of tokens. This is needed to allow the differentiation of metadata that must not be altered from scratchpad data that may be modified during packet transit. In addition, we need to allow intermediate nodes along a path to authenticate data that they modify, e.g. scratchpad data items that they create.

The approach used in TCR allows the authentication of tokens and processing guidance they contain for additional security. Furthermore there is flexibility for intermediate hops to provide their own authentication, to secure scratchpad-type data added to packets along a path. This also allows for "authenticity chains" in which nodes verify the authenticity of data items they operate on before modifying and signing the update.

It is clear that the above approach is different from earlier protocols where payloads are generally signed in their entirety and do not include support for differentiated signing, accommodating multiple signers of different packet aspects along a path.

Most protocols secure their payload in its entirety, exposing only the packet header for processing (unless that is tunneled as well). TCR is a protocols that include additional packet components that may require more differentiated securing. Specifically, it includes guidance for how to process packets, including tokens, and metadata. In addition, TCR includes some packet components that can be modified or added by intermediate nodes in transit, specifically scratchpad data. This includes telemetry and iOAM data as well as data to indicate and verify certain properties of nodes that were traversed. While some data must not be modified, other data items might be added

and/or be subject to modification. An example would be data that aggregates or analyzes telemetry data encountered in transit, for example an indicator of a minimum or maximum (queue length, utilization, latency) encountered along a path. TCR thus includes a mechanism that allows the operator to ensure the authenticity of packet data beyond the payload, but that allows them to do this in a way that exempts certain data items which are allowed to be modified along the way, with the option to allow the corresponding nodes to secure these modifications. This allows receiving applications to (for example) verify the authenticity of scratchpad data, and allow for the modification of data items where such modification is permitted without compromising the authenticity of the remaining portions of the packet.

The design of the security token is described in Section 10.4 . This can only be used to sign itself and tokens or token contents after the security token. By including in the security token a mask structure it is possible to select what is to be signed. The efficiency of this method is described in Section 10.4.

Matters related to inter-domain security will be considered in a future version of this text.

14. IANA Considerations

This document makes no IANA requests.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

15.2. Informative References

- [DOI.10.1109_NOMS47738.2020.9110431] Clemm, A. and T. Eckert, "High-Precision Latency Forwarding over Packet-Programmable Networks", NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, DOI 10.1109/noms47738.2020.9110431, April 2020.

- [I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-17 (work in progress), December 2021.
- [RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <<https://www.rfc-editor.org/info/rfc3270>>.
- [RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <<https://www.rfc-editor.org/info/rfc3985>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

Authors' Addresses

Stewart Bryant
University of Surrey 5GIC

Email: sb@stewartbryant.com

Alexander Clemm
Futurewei Technologies, Inc.

Email: ludwig@clemm.org

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 22 June 2022

H. Chen
China Telecom
Z. Hu
Huawei Technologies
H. Chen
Futurewei
X. Geng
Huawei Technologies
Y. Liu
China Mobile
G. Mishra
Verizon Inc.
19 December 2021

SRv6 Midpoint Protection
draft-chen-rtgwg-srv6-midpoint-protection-06

Abstract

The current local repair mechanism, e.g., TI-LFA, allows local repair actions on the direct neighbors of the failed node to temporarily route traffic to the destination. This mechanism could not work properly when the failure happens in the destination point or the link connected to the destination. In SRv6 TE, the IPv6 destination address in the outer IPv6 header could be the dedicated endpoint of the TE path rather than the destination of the TE path. When the endpoint fails, local repair couldn't work on the direct neighbor of the failed endpoint either. This document defines midpoint protection for SRv6 TE path, which enables the direct neighbor of the failed endpoint to do the function of the endpoint, replace the IPv6 destination address to the other endpoint, and choose the next hop based on the new destination address.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 June 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. SRv6 Midpoint Protection Mechanism	3
3. SRv6 Midpoint Protection Example	3
4. SRv6 Midpoint Protection Behavior	5
4.1. Transit Node as Repair Node	5
4.2. Endpoint Node as Repair Node	6
4.3. Endpoint x Node as Repair Node	6
5. Determining whether the Endpoint could Be Bypassed	7
6. Security Considerations	7
7. IANA Considerations	7
8. Acknowledgements	8
9. References	8
9.1. Normative References	8
9.2. Informative References	8
Authors' Addresses	9

1. Introduction

The current mechanism, e.g., TI-LFA ([I-D.ietf-rtgwg-segment-routing-ti-lfa]), allows local repair actions on the direct neighbors of the failed node to temporarily route traffic to the destination. This mechanism could not work properly when the failure happens in the destination point or the link connected to the destination. In SRv6 TE, the IPv6 destination address in the outer IPv6 header could be the dedicated endpoint of the TE path rather than the destination of the TE path ([RFC8986]). When the endpoint fails, local repair couldn't work on the direct neighbor of the failed endpoint either. This document defines midpoint protection for SRv6 TE path, which enables the direct neighbor of the failed endpoint to do the function of the endpoint, replace the IPv6 destination address to the other endpoint, and choose the next hop based on the new destination address.

2. SRv6 Midpoint Protection Mechanism

When an endpoint node fails, the packet needs to bypass the failed endpoint node and be forwarded to the next endpoint node of the failed endpoint. There are two stages or time periods after an endpoint node fails. The first is the time period from the failure until the IGP converges on the failure. The second is the time period after the IGP converges on the failure.

During the first time period, the packet will be sent to the direct neighbor of the failed endpoint node. After detecting the failure of its interface to the failed endpoint node, the neighbor forwards the packets around the failed endpoint node. It changes the IPv6 destination address with the IPv6 address of the next endpoint node (or the last or other reasonable endpoint node) which could avoid going through the failed endpoint.

During the second time period, the packet of a SRv6 TE path may not be sent to the direct neighbor of the failed endpoint node. There is no route to the failed endpoint node after the IGP converges. When a previous hop node of the failed endpoint node finds out that there is no route to the IPv6 destination address (of the failed endpoint node), it changes the IPv6 destination address with the IPv6 address of the next endpoint node. Note that the previous hop node may not be the direct neighbor of the failed endpoint node.

3. SRv6 Midpoint Protection Example

The topology in Figure 1 illustrates an example of network topology with SRv6 enabled on each node.

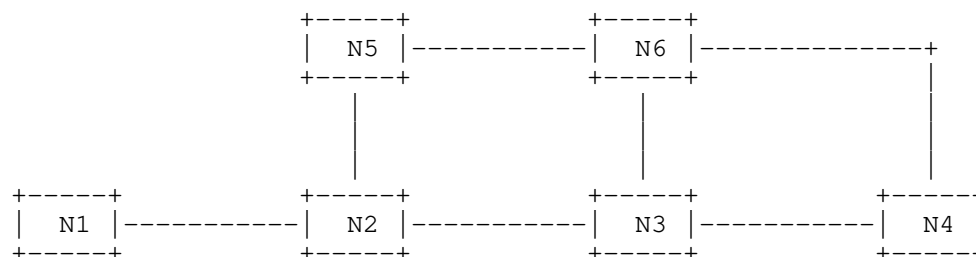


Figure 1: An example of network for midpoint protection

In this document, an end SID at node n with locator block B is represented as $B:n$. An end.x SID at node n towards node k with locator block B is represented as $B:n:k$. A SID list is represented as $\langle S1, S2, S3 \rangle$ where $S1$ is the first SID to visit, $S2$ is the second SID to visit and $S3$ is the last SID to visit along the SRv6 TE path.

In the reference topology, suppose that Node $N1$ is an ingress node of SRv6 TE path going through $N3$ and $N4$. Node $N1$ steers a packet into a segment list $\langle B:3, B:4 \rangle$.

When node $N3$ fails, the packet needs to bypass the failed endpoint node and be forwarded to the next endpoint node after the failed endpoint in the TE path. When outbound interface failure happens in the Repair Node (which is not limited to the previous hop node of the failed endpoint node), it performs the proxy forwarding as follows:

During the first time period (i.e., before the IGP converges), node $N2$ (direct neighbor of $N3$) as a Repair Node forwards the packets around the failed endpoint $N3$ after detecting the failure of the outbound interface to the endpoint $B:3$. It changes the IPv6 destination address with the next sid $B:4$. $N2$ detects the failure of outbound interface to $B:4$ in the current route, it could use the normal Ti-LFA repair path to forward the packet, because it is not directly connected to the node $N4$. $N2$ encapsulates the packet with the segment list $\langle B:5:6 \rangle$ as a repair path.

During the second time period (i.e., after the IGP converges), node $N1$ does not have any route to the failed endpoint $N3$ in its FIB. Node $N1$, as a Repair Node, forwards the packets around the failed endpoint $N3$ to the next endpoint node (e.g., $N4$) directly. There is no need to check whether the failed endpoint node is directly connected to $N1$. $N1$ changes the IPv6 destination address with the next sid $B:4$. Since IGP has completed convergence, it forwards packets directly based on the IGP SPF path

4. SRv6 Midpoint Protection Behavior

A node N protecting the failure of an endpoint node on a SRv6 path may be one of the following types:

- * a transit node: the destination address (DA) of the packet received by N is not N's local SID.
- * an endpoint node: the destination address (DA) of the packet received by N is a N's local END SID.
- * an endpoint x node (i.e., an endpoint with cross-connect node): the destination address (DA) of the packet received by N is a N's local End.X SID with an array of layer 3 adjacencies.

This section describes the behavior of each of these nodes as a repair node for the two time periods after the endpoint node fails.

4.1. Transit Node as Repair Node

When the Repair Node is a transit node, it provides fast protection against the endpoint node failure as follows after looking up the FIB.

```
IF the primary outbound interface used to forward the packet failed
  IF NH = SRH && SL != 0 and
    the failed endpoint is directly connected to Repair Node THEN
    SL decreases*; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    forward the packet according to the backup nexthop;
ELSE IF there is no FIB entry for forwarding the packet THEN
  IF NH = SRH && SL != 0 THEN
    SL decreases*; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    drop the packet;
ELSE
  forward accordingly to the matched entry;
```

*: SL could be decreased by any dedicated value from [1-N], where N is the current value of SL.

4.2. Endpoint Node as Repair Node

When the Repair Node is an endpoint node, it provides fast protections for the failure through executing the following procedure after looking up the FIB for the updated DA.

```
IF the primary outbound interface used to forward the packet failed
  IF NH = SRH && SL != 0 and
    the failed endpoint is directly connected to Repair Node THEN
    SL decreases; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    forward the packet according to the backup nexthop;
ELSE IF there is no FIB entry for forwarding the packet THEN
  IF NH = SRH && SL != 0 THEN
    SL decreases; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    drop the packet;
ELSE
  forward accordingly to the matched entry;
```

4.3. Endpoint x Node as Repair Node

When the Repair Node is an endpoint x node, it provides fast protections for the failure through executing the following procedure after updating DA.

```
IF the layer-3 adjacency interface is down THEN
  FIB lookup on the updated DA;
  IF the primary interface used to forward the packet failed THEN
    IF NH = SRH && SL != 0 and
      the failed endpoint directly connected to Repair Node THEN
      SL decreases; update the IPv6 DA with SRH[SL];
      FIB lookup on the updated DA;
      forward the packet according to the matched entry;
    ELSE
      forward the packet according to the backup nexthop;
  ELSE IF there is no FIB entry for forwarding the packet THEN
    IF NH = SRH && SL != 0 THEN
      SL decreases; update the IPv6 DA with SRH[SL];
      FIB lookup on the updated DA;
      forward the packet according to the matched entry;
    ELSE
      drop the packet;
  ELSE
    forward accordingly to the matched entry;
```

5. Determining whether the Endpoint could Be Bypassed

SRv6 Midpoint Protection provides a mechanism to bypass a failed endpoint. But in some scenarios, some important functions may be implemented in the bypassed failed endpoints that should not be bypassed, such as firewall functionality or In-situ Flow Information Telemetry of a specified path. Therefore, a mechanism is needed to indicate whether an endpoint can be bypassed or not. [I-D.li-rtgwg-enhanced-ti-lfa] provides method to determine whether enable SRv6 midpoint protection or not by defining a "no bypass" flag for the SIDs in IGP.

6. Security Considerations

This section reviews security considerations related to SRv6 Midpoint protection processing discussed in this document. To ensure that the Repair node does not modify the SRH header Encapsulated by nodes outside the SRv6 Domain. Only the segment within the SRH is same domain as the repair node. So it is necessary to check the skipped segment have same block as repair node.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Acknowledgements

9. References

9.1. Normative References

- [I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Segment Routing over IPv6 Dataplane", Work in Progress, Internet-Draft, draft-ietf-lsr-isis-srv6-extensions-18, 20 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-lsr-isis-srv6-extensions-18.txt>>.
- [I-D.ietf-lsr-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", Work in Progress, Internet-Draft, draft-ietf-lsr-ospfv3-srv6-extensions-03, 19 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-lsr-ospfv3-srv6-extensions-03.txt>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7356] Ginsberg, L., Previdi, S., and Y. Yang, "IS-IS Flooding Scope Link State PDUs (LSPs)", RFC 7356, DOI 10.17487/RFC7356, September 2014, <<https://www.rfc-editor.org/info/rfc7356>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <<https://www.rfc-editor.org/info/rfc8986>>.

9.2. Informative References

- [I-D.hu-spring-segment-routing-proxy-forwarding]
Hu, Z., Chen, H., Yao, J., Bowers, C., Yongqing, and Yisong, "SR-TE Path Midpoint Restoration", Work in Progress, Internet-Draft, draft-hu-spring-segment-routing-

proxy-forwarding-15, 24 October 2021,
<<https://www.ietf.org/archive/id/draft-hu-spring-segment-routing-proxy-forwarding-15.txt>>.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", Work in Progress, Internet-Draft, draft-ietf-rtgwg-segment-routing-ti-lfa-07, 29 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-rtgwg-segment-routing-ti-lfa-07.txt>>.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", Work in Progress, Internet-Draft, draft-ietf-spring-segment-routing-policy-14, 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-spring-segment-routing-policy-14.txt>>.

[I-D.li-rtgwg-enhanced-ti-lfa]
Li, C., Hu, Z., Zhu, Y., and S. Hegde, "Enhanced Topology Independent Loop-free Alternate Fast Re-route", Work in Progress, Internet-Draft, draft-li-rtgwg-enhanced-ti-lfa-05, 21 October 2021, <<https://www.ietf.org/archive/id/draft-li-rtgwg-enhanced-ti-lfa-05.txt>>.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", Work in Progress, Internet-Draft, draft-sivabalan-pce-binding-label-sid-07, 8 July 2019, <<https://www.ietf.org/archive/id/draft-sivabalan-pce-binding-label-sid-07.txt>>.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <<https://www.rfc-editor.org/info/rfc5462>>.

Authors' Addresses

Huanan Chen
China Telecom
109, West Zhongshan Road, Tianhe District
Guangzhou
510000
China

Email: chenhuan6@chinatelecom.cn

Zhibo Hu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing
100095
China

Email: huzhibo@huawei.com

Huaimo Chen
Futurewei
Boston, MA,
United States of America

Email: Huaimo.chen@futurewei.com

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Yisong Liu
China Mobile

Email: liuyisong@chinamobile.com

Gyan S. Mishra
Verizon Inc.
13101 Columbia Pike
Silver Spring, MD 20904
United States of America

Phone: 301 502-1347
Email: gyan.s.mishra@verizon.com

INTAREA
Internet-Draft
Intended status: Informational
Expires: 28 April 2022

T. Eckert
Futurewei Technologies USA
N. Shenoy
Rochester Institute of Technology
25 October 2021

Functional Addressing (FA) for internets with Independent Network
Address Spaces (IINAS)
draft-eckert-intarea-functional-addr-internets-01

Abstract

Recent work has raised interest in exploring network layer addressing that is more flexible than fixed-length addressing as used in IPv4 (32 bit) and IPv6 (128 bit).

The reasons for the interest include both support for multiple and potentially novel address semantics, but also optimizations of addressing for existing semantics such as unicast tailored not for the global Internet but to better support private networks / limited domains.

This memo explores in the view of the author yet little explored reasons for more flexible addresses namely the problems and opportunities for Internetworking with Independent Network Address Spaces (IINAS).

To better enable such internetworks, this memo proposes a framework for a Functional Addressing model. This model also intends to support several other addressing goals including programmability and multiple semantics.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Overview	3
1.2. Disclaimer	3
2. Challenges	4
2.1. High level observations	4
2.2. Internetworking limited domain networks with IP addressing	5
2.3. Shorter addresses	9
2.4. Additional semantics	9
2.5. Programmability	10
3. FA-IINAS: Functional Addressing (FA) for Internetworking with Independent Network Address Spaces (IINAS)	10
3.1. Addressing for unicast	11
3.2. Forwarding	12
3.2.1. Dispose Function	12
3.2.2. Steering Function	12
3.2.3. Multiple semantics	12
3.2.4. Internetworking Function	14
3.3. Control Plane	16
3.3.1. Unicast routing	16
3.3.2. Naming	17
3.3.3. Routing	18
3.3.4. Routing policies	19
3.4. Hardware considerations	20
3.4.1. Forwarding plane simplicity	20
3.4.2. Optimizing for smaller networks	21
3.4.3. Maximum address sizes	21
3.5. Example packet header encoding	21
4. Inspirations	22
4.1. E.164	23

4.2. MPLS	25
4.3. Segment Routing SR-MPLS / SRv6	25
4.4. Research	26
5. Summary and conclusions	26
6. Changelog	27
7. Informative References	27
Authors' Addresses	30

1. Introduction

1.1. Overview

Recent work has examined the value of more flexible than fixed-length addressing used in IPv4 (32 bit) and IPv6 (128 bit), see for example [I-D.jia-intarea-scenarios-problems-addressing], and [I-D.jia-flex-ip-address-structure].

The reasons for this interest include both support for multiple and potentially novel address semantics, see for example [I-D.king-irtf-semantic-routing-survey] and [I-D.king-irtf-challenges-in-routing], but also optimizations of addressing for existing semantics, such as unicast, that are tailored not for the global Internet but to better support private networks and limited domains ([RFC8799]).

This memo describes one, in the view of the author yet little explored reason, for more flexible addresses namely the problems and opportunities for Internetworking with Independent Network Address Spaces (IINAS).

To better enable such internetworks, this memo proposes a framework for a Functional Addressing model. This model also intends to support several other addressing model goals including programmability and multiple semantics.

This memo calls the addressing model functional, because addresses are constructed as a structure of
`func1{parameter(s),func2{parameter(s),...i.funcN{parameter(s)}}}`.

1.2. Disclaimer

Any proposals made by this document are explicitly for the purpose of presenting example options of realizing concepts introduced in the memo. There is no intent for any proposals in this document to directly become anything more than just experimental implementations for proof of concept purposes. Equally so or even more so, readers are welcome to pick up any subset of ideas from this memo that they are interested in and reuse it in other designs.

2. Challenges

This section discusses challenges that gave rise to the proposal in this document. It explores in more detail the core challenge not well explored elsewhere and already detailed elsewhere.

2.1. High level observations

There are three core challenges we can observe that limit the ability to build more varied internetworking solutions for non-solely Internet use-cases with especially IPv6:

- * Fixed size address space: IPv4/IPv6 address space is fixed length, not allowing to adopt address length to shorter or longer demands. While it is possible to add more addressing via extension headers, there is no option to not send, or shorten the IPv4/IPv6 base header addresses, when they are not required. While the reasons for fixed size addressing in IPv4/IPv6 can be understood for the feasible high-speed, low-cost forwarders of the 1900th, when IPv6 was conceived, these reasons are today (in the opinion of the author) as obsolete as ATM cells where by the end of the 1990th when both hardware forwarding and mathematical models allowed to provide all ATM type QoS with variable sized packets.
- * The Internet as the primary, if not only use-case driving the design: The address space semantics provided especially by IPv6 is very much focused on the one use-case that drove the development of IPv6: The Internet. While it was and will continue to be the core and sufficient reason for maintaining IPv6, it is not sufficient in the opinion of the author for the much broader use of IPv6. As of today, a likely overwhelming number of hosts using TCP/IP(v6) protocol stacks are not "on the Internet" and the majority likely is not even "connected to the Internet", but instead, they are part of limited domains. This even includes many routers in large service providers that are used to service Internet traffic. Routers in these networks are only in networks that may be called an "underlay" limited domain networks using MPLS, SR-MPLS or SRv6 and Internet traffic is tunneled across them. When the network design is secure, those routers are neither "on" the internet nor "connect to" the Internet.
- * Transparent end-to-end addressing at the core of the IP/IPv6 protocol design, but an ever more diverse reality breaking that design for good reasons: The current core principle of IPv4 and IPv6 is that forwarders have to be passing network layer (IPv4/IPv6) addresses transparently and are not allowed to touch/modifying them. This is the core behavior to support primarily the Internet use case. Yet, the IPv4 Internet today would not

work without NAT, and arguably, the same may also happen to the IPv6 Internet, especially when networks attaching to inexpensive Internet offerings want to avoid complex src/dst forwarding for IPv6 multihoming, and/or avoid renumbering upon change of provider addresses. Even more so, interconnecting IPv4 and IPv6 networks has resulted in no fewer than 24 IPv4/IPv6 NAT solutions (see https://en.wikipedia.org/wiki/IPv6_transition_mechanism), giving rise to the question if and how on-path processing of addressing can be proactively become part of future addressing designs to support more flexible internetworking - translating the best of past NAT experience into better future designs. This is a core option of what FA-IINAS can do.

2.2. Internetworking limited domain networks with IP addressing

One of the core challenges of the existing IP(v4) and IPv6 addressing model are the addressing they provide for private networks with or without connectivity to the Internet, which are also called limited domain networks [RFC8799].

One reference example is that of networking inside a particular product/solution/installation, and then compositing this product with other products, probably even multiple times, hierarchical, as show in picture Figure 1. These type of designs are traditional in industrial networks. Similar issues and solutions can be found in networks with multiple layers of NAT such as Home Networks that are dorm rooms connected via NAT to a dorm network, connected via another NAT to a campus network, connected via yet another NAT to maybe finally, the Internet. Similarly designs can happen with more complex topologies in federated private networks.

In pre-IP industrial networks, individual products were hiding their interior elements by some (combination) of elements that controlled the interior behavior completely and provided only an abstracted view of the machinery to the outside.

With the introduction of IP networking into these type of solutions, the ability for gateways to become IP routers and providing connectivity into the machinery throughout the larger internetwork opened up many important improvements, but of course also challenges, especially for security.

Benefits of network layer internetwork connectivity includes options such as control loops that can more easily be built across multiple components/levels of the hierarchy and controllers that can be pulled out of machinery and positioned elsewhere in the network, enabling virtualization and resource multiplexing. Multiple independently running control systems can be implemented in parallel, including

solutions like device vendor preventive maintenance telemetry, operator managed firmware update or third-party orchestrated security audits or intrusion detection/prevention, just to name a few.

With IP connectivity, all this can be built without the need of understanding how to get through various layers of fixed-functionality higher-than-network layer gateways that can not be extended by third parties. Instead, new designs are based on end-to-end IP connectivity - plus appropriate set of security measures at gateway routers, of course an appropriate set of security/filtering measures, for example MUD, [RFC8520].

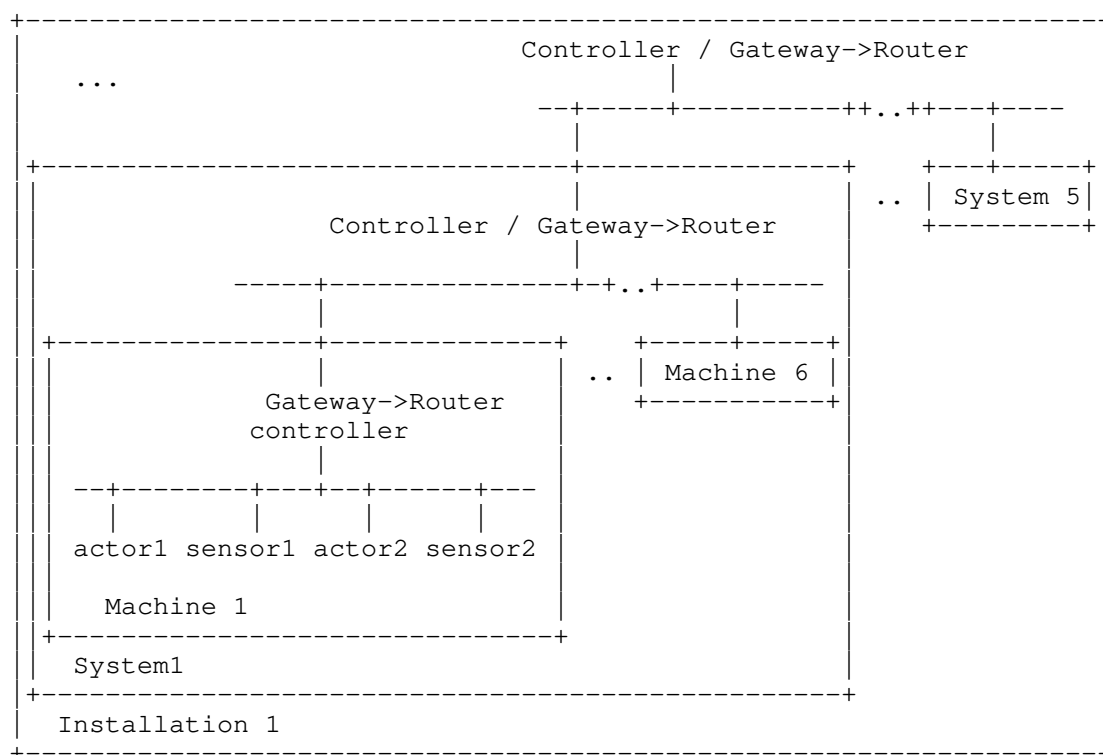


Figure 1: Example hierarchical composed internetwork

In the opinion of the author, the most easily adopted addressing architecture in these type of solutions today is also the one widely used: IPv4 with [RFC1918] addresses. These addresses are actually owned permanently for each deployment case - as long as the scope of addressing is well defined.

In result, a common scheme of addressing in machinery such as the one shown in Figure 1 is to reuse the same 10.0.0.0/8 or 192.168.0.0/16 addresses for every instance of a product/machinery manufactured. In the example, actor1 could use 10.0.0.1, sensor1 10.0.0.2 and so on. But equally, if Machine 3 was the same or similar, its internal components would share the same machinery. And when hundreds of these products are produced, they would all have the same addresses.

To allow deployment and composing those type of machineries, the router/switch connecting to the outside/next-level in a hierarchy will need simple NATing function for example statically mapping the 10.0.0.x on the inside to 10.0.1.x on the outside for Machine 1, where the same router/switch for Machine 3 would be configured to NAT from 10.0.0.x to 10.0.3.x. And likewise at the next layer of hierarchy, 10.0.y.x could be mapped to 10.z.y.x with a different y for every instance.

In support of solutions like this, many if not most industrial ethernet switches deployable as machinery gateways do therefore support this type of static NAT mappings. Likewise, common practices in industries rely on this addressing with composition via NAT approaches, including machineries as large as production lines or in transportation networks train cars and all their included machineries/equipment.

The desire to avoid NAT in IPv6 and availability of sufficient addressing space lead to replacing the concept of [RFC1918] in IPv4 with the concept of Unique Local Addresses (ULA) in IPv6, standardized in [RFC4193]. Instead of the few scoped prefixes of [RFC1918], ULA provide for 2^{40} different prefixes, and the design guidelines are theoretically simple: pick a random prefix and then you can interconnect your networks later on with a very low probability of address prefix collision/reuse.

Unfortunately, low probabilities of address collision is not a good design principle for most of these type of environments because there is really no good operational solution what do if such collision occurs, and rare errors are also very hard to build resilient solutions for. Also the probabilities begin to become much higher when not looking at a connection of just two or few of such ULA networks, but when there can be thousands of such networks, such as in the transportation networks use case.

In result, ULA is not very persuasive for many such deployments, especially when the alternative with IPv4 is address prefix mapping as required for NAT, when NAT an an almost free provisioning side effect of setting up the required connectivity via permit lists via network/transport filters. The need to automate such in-network filtering to secure such deployments can also be seen in the advent of MUD, [RFC8520].

If one considers that most of these subnet networks will have fewer than 253 hosts connected to it, then the IPv6 ULA solution does also not provide for any more bits for subnets than the 16 bits of z.y in the above example using IPv4 10.z.y.x with x being the host part: The lower 64 bits of the IPv6 address is hard to use for anything than the host parts with non-router hosts. The whole ULA prefix is 48 bits, leaving just 16 bit (128 - 64 - 48). Add to that the non insignificant IPv6 packet header overhead plus fewer availability of NAT in IPv6 products because it is assumed to be less required, plus the insufficiency of "low likelihood of collisions" when attempting to utilize only ULA.

Vendors of equipment that have assigned Provider Independent IPv6 address space could of course allocate addressing from that space for equipment they manufacture or integrate, whether it is globally unique or "generic", e.g.: reused across every instance of a product and hence requiring NAT. Unfortunately, and unlike ethernet, where one actually does own addresses after buying an OUI, assigned IPv6 addressing is not permanent, and even though revocation of address allocation is not standard practice, standardized solutions for global IPv6 address space (like IPv4 global address space) really need to allow the ability for those addresses to be returnable instead of being handed off in products to customers.

Even though in hindsight, the hierarchical address allocation from the available 16 bits in 10.x.y.z for two layers of interconnections in the above example looks obvious and simple, in many cases the creation of multiple hierarchies is only an afterthought and the fixed address length and prior suboptimal assignment of addressing in a deployment will cause the need for a lot of re-addressing. This is a recurring problem in larger enterprise/commercial networks under unplanned growth or mergers & acquisitions, especially of course in IPv4. Likewise, once the 16 available bits in the above described NAT approach are used up, whether it is IPv4 or IPv6 with ULA, no further extensions of the design are possible.

2.3. Shorter addresses

As has been noted in prior memos, shorter addresses than IPv6 128 bit are highly desirable in private networks / limited domains whenever it is clear that the total required addressing space is much smaller and connectivity to e.g.: the Internet is not required. Evidence of such requirements can be found for example in header compression for IoT networks such as [RFC6282]. Such compression introduces yet another layer of complexity - the whole ecosystem of devices and diagnostic options has to support it to be equally acceptable as uncompressed packets.

2.4. Additional semantics

New semantics can only be introduced into existing IPv4/IPv6 when their required address size fits nicely into the 32 or 128 bit address space.

This section does not aim to be complete, see [I-D.king-irtf-semantic-routing-survey] for a broader survey. Instead it will provide additional levels of details for the benefits of fittingly sized addresses for few examples, that the author is familiar with.

When ignoring Anycast, IP Multicast is likely the most widely adopted additional semantic added to IPv4. With IPv6, IP Multicast became even more flexible and easy to deploy, because the additional bits of IPv6 addresses allowed to encode additional IP multicast parameters through additional fields in IPv6 addresses: Scope address field [RFC4291], SSM addresses [RFC4607], Unicast prefix multicast addresses [RFC3306] and embedded-RP [RFC3956]. Nevertheless, especially embedded-RP could have benefitted from even longer addresses because with the 128 bits available the solution had to take a hit in the complexity of deployment. It requires to engineer that RP address such that its non-0 host port is very short (4 bits).

In contrast, Bit Indexed Explicit Replication (BIER) which started in the IETF in 2014 and resulted in the architecture [RFC8279], did not choose the option to integrate into IP/IPv6 because it desired addresses sizes of at least per-network configurable from 64 to 4096 bit plus additional qualifiers of at least 16 bits (so-called SD, SI address qualifiers). This made it necessary for BIER to (re-)invent its own network layer packet header, [RFC8296] which duplicates pretty much all packet header fields of MPLS plus IP packets plus additional BIER header fields, so that it can be used in both MPLS and non-MPLS networks.

Similar arguments about the limited size of IPv6 address could likely be made for ICN/CCN networks because the semantic of their addresses is that of data items such as time slices of specific spatial and temporal resolutions of some media such as an audio/video recording - and those name spaces would ideally have addresses as long as URLs.

2.5. Programmability

Segment Routing via IPv6 (SRv6) introduced with [RFC8986] and [RFC8754] (SRH) and architecture in which source routing with an IPv6 extension header is combined with encoding of additional processing semantics into the destination and source routing hops IPv6 addresses. SRv6 calls this programmability.

SRv6 is a very flexible and theoretically extensible concept but challenged by the fixed address length design of IPv6. For most steering hop addresses, the bits reserved for this additional packet processing are not required, but when they are required there may even be too few bits available. Variable length addresses allowing for variable long programming field in the address would in the opinion of the author be highly beneficial.

One evidence for the programmability bits seen as wasteful in many cases is a variety of currently proposed drafts to provide more compressed source routing options for SRv6 (as of mid 2021).

3. FA-IINAS: Functional Addressing (FA) for Internetworking with Independent Network Address Spaces (IINAS)

This section outlines an addressing design that attempts to solve the above described challenges and calls it tentatively FA-IINAS. Functional Addressing refers to the design aspect that addresses in this design can be interpreted as functions with parameters.

Notwithstanding other granularities or options, this document assumes that addresses are textually represented in hexadecimal and that the minimum structure element of an address is 4 bit so that the different structural elements of an address can simply be shown as concatenation of hex digits. The "." character is inserted optionally to show where in an address one semantic part ends and another starts.

Like in IPv6 IoT networks, such as those using RPL ([RFC6550]) as their routing protocol, this memo starts by assuming all nodes are routers and that addresses are predominantly node addresses as opposed to IP/IPv6, which defines unicast addresses to be interface addresses. This is but an academic differentiation, because node addresses can also be represented as interface addresses of so-called "loopback" interfaces.

A network in this design is an independent address space, not shared with other networks. A network has theoretically unlimited long addresses whose prefixes are mapped onto the nodes of the network, which are expected to form a graph of transitively connected nodes. Practical limits to address length are subject to acceptable packetization.

3.1. Addressing for unicast

Each node is assigned one or more node prefixes from the networks address space and none of these node prefixes can be overlapping. In other words, no assigned nodeprefix can be a prefix of another assigned nodeprefix. This rule ensures that every node "owns" any address equal or longer to its assigned nodeprefix. Allocation of node prefixes is currently out of scope for this memo but could rely on any well-known methods including manual operator assigned, SDN controll managed, or as initially described in this document assigned by manufacturer/vendor.

Routing in a network is assumed to enable forwarding across the graph of the network to the node owning the nodeprefix of the address.

Given variable long addresses, the first observation of this addressing scheme is that it allows to combine short addresses with extensibility.

In a simple example the first 200 nodes are assigned addresses 01 ... c8, at which point in time the network operator gets worried about growth exceeding the 256 mark and starts to assign longer addresses: c90 ... f000, at which point in time ever increasing success might cause assignment of even longer prefixes.

Addresses longer than the assigned "nodeprefix" are used to instantiate a specific function on the node itself. A generic representation of an address could be
nodeprefix.function.{parameter}.

3.2. Forwarding

3.2.1. Dispose Function

When using a single digit function field, function = 0 could for example be "dispose" to decapsulate the packets payload and deliver it to the host stack. Parameter could for example be the next-protocol value, eliminating the need to have a separate packet header field for this parameter.

While not being the same crucial issue as for the node prefixes themselves, putting the next-protocol into the address makes it extensible too, so one would not run out of a 256 space as IPv4/IPv6 might do at some point.

3.2.2. Steering Function

Command = 1 could be a "steer" command and the parameter is another address. To act on the command, the node would strip the nodeprefix and command part of the address and forward it based on the address parameter. For example node 73 (e.g.: node with nodeprefix 73) receives a packet with destination address 73.1.55.1.33.0. It forwards the same packet with the stripped destination address 55.1.33.0 to node 55, which likewise forwards the packet with stripped destination address 33.0 to node 33, which ultimately receives it.

3.2.3. Multiple semantics

To introduce additional semantics into a network, such as for example multicasting, we need to generalize how to interpret the first part of the address, which so far was only interpreted to be a nodeprefix for unicast forwarding.

```
address = prefix{.nodefunction{.nodefunction-parameters}}
prefix = semantic{.semantic-parameters}
```

```
semantic / = unicast-forward
```

```
unicast-forward = <set of prefixes>
unicast-forward-parameters = node-prefix
```

```
semantic /= multicast-forward
multicast-forward = <set of prefixes>
multicast-forward-parameters = multicast-group
```

Figure 2

In other words, the prefix at the start of the address is composed of a semantic and its parameter, and the case discussed so far is simply the unicast-forward semantic followed by a node-prefix parameter.

Again, semantic can be an arbitrarily long or short prefix, but no semantic can be a prefix of another semantic.

In a practical example, this scheme is easily applied to existing IPv4 / IPv6 address spaces. For IPv4:

```
unicast-forward = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D
multicast-forward = E
```

Figure 3

In other words, because IP multicast uses addresses 224.0.0.0/4, its non-overlapping semantic prefix is E, and IPv4 unicast addresses use the non-overlapping prefixes 0...D. Assume further that a node in the network had assigned prefix 10.0.0.0/24, then this would translate in our scheme into:

0.A0000.XX

Figure 4

When a node processes this address, the 4-bit prefix 0 indicates that the following prefix has to be looked up in unicast forwarding. This prefix is A0000. Once the packet is delivered to the node, the remaining 8 bit XX can accordingly be interpreted by the node as a nodefunction with parameters.

Likewise, an address 239.1.2.3 would translate into E.F010203, so the first 4-bit E value would indicate that multicast forwarding needs to be applied to the rest of the address, and with IP Multicast forwarding not having further structure (ignoring willfully for simplicity of the example that it does, for example with SSM), all the remainder of the IPv4 address is the multicast-group

In summary, the logic does really only generalize what routers today already do when they do prefix lookups, except for the following core differences:

- * In IPv4/IPv6, the address semantic is hard-coded by IETF standards. In FA-IINAS they are definable by every network.
- * In IPv4/IPv6, there is no notion of nodefunction{.nodefunction-parameters}, only SRv6 has this concept.

In actual IPv4/IPv6 hardware forwarding lookups, one would not do one lookup for the semantic, followed by another lookup for the semantic-parameters for the case of unicast-forward, instead this would be flattened. The same type of flattening would of course be useable in FA-IINAS. Whether or how flattening or other optimizations are feasible for other semantics such as multicast is of course highly semantic and node implementation specific.

3.2.4. Internetworking Function

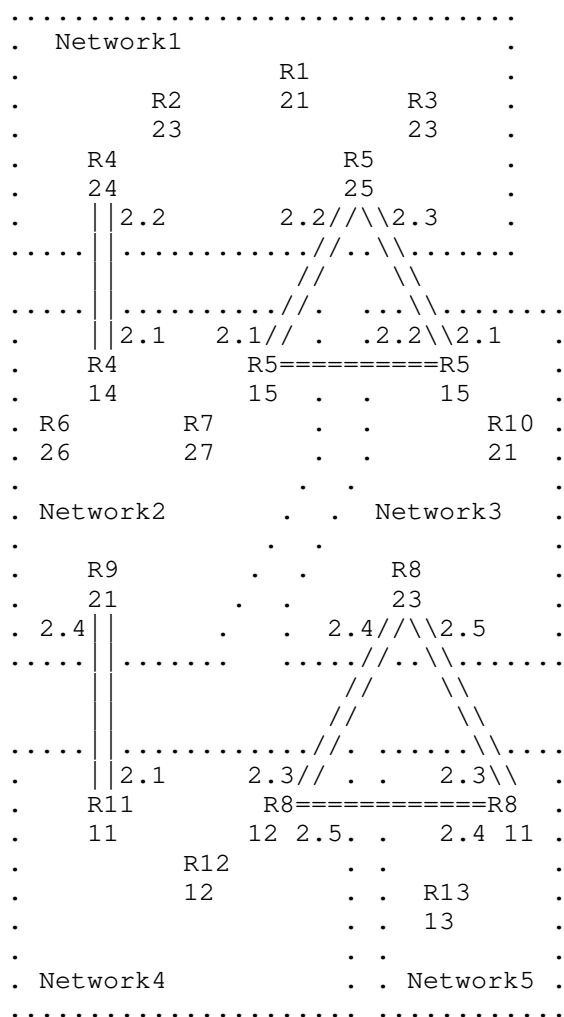


Figure 5: Internetworking example

Figure 5 shows an example internetworking topology of 5 networks, each with its own independent address space. Globally unique Rxx numbers are used to refer to routers.

An edge node is a router that has prefixes from two or more networks into which it connects. In the example, R4 connects into Network1 with prefix 24 and into Network2 with Prefix 14. Likewise, R8 connects into Network3 with prefix 23, into Network4 with prefix 12 and into Network5 with prefix 11. An edge node can be a router simply with different interfaces into different networks, or it can be decomposed into multiple devices, each in a separate network. In this section we describe behavior as if it was a single device.

For an edge node to pass a network into a separate network, the internetworking function on the node has to be called. In the example, this function is codepoint 2 on all edge nodes, and the first parameter is an identifier of local relevance for the network into which to pass the packet. In actual deployment, this function number can of course be locally significant to the Network and/or even each edge router, assuming appropriate control plane to assign the number to this function.

Assume R12 (12) in Network4 wants to send a packet to R1 (21) in Network1. To send it R12->R8->R5->R1, R12 would have to use a destination address of 12.2.3.15.2.1.21.0, or numerically without separators 0x12231521210.

12 will route the packet in Network4 towards R8 because of the destination address 12/8 prefix. .2 indicates to R8 that it should invoke the interworking function and pass the packet into Network 3. As part of the interworking function, R8 then strips all the address prefix it has processed so far from the destination address, leaving 15.2.1.21.0. R8 then forwards the packet with this destination address into Network 3, where it will be received by R5, which again invokes the interworking function due to .2, forwarding the packet into Network1, stripping 15.2.1.0 from the destination address and forwarding the packet with destination address 21.0 into Network1, where it will finally be received by R1 which passes the packet to its host stack because of dispose function 0.

To (optionally) allow for a return path, each edge node could equally but inversely process the source address: When R12 sends the packet, it would indicate a source address of 12.0. When R8 passes the packet via its interworking function into Network3, it would prepend its return path interworking function address, making the source address 23.2.4.12.0, where 23 is R8 address prefix in Network3 and 2.4 interworking function to return the packet into Network4. Likewise, when R5 processes the packet by its interworking function,

it would prepend its return path address element to the source address, before sending the packet into Network1, making the source address 25.2.3.23.2.4.12.0. This is then the address to which R1 could send return packets, and likewise, on its way towards R1, the address, for example when travelling via Network3 always has a returnable source address.

With this behavior of the interworking function, it is obvious, that address management of networks would want to keep a sufficiently large number of very short prefixes, such as those in this example or even shorter to address the interworking function in a sufficiently larger number of edge routers so that a complete internetwork path address will not become too long to exceed the maximum address lengths.

3.3. Control Plane

This section reviews a range of control plane considerations necessary to build a working solution out of the functional addressing. In short, what is required for functions to be flexibly configurable and extensible in the network, it requires a control plane that in its principles is very much based on what was learned in MPLS.

3.3.1. Unicast routing

FA-IINAS expects a control plane that supports routing for unicast-forward parameters (address prefixes) in the same way as it is done today for IPv4/IPv6. Except that it would be for address prefixes (multicast-forward-parameter) of different length and not limited to just 32/128 bits as in IPv4/IPv6.

In addition, FA-IINAS needs control-plane functions that allow defining the semantics and their prefixes, like the above example of 0...D for IPv4 style unicast-forwarding semantic and D for IPv4 style multicast-forwarding semantic.

One of the core challenges for this control plane function is that inconsistency between nodes can have significant different negative impacts than the today accepted "eventual consistency" in IPv4/IPv6 unicast routing that is achieved by the most widely deployed unicast forwarding control planes: distributed routing protocols (IGP/BGP).

The degree of concerns will highly depend on the actual new issues that could happen in the face of inconsistencies, and this can only be vetted with a given set of semantics.

In a most simple example, semantics may simple be configurable via a management plane, and such an approach can be pre-staged, pre-configured, validated network devices, such as in industrial or embedded environments.

In the case of a most flexible, agile type of network, control plane mechanisms would have to be extended to support strong consistency models, for example through node-to-node security associations coupled with a strong consistency network-wide-core-config mechanism. Such mechanisms could in the opinion of the author easily be built on the framework provided by [RFC8994] which provides these hop-by-hop security associations and inband control plane infrastructure, coupled with [RFC8990] as the protocol to negotiate the configuration with strong consistency.

3.3.2. Naming

3.3.2.1. Intra network naming

In FA-IINAS, nodes are acting as routers, and the addresses described are assigned to them persistently. This eliminates in many cases, especially when the network is primarily for m2m communications the need for DNS names, because effectively the address of a node is its persistent name.

In networks small enough, e.g.: maybe $\leq 20,000$ nodes, the very same argument can also apply to nodes that are hosts, e.g. without the need to support full routing/FA-IINAS operations, but still having a persistent address assigned that is routed in the networks routing protocol.

If indeed there is a need to use DNS or other naming schemes, then this is no different than applying naming with DNS to today's [RFC1918] addresses.

3.3.2.2. Simple inter network naming

The need to support (DNS) names is equally lower in interconnected FA-IINAS networks assuming the intra network naming arguments outlined before apply to the interconnected networks.

Because an address in a different FA-IINAS network is dependent on the path from/to its corresponding peer, it is of course not sufficient to simply have a global internetwork name to address mapping.

One of the likely oldest solutions is to align name resolution with packet forwarding so that the very same edge nodes between two networks that do translate addresses can accordingly also translate their name resolution. This was productized and fairly widely deployed as early as the late 1990th for IPv4 with rfc1918 addresses, see for example [CiscoNAT].

This type of solutions relies on well-known routing policies such as simple hierarchical routing though and are not generic for arbitrary topologies.

3.3.3. Routing

3.3.3.1. With internetwork topology knowledge

When FA-IINAS networks are connected in an arbitrary topology instead of a simple hierarchy, the fundamental problem is that of constructing the address of a target peer as a path through a set of appropriate network edge nodes in the address, followed by the nodes address within its network.

In many interconnected FA-IINAS networks, one can assume to have systems that can do this, such as in an industrial setting where a global view of the topology of networks exists and a PCE/SDN-controller will choose the path and can accordingly calculate also the addresses from the path.

3.3.3.2. With internetwork naming knowledge

A decentralized solution can be built by relying on a combination of naming and internetwork routing.

Every network (name space) is assigned a globally unique identifier. This identifier is only used in the control-plane, so it should be reasonably easy to have a set of construction mechanisms allowing everyone to easily create its own namespace, such as for example from some owned location (street address) and/or other owned names/identifier.

When a global naming system like DNS then exists, an FA-IINAS address is the combination of FA-IINAS network identifier and address within that network.

Across the interconnected FA-IINAS networks, the edge-routers would operate extended versions of a protocol like BGP through which any party can calculate desired paths. The extensions would include the FA-IINAS network identifiers and address prefix mapping rules of the edge-nodes, thereby allowing to also calculate addresses from FA-IINAS network identifiers and address.

When large number of small networks (such as users homes) connect to larger networks (such as an ISP), those ISP would be concerned of having to propagate millions of small FA-IINAS network mappings into BGP. This is not done today with IPv4/IPv6, and it would not scale any better with FA-IINAS. Instead, the fact that the home network would be reachable with one or more ISP could be done by also creating naming system mappings from the home networks identifier to the identifier and address prefix mappings of the ISP to which the home network is connected.

When a peer looks up a name and retrieves an FA-IINAS address but cannot find the FA-IINAS network identifier in its internetwork routing information, it can instead resolve it to the "next higher up" ISP FA-IINAS network-identifier/prefix - and recurse this until it has routing information.

Likewise, when a peer does not have any routing information (because it does not participate in internet routing information), it has to forward the appropriate resolution request hierarchically upward.

In summary, it would be architecturally "easy" to extend DNS and BGP with the necessary extensions to resolve names to FA-IINAS addresses and construct relative FA-IINAS addresses from this information.

3.3.4. Routing policies

Note that this "easy" part does not include the possible desire to be more or less flexible in path selection. Whereas today, packets, once they enter "the Internet" are not under steering control of the sender but under "hop by hop hot-potato steering" control of the ISP, with FA-IINAS this may be different - or the same. If a sender then constructed an FA-IINAS address implying an internetwork path that was not desirable for this traffic by the indicated transit networks, this would cause an error. Therefore, the above outlined procedures hinted at relying on the internetwork routing information whenever available and only resort to using naming system to fill in the additional (edge) information.

Today it is becoming more common to use alternative than "native Internet" paths by steering traffic across virtual/container routers in cloud DC, many of which have ample and underutilized international

connectivity. However, additional charges for compute and forwarding will apply. These type of high-overhead solutions could be replaced by FA-IINAS to steer traffic across such additional networks and without the need to instantiate VM/containers. It would require appropriate and lightweight identity and accounting forwarding plane packet header information so that those additional charges could be applied.

3.4. Hardware considerations

3.4.1. Forwarding plane simplicity

Forwarding of FA-IINAS packets based on destination address is the same type of prefix lookup on the destination address as it is today in IPv4/IPv6, except that the maximum lookup prefix can be shorter or longer, this is detailed in the next section.

The steering function should have a lookup complexity whose complexity is in the order of SR-MPLS or even simpler. It can constitute of a prefix lookup in the same forwarding table as non-steered forwarding, but the adjacency would then have to strip the looked up prefix from the destination address (comparable to MPLS label pop) and forward the packet again based on the remainder of the destination address - unless additional on-node service functions have to be invoked.

The interworking function is very much like the steering function, but it also prepends a return prefix to the source address field, making it the most expensive forwarding plane operation.

In general, the author assumes that packet processing that strips a prefix from the destination address and optionally adds a prefix to the source address is well feasible in next generation, highest-speed, lowest-cost forwarding engines.

Optimizations beyond this are possible but would break the independent address allocation across networks. For example, if it is possible for an edge node to have the same prefix length across the networks it connects to, and source address follows destination address in the packet encoding, then stripping the destination address could be achieved by shifting the destination address in a contiguous packet buffer, making head for the source address prefix to be prepended to the following source address field.

3.4.2. Optimizing for smaller networks

One of the benefits of FI-IINAS is that it allows to adopt the address space size based on the requirements of networks and therefore also allows to optimize hardware known to be built/sold only into limited size networks, such as many industrial and almost all embedded networks.

For example, low-cost, high-speed hardware forwarding might be possible to design less expensive with just 16 bit lookups instead of for up to 128 bit lookups, as may be required for IPv6. Equipment could be sold with that profile parameters "for networks with up to 2^{16} nodes".

Because of the way FA-IINAS is designed, a limit to 2^{16} nodes does not mean that FA-IINAS addresses are only 16 bits. Instead they can still be "arbitrary" long (where arbitrary is subject to a discussion point further below in this section). Just the length of the unicast-forward part of the address is limited to 16 bits.

3.4.3. Maximum address sizes

The permissible maximum size of source and destination address are primarily subject to the header size that inexpensive hardware forwarding can examine and modify. For future generations, this might likely be as much as 512 bytes, so to optimize hardware lookup it might be interesting to consider the option of carrying the addresses not consecutively, but carry them as

3.5. Example packet header encoding

The following encodings propose a couple of ideas that could be interesting in addressing.

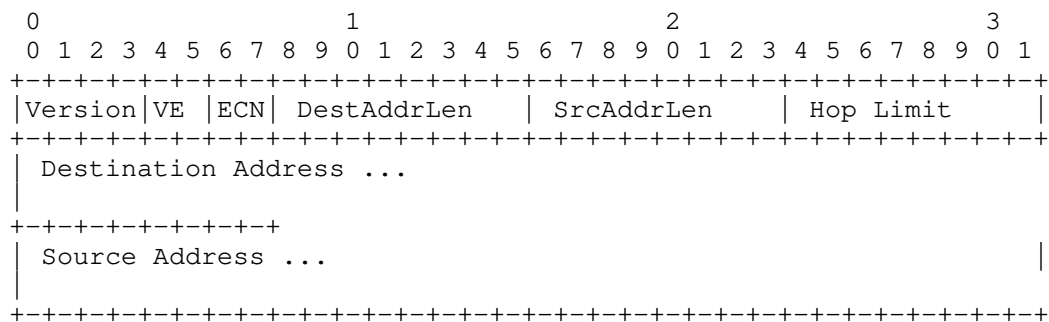


Figure 6: Example packet header encoding

Version: A version number for this packet header from the same registry as the IPv4/IPv6 version number field.

VE: Version Extension. 00. Reserved for future variations of the header, such as new extension header formats if desired, so as to not use up any more than one Version code point.

DestAddrLen: The length of the Destination Address field in bytes. Valid values are 1...255 bytes. One byte minimum length is mandatory because of the need to indicate some semantic for processing the packet.

SrcAddrLen: The length of the Source Address field in bytes. Valid values are 0...255 bytes. The Source Address field is therefore optional.

ECN: See [RFC3168] and the documents updating it.

Rsv: Reserved.

Hop Limit: As in IPv6

Beside the variable length of the Source and Destination address fields and hence their length indications, the difference to the IPv6 header are as follows:

Only the two ECN bits are maintained from the IPv4/IPv6 Traffic Class field. This is because in the majority of networks, the other 6 bits of Traffic Class, DSCP are not being used, and where QoS differentiation would be used, often additional or different QoS parameters may be required that are not supported by IPv4/IPv6. Such a new network header would thus be a great opportunity to improve on QoS header parameters through a better QoS extension header, where it is needed (outside scope of this document), and not proliferate not ubiquitously used elements in the base header. The same reason applies to removing the Flow Label field.

ECN on the other hand is very fundamental for the majority of all traffic in Internet and limited domain networks.

4. Inspirations

This section reviews prior addressing and networking technologies that did inspire this memo and compares it with them.

4.1. E.164

E.164 telephone numbers traditionally worked (and may still work) similar to this mechanisms handling of addresses by adding and removing prefixes and allowing to grow networks hierarchically.

In Germany for example a town/city might have had a subscriber numbering plan starting with 3 digit numbers and expanding over time into 5 digits. 0 was excluded as the first digit of any assigned number. Let our example subscriber have number 1234

When the phone systems of towns/cities where connected, dialing a different town/city would use a concatenation of the inter-city traffic discrimination code "0" followed by the dial code for the town/city, followed by the subscriber number. Let our example town dial code be 4111, the subscriber number dialed from a different city would be 04111 1234. Again, "0" was excluded as the first digit of a trunk prefix.

When finally the phone systems of countries where connected, dialing a different country would use a concatenation of the international traffic discrimination code "00" followed by the country dial code, which in our example is 49 for Germany followed by the dial code for the city, followed by the subscriber number - 0049 4111 1234 for our example subscriber. Note that this number would of course only work when calling from countries that also do use "00" as the international traffic discrimination code. When calling the number from the USA, one would have to dial 011 4111 1234, because the USA uses 011 as the internal traffic discrimination code.

Of course, understanding foreign countries traffic discrimination code rules to reverse engineer a foreign telephone number so as to translate it to the according rules of the calling-from country is one of the problems that is leading more and more subscribers to prefer the absolute E.164 telephone numbers like +49 4111 1234.

On the other hand, when the interplanetary telephone network will "soon" [I-D.draft-farrel-soon] arrive and there are not enough country codes available in Earth's existing numbering plan, one would have to find a way to attach prefixes in front of existing E.164 numbers, something that E.164 likely cannot afford, but which would be possible with UPVLA.

In our example the UPVLA address could be 0003 49 4111 1234 and a new solar system "absolute" address could be ++3 49 4111 1234.

Obviously, Mercury has to get 1, Venus 2 and Earth 3 and so on, so that it would be easier to remember how to dial other planets than it is now to remember how to dial other countries.

If one was to use the solution proposed in this memo to build the phone network addressing system with the example numbering plan, one could set up a multi-tiered internetwork as shown in Figure 7.

Soon:

```

.
. Solar System network .
.
. prefix "3" . |
. .... v strip 3 from dst, prepend 0 to dst
...| Planet Edge Node .... forward into global network
. .... ^ strip 0 from dst, prepend 3 to src
. prefix "0" . | forward into solar system network
.

```

Today:

```

.
. "global" network .
.
. prefix "49" . |
. +-----+ v strip 49 from dst, prepend 0 to dst
...| Country Edge Node |... forward into country network
. +-----+ ^ strip 0 from dst, prepend 49 to src
. prefix "0" . | forward into global network
.
. "country" network .
.
. prefix "4111" . |
. +-----+ v strip 4111 from dst, prepend 0 to dst
...| City Edge Node |... forward into city network
. +-----+ ^ strip 0 from dst, prepend 4111 to src
. prefix "0" . forward into country network
.
. city network .
.
. subscriber 1234 .
.....

```

Figure 7: Example internetwork for E.164 style address structure with FA-INAAS

Packets destined to an address starting with "0" would be routed to an edge node of the city network, which "owns" the "0" prefix, there that "0" prefix is stripped, but the cities own prefix of in the example "4111" is prepended to the source address, and then the packet is forwarded into the country network.

When a packet is received from the country network on a city edge node, the opposite happens, the cities own prefix, e.g.: 4111 is stripped from the destination address and 0 is prepended to the source address, then the packet is forwarded into the city network and routed to the destination. Which can generate return packets by just swapping source and destination addresses.

The same process will happen across 2 network tiers when a 00 prefix is used or even 3 network tiers, once we have (soon ;-) a Solar System Network.

Of course, each tier and each instance of each tier can choose its own addressing scheme and prefixes for the edge routers. It is left as an exercise to the reader for example to amend the example with its own home countries traffic discrimination codes.

4.2. MPLS

Adding/Removing or swapping prefixes is the core forwarding process step in Multiprotocol Label Switching [RFC3031]. Due to the time MPLS was designed, it had to have a very fixed size and functionality stack architecture, but as claimed in before, the author thinks that today an MPLS stack could easily be built just with the proposed addressing scheme address.

Compared to MPLS, the proposed scheme does not mandate that that every steering address needs to contain QoS (EXP) and TTL fields as are present in MPLS Label Stack entries, but of course they would be equally possible as parameters of appropriate functions.

Likewise the proposal does not think it is appropriate to require complicated scanning ahead into the address in search of Special Label Stack entries. Therefore, FA-IINAS would require that any per-hop accessible information that is not included in the hops function/parameters would have to be carried would have to be carried in a separated extensions header.

4.3. Segment Routing SR-MPLS / SRv6

FA-IINAS can support more compact packet steering than SR-MPLS when the prefixes are accordingly chosen to be shorter than the 32 bits for an LSE.

While it would be possible to emulate MPLS LSE by using prefixes of 20 bit and following them with 12 bit of functional parameters indicating EXP and TTL, the proposal in this memo does not assume that transit routers would be able to act on those EXP or TTL bits. While it would be easily possible to define such additional transit hop semantic through extensions to the control plane, the author believes that the per-path parameters of TTL in a base header and more flexible QoS in an extension header is the more likely most useful option for these two functions.

In comparison to SRv6, FA-IINAS allows of course more compact representation of steering hops and also more easily few or many per-hop bits for programmability, as desired.

What FA-IINAS does not provide for is to keep the sequence of steering addresses in the header up to the final receiver. This might be useful for diagnostics, but it is seemingly not so important that it did stop the adoption of SR-MPLS, where the steering hops are likewise removed from the packet header when steering happens.

Other functions than steering and per-steering hop programmability provided by SRv6 via SRH (such as its TLV for the receiver) are unaffected by this proposal and could equally be provided for by an SRH style extension header without the source routing part.

4.4. Research

[Haoyu] proposes a hierarchical addressing scheme and provides reviews in a lot more detail a set of other reasons for such addressing scheme. That paper does not allow for arbitrary composition of networks without a clear hierarchy or root thereof, as FA-IINAS does.

5. Summary and conclusions

This memo introduces a simple but hopefully very attractive addressing scheme that leverages variable length address sizes with the potential for simple address prefix processing (push/pop/swap) on steering hops, service-function hops and especially network edge nodes.

Push/pop/swap of network prefixes on network edge nodes allows to introduce a non-global internetworking address architecture that should make it a lot easier to build and manage many embedded, private or otherwise limited domain internetworks and optimize forwarding engines for a variety of different of these type of networks such as through known maximum network prefix lengths.

When network addresses as in FA-IINAS become effectively internetwork path addresses, they also allow for a much wider range of possible routing policies. In a time where the classical Internet with its "sender just gets one path", this can be a highly beneficial enhancement to explore (not that this was already proposed, maybe way ahead of its time and with vastly different mechanisms in solutions as early as [RFC1621], [RFC1622]).

In this version of the memo, these are only limited considerations about how to refine details of the proposal to find incremental, near-term deployment options, for example by using existing IPv6 headers and using an unassigned prefix to define FA-IINAS addressing semantic into it (limited of course to 128 bit then). These type of considerations can be subject for future revisions of this memo.

6. Changelog

00: Initial version

01: Refresh, new co-author

7. Informative References

[CiscoNAT] Akkiraju, P., Delgadillo, K., and Y. Rekhter, "Enabling Enterprise Multihoming with Cisco IOS Network Address Translation (NAT)", 2000, <http://staff.ustc.edu.cn/~james/cisco/nat/emios_wp.htm>.

[Haoyu] Song, H., Zhang, Z., Qu, Y., and J. Guichard, "Adaptive Addresses for Next Generation IP Protocol in Hierarchical Networks", IEEE 2020 IEEE 28th International Conference on Network Protocols (ICNP), n.d..

[I-D.draft-farrel-soon] Farrel, A., "A Definition of the Term "Soon" for Use in Discussions with Working Group Chairs and Area Directors", Work in Progress, Internet-Draft, draft-farrel-soon-07, 8 March 2021, <<https://www.ietf.org/archive/id/draft-farrel-soon-07.txt>>.

[I-D.jia-flex-ip-address-structure] Jia, Y., Chen, Z., and S. Jiang, "Flexible IP: An Adaptable IP Address Structure", Work in Progress, Internet-Draft, draft-jia-flex-ip-address-structure-00, 31 October 2020, <<https://www.ietf.org/archive/id/draft-jia-flex-ip-address-structure-00.txt>>.

- [I-D.jia-intarea-scenarios-problems-addressing]
Jia, Y., Trossen, D., Iannone, L., Shenoy, N., Mendes, P., 3rd, D. E. E., and P. Liu, "Challenging Scenarios and Problems in Internet Addressing", Work in Progress, Internet-Draft, draft-jia-intarea-scenarios-problems-addressing-02, 23 October 2021, <<https://www.ietf.org/archive/id/draft-jia-intarea-scenarios-problems-addressing-02.txt>>.
- [I-D.king-irtf-challenges-in-routing]
King, D. and A. Farrel, "Challenges for the Internet Routing Infrastructure Introduced by Changes in Address Semantics", Work in Progress, Internet-Draft, draft-king-irtf-challenges-in-routing-03, 14 June 2021, <<https://www.ietf.org/archive/id/draft-king-irtf-challenges-in-routing-03.txt>>.
- [I-D.king-irtf-semantic-routing-survey]
King, D. and A. Farrel, "A Survey of Semantic Internet Routing Techniques", Work in Progress, Internet-Draft, draft-king-irtf-semantic-routing-survey-02, 28 June 2021, <<https://www.ietf.org/archive/id/draft-king-irtf-semantic-routing-survey-02.txt>>.
- [RFC1621] Francis, P., "Pip Near-term Architecture", RFC 1621, DOI 10.17487/RFC1621, May 1994, <<https://www.rfc-editor.org/info/rfc1621>>.
- [RFC1622] Francis, P., "Pip Header Processing", RFC 1622, DOI 10.17487/RFC1622, May 1994, <<https://www.rfc-editor.org/info/rfc1622>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.

- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, DOI 10.17487/RFC3306, August 2002, <<https://www.rfc-editor.org/info/rfc3306>>.
- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, DOI 10.17487/RFC3956, November 2004, <<https://www.rfc-editor.org/info/rfc3956>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <<https://www.rfc-editor.org/info/rfc8799>>.
- [RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <<https://www.rfc-editor.org/info/rfc8986>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.

Authors' Addresses

Toerless Eckert
Futurewei Technologies USA
Santa Clara, CA 95050
United States of America

Email: tte@cs.fau.de

Nirmala Shenoy
Rochester Institute of Technology
New York, NY 14623
United States of America

Email: nxsvks@rit.edu

RTGWG
Internet-Draft
Intended status: Standards Track
Expires: 11 January 2022

D.H. Daniel
B.T. Bin
ZTE Corporation
P.L. Peng
China Mobile
10 July 2021

Computing Delivery in Routing Network
draft-huang-computing-delivery-in-routing-network-00

Abstract

This document drafts a proposal of Computing Delivery in Routing Network which incorporates both computing and networking metrics into the routing policies and enables the network sensing and scheduling computing services based upon traditional networking services. A mechanism of two-class computing power granularity and two segment forwarding is illustrated for end-to-end networking and computing service in the cloud sites, while major networking and computing actors is defined in terms of functionality. An example work flow is demonstrated, and both control plane and data plane solution consideration is proposed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Terminology	3
3. Computing delivery in routing network reference architecture	5
3.1. Hierarchical granularity routing scheme	5
3.2. Two-segment routing and forwarding	6
3.3. CSI routing	7
3.4. Traffic affinity	7
4. Computing delivery in routing network architecture work flow	7
4.1. Computing resource and service update work flow	7
4.2. Service flow routing and forwarding work flow	8
5. Control plane	8
5.1. Centralized control plane	8
5.2. Distributed control plane	9
5.3. Hybrid control plane	9
6. Data plane	9
6.1. CSI encapsulation	9
6.2. CSI for GCR, CUR and LCR	9
7. Summary	10
8. Acknowledgements	10
9. IANA Considerations	10
10. Security Considerations	10
11. Informative References	10
Authors' Addresses	11

1. Introduction

Computing-related services have been provided in such a way that computing resources either are confined within isolated sites (data centers, MECs etc.) without coordination among multiple sites or they are coordinated and managed within specific and closed service systems, while the industry develops into an era in which the computing resources becomes more and more ubiquitous. Therefore substantial benefits in light of both cost and efficiency resulting from scale of economy, would be brought into multiple industries by

intelligently and dynamically connecting the distributed computing resources and rendering the coordinated computing resources as a single virtual resource pool. Although it could be achieved in a routing network-agnostic way as pure application-level solution, additional gains could be reaped with the converged solution of computing and networking routing. Some impressive drafts such as [I-D.liu-dyncast-ps-usecases] and [I-D.li-dyncast-architecture] analyze the benefits of routing related solution, and give the reference architecture and preliminary test results. End applications could be served not only by fine-grained computing services but also fine-grained networking services rather than the best-effort networking services without routing network involved otherwise. The cost is the burden of maintaining and sensing computing resource status in the networking nodes, and it's bear in mind while formulating this proposal and the issue is addressed to a degree the cost could be acceptable. This draft puts forward some considerations of computing delivery in routing network, which proposes a way to optimize routing by two-class computing power granularity and two segments forwarding.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Terminology

- * Global Computing-related Routing Node (GCR): routing node maintaining computing resource as well as service status from across multiple cloud sites, and executing the cross-site routing policies in terms of the aforementioned status as well as the identification of computing resource and service. GCR usually resides at the network edge and works as ingress of the end to end service flow.
- * Local Computing-related Routing Node (LCR): routing node maintaining computing resource as well as service status from the geographically local cloud sites and being responsible for the last hop of the service flow towards the computing resource and service instance in the specific cloud site. LCR usually resides at the network edge and works as egress of the end to end service flow.

- * Computing Unaware Routing Node (CUR): routing node unaware of computing resource and service status and disregarding encapsulation of the identification of computing resource and service. CUR usually resides between GCR and LCR and works as ordinary routing nodes.
- * Global Computing Resource and Service Status (GCRS): General cloud site status of the computing resource and service which consists of overall resource occupation and types of computing service (algorithms, functions etc.) the specific cloud site provides. GCRS is maintained at GCR and expected to remain relatively stable and change in slow frequency.
- * Local Computing Resource and Service Status (LCRS): fine-grained cloud site status of the computing resource and service which consists of status of each active computing service instance as well as its parameters which impact the way the instance would be selected and visited by LCR. LCRS is maintained at LCR and expected to stay quite active and change in high frequency.
- * Computing Service Identification (CSI): a globally unique identification of a computing service with optional parameters, and it could be an IPv6 address or specifically designed address-like structure.
- * Instantiated Computing Service (ICS): an active instance of a computing service identification which resides in a host usually purporting a server, container or virtual machine.

3. Computing delivery in routing network reference architecture

Routing network is enabled sensing the computing resource and service in the cloud sites and routing the application flow according to both network and computing metrics by a computing delivery in routing network architecture as illustrated in figure 1. The architecture is a horizontal convergence of cloud and network, while the latter maintains the converged resource status and thus is able to achieve an end to end routing and forwarding policy from a perspective of cloud and network resource. PE1 maintains GCRS with a whole picture of the multiple cloud sites, and executes the routing policy for the network segment between PE1 and PE2 or PE3, namely between ingress and egress, while PE2 maintains LCRS with a focus picture of the cloud site where S1 resides, and establishes a connection towards S1. S1 is an active instance of a specific computing service type (CSI). On top of the role of LCR which maintains LCRS, PE2 and PE3 also fulfill the role GCR which maintains GCRS from neighboring cloud sites. P provides traditional routing and forwarding functionality for computing service flow, and remains unaware of any computing-related status as well as CSI encapsulations.

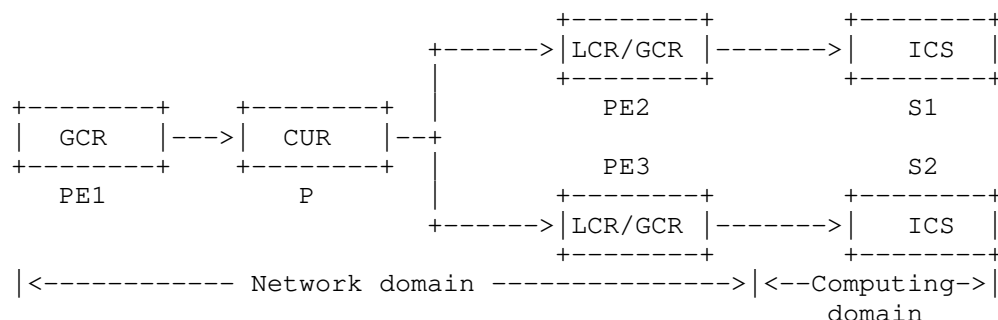


Figure 1

3.1. Hierarchical granularity routing scheme

Status updates of computing resource and service in the cloud sites stay in a quite broad range from relatively stable service types and overall resource occupation to extremely dynamic capacity changes as well as busy and idle cycle of service instance. It would be a disaster to build all of the status updates in the network layer which would bring overburdened and volatile routing tables.

It should be reasonable to divide the wide range of computing resource and services into different categories with differentiated characteristics from routing perspective. GCRS and LCRS correspond to cross-site domain and local site domain respectively, and GCRS aggregates the computing resource and service status with low update frequency from multiple cloud sites while LCRS focuses only upon the status with high frequency in the local sites. Under this two-granularity scheme, computing-related routing table of GCRS in the GCR remains in a position roughly as stable as the traditional routing table, and the LCRS in the LCR maintains a near synchronized state table of the highly dynamic updates of computing service instances in the local cloud site. Nonetheless, LCRS focusing upon a single and local cloud site is the normal case while upon multiple sites should be exemption if not impossible.

3.2. Two-segment routing and forwarding

When it comes to end to end service flow routing and forwarding, there is an status information gap between GCRS and LCRS, therefore a two-segment mechanism has to be in place in line with the two-granularity routing scheme demonstrated in 3.1. As is illustrated in figure 2, R1 ingress determines the specific service flow's egress which turns out to be R2 according to policy calculation from GCRS. In particular, the CSI from either in-band or out-band is the only index for R1 to calculate and determine the egress, it's highly possible to make this egress calculation in terms of both networking (bandwidth, latency etc) and computing Service Agreement Level. Nevertheless, the two SLA routing optimization could be decoupled to such a degree that the traditional routing algorithms could remain as they are. The convergence of the SLA policies as well as the methods to make GCR aware of the two SLA is out of scope of this proposal.

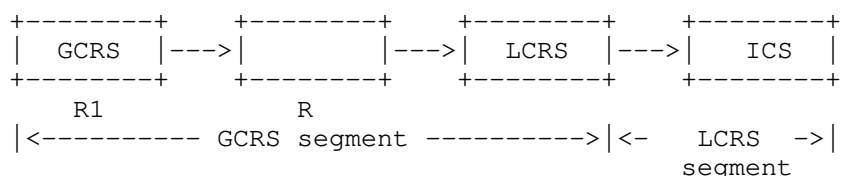


Figure 2

When the service flow arrives at R2 which terminates the GCRS segment routing and determines S1 which is the service instance selected according to LCRS maintained at R2. Again CSI is the only index for LCRS segment routing process.

3.3. CSI routing

CSI encapsulated in the headers and maintained in LCRS and GCRS indicates an abstract service type rather than a geographically explicit destination label, thus the routing scheme based upon CSI is actually a two-part and two-layer process in which CSI only indicates the routing intention of user's requested computing service type where routing does not actually materialize in forwarding plane and the explicit routing destination would be determined by LCRS and GCRS. Therefore the actual routing falls within the traditional routing scheme which remains intact.

3.4. Traffic affinity

CSI holds the only semantics of the service type that could be deployed as multiple instances within specific cloud site or across multiple cloud sites, CSI in the destination field is not explicit enough for all of the service flow packets to be forwarded to a specific destination. Traffic affinity has to be guaranteed at both GCR and LCR. Once the egress is determined at GCR, the binding relationship between the egress and the service flow's unique identification (5-tuple or other specifically designed labels) is maintained and the subsequent flow could be forwarded upon this binding table. Likewise LCR maintains the binding relationship between the service flow identification and the selected service instance.

Traffic affinity could be guaranteed by mechanisms beyond routing layer, but they will not be in the scope of this proposal.

4. Computing delivery in routing network architecture work flow

4.1. Computing resource and service update work flow

The full range of computing resource and service status from a specific cloud site is registered at LCR which maintains LCRS in itself and notifies the part of GCRS to remote GCRs where GCRS would be thus maintained and updated. As is illustrated in figure 3, GCR in R1 from site1 and site 2 is updated by R2 and R3, while LCRS of site 1 in R2 is updated by S1 and LCRS of site 2 in R3 is updated by S2. GCRS in R2 and R3 is updated by each other. Edge routers associating with local cloud site establish a mesh network to update the according GCRS among the whole network domain, the computing resource and services in distributed cloud sites thus are connected and could be utilized as a single pool for the applications rather than the isolated islands.

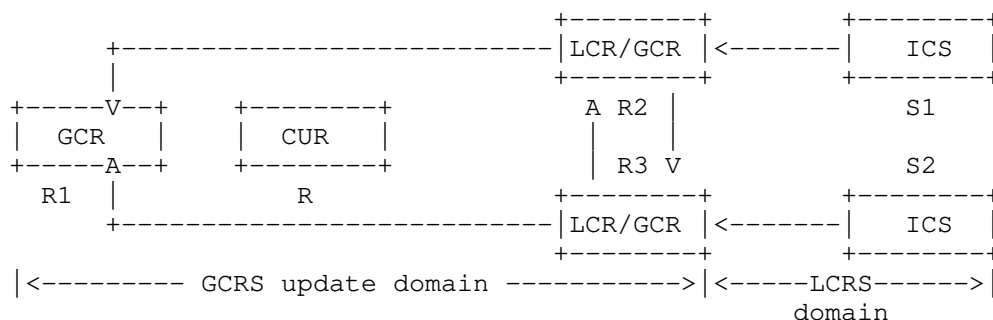


Figure 3

4.2. Service flow routing and forwarding work flow

From perspective of the service work flow, more details have actually been demonstrated in 3.2 and 3.3. Rather than the traditional destination-oriented routing mechanism and the segment routing in which the ingress router is explicitly aware of a specific destination, CSI as an abstract label without semantics of physical address works as the required destination from viewpoint of the user in computing delivery in routing network architecture. Therefore the service flow has to be routed and forwarded segment by segment in which the two segment destinations are determined by GCRS and LCRS respectively.

5. Control plane

5.1. Centralized control plane

LCRS's volatility makes it infeasible to be maintained and controlled in a centralized entity, GCRS is the chief computing resource and service status information to be collected and managed in the controller when it comes to centralized control plane with regard to computing delivery in routing network architecture. Routing and forwarding policies from GCRS calculated in the centralized controller, as is demonstrated in 3.2, apply only to the segment from ingress and egress, while the second segment routing policy from egress to the selected service instance in the cloud site is determined by LCRS at egress.

Hierarchically centralized control plane architecture would be strongly recommended under the circumstances of nationwide network and cloud management.

5.2. Distributed control plane

GCRS is updated among the edge routers which have been connected in a mesh way that each pair of edge routers could exchange GCRS to each other, while LCRS will be unidirectionally updated from cloud site to the associated edge router in which LCRS is maintained and its update process is terminated.

Protocol consideration upon which GCRS and LCRS is updated is out of the scope of this proposal.

5.3. Hybrid control plane

It should be more efficient to update the GCRS by a distributed way than a centralized way in terms of routing request and response in a limited network and cloud domain, but be the opposite case in a nationwide circumstance. This is how hybrid control plane could be deployed in such a scheme that overall optimization is achieved.

6. Data plane

6.1. CSI encapsulation

Computing service identification is the predominant index across the entire computing delivery in routing network architecture under which a new virtual routing scheme is employed with CSI working as the virtual destination. Data plane indicates the routing and forwarding orientation with CSI by inquiring GCRS and LCRS at GCR and LCR respectively. CSI encapsulation could be achieved by extending the existing packet header and also achieved by designing a dedicated shim layer, which along with the specific structure of CSI are out of the scope of this proposal.

6.2. CSI for GCR, CUR and LCR

GCR encapsulates CSI in a designated header format as a proxy by translating the user-originated CSI format, and makes the first segment routing policy and starts routing and forwarding the service traffic. CUR ignores CSI and simply forwards the traffic as usual. LCR decapsulates CSI and makes the second segment routing policy and completes the last hop routing and forwarding.

7. Summary

It would significantly benefit the industry by connecting and coordinating the distributed computing resources and services and more so by further converging networking and computing resource. Uncertainty and the potential impacts over the ongoing network architecture is the main reason for the community to think twice. By classifying the end to end routing and forwarding path into two segments, the impacts from computing status and metrics are to be reduced to a degree they would be as acceptable and comfortable enough as they are as networking status and metrics. In particular, employment of CSI in computing delivery in routing network architecture enables a new service routing possibility perfectly compatible with the ongoing routing architecture.

8. Acknowledgements

To be added upon contributions, comments and suggestions.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

As information originated from the third party (cloud sites), both GCRS and LCRS would be frequently updated in the network domain, both security threats against the routing mechanisms and credibility and security issues of the computing services should be taken into account by architecture designing. The detailed analysis as well as solution consideration will be proposed in the updated version of the draft.

11. Informative References

[I-D.li-dyncast-architecture]

Li, Y., "Dynamic-Anycast Architecture", February 2021,
<<https://datatracker.ietf.org/doc/draft-li-dyncast-architecture/>>.

[I-D.liu-dyncast-ps-usecases]

Liu, Peng., "Dynamic-Anycast (Dyncast) Use Cases and Problem Statement", February 2021,
<<https://datatracker.ietf.org/doc/draft-liu-dyncast-ps-usecases/>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

Authors' Addresses

Daniel Huang
ZTE Corporation
Nanjing

Phone: +86 13770311052
Email: huang.guangping@zte.com.cn

Bin Tan
ZTE Corporation
Nanjing

Phone: +86 13918622159
Email: tan.bin@zte.com.cn

Peng Liu
China Mobile
Beijing

Phone: +86 13810146105
Email: liupengyjy@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 21 October 2022

F. L. Templin, Ed.
G. Saccone
Boeing Research & Technology
G. Dawra
LinkedIn
A. Lindem
V. Moreno
Cisco Systems, Inc.
19 April 2022

A Simple BGP-based Mobile Routing System for the Aeronautical
Telecommunications Network
draft-ietf-rtgwg-atn-bgp-17

Abstract

The International Civil Aviation Organization (ICAO) is investigating mobile routing solutions for a worldwide Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). The ATN/IPS will eventually replace existing communication services with an IP-based service supporting pervasive Air Traffic Management (ATM) for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. This informational document describes a simple and extensible mobile routing service based on industry-standard BGP to address the ATN/IPS requirements.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Terminology	7
3. ATN/IPS Routing System	9
4. ATN/IPS (Radio) Access Network (ANET) Model	14
5. ATN/IPS Route Optimization	16
6. BGP Protocol Considerations	19
7. Stub AS Mobile Routing Services	21
8. Implementation Status	21
9. IANA Considerations	21
10. Security Considerations	21
10.1. Public Key Infrastructure (PKI) Considerations	22
11. Acknowledgements	23
12. References	23
12.1. Normative References	23
12.2. Informative References	24
Appendix A. BGP Convergence Considerations	26
Appendix B. Change Log	26
Authors' Addresses	27

1. Introduction

The worldwide Air Traffic Management (ATM) system today uses a service known as Aeronautical Telecommunications Network based on Open Systems Interconnection (ATN/OSI). The service is used to augment controller to pilot voice communications with rudimentary short text command and control messages. The service has seen successful deployment in a limited set of worldwide ATM domains.

The International Civil Aviation Organization (ICAO) is now undertaking the development of a next-generation replacement for ATN/OSI known as Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) [ATN][ATN-IPS]. ATN/IPS will eventually

provide an IPv6-based [RFC8200] service supporting pervasive ATM for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. As part of the ATN/IPS undertaking, a new mobile routing service will be needed. This document presents an approach based on the Border Gateway Protocol (BGP) [RFC4271].

Aircraft communicate via wireless aviation data links that typically support much lower data rates than terrestrial wireless and wired-line communications. For example, some Very High Frequency (VHF)-based data links only support data rates on the order of 32Kbps and an emerging L-Band data link that is expected to play a key role in future aeronautical communications only supports rates on the order of 1Mbps. Although satellite data links can provide much higher data rates during optimal conditions, like any other aviation data link they are subject to errors, delay, disruption, signal intermittence, degradation due to atmospheric conditions, etc. The well-connected ground domain ATN/IPS network should therefore treat each safety-of-flight critical packet produced by (or destined to) an aircraft as a precious commodity and strive for an optimized service that provides the highest possible degree of reliability. Furthermore, continuous performance-intensive control messaging services such as BGP peering sessions must be carried only over the well-connected ground domain ATN/IPS network and never over low-end aviation data links.

The ATN/IPS is an IP-based overlay network configured over one or more Internetworking underlays ("INETs") maintained by aeronautical network service providers such as ARINC, SITA and Inmarsat. The Overlay Multilink Network Interface (OMNI) [I-D.templin-6man-omni] uses an adaptation layer encapsulation to create a Non-Broadcast, Multiple Access (NBMA) virtual link spanning the entire ATN/IPS. Each aircraft connects to the OMNI link via an OMNI interface configured over the aircraft's underlying physical and/or virtual access network interfaces.

Each underlying INET comprises one or more "partitions" where all nodes within a partition can exchange packets with all other nodes, i.e., the partition is connected internally. There is no requirement that each INET partition uses the same IP protocol version nor has consistent IP addressing plans in comparison with other partitions. Instead, the OMNI link sees each partition as a "segment" of a link-layer topology concatenated by a service known as the OMNI Adaptation Layer (OAL) [I-D.templin-6man-omni] based on IPv6 encapsulation [RFC2473].

The IPv6 addressing architecture provides different classes of addresses, including Global Unicast Addresses (GUAs), Unique Local Addresses (ULAs) and Link-Local Addresses (LLAs) [RFC4291][RFC4193].

The ATN/IPS receives an IPv6 GUA Mobility Service Prefix (MSP) from an Internet assigned numbers authority, and each aircraft will receive a Mobile Network Prefix (MNP) delegation from the MSP that accompanies the aircraft wherever it travels. ATCs and AOCs will likewise receive MNPs, but they would typically appear in static (not mobile) deployments such as air traffic control towers, airline headquarters, etc. (Note that while IPv6 GUAs are assumed for ATN/IPS, IPv4 with public/private address could also be used.)

The adaptation layer uses ULAs in the source and destination addresses of adaptation layer IPv6 encapsulation headers. Each ULA includes an MNP in the interface identifier ("MNP-ULA"), as discussed in [I-D.templin-6man-omni]. Due to MNP delegation policies and random node mobility properties, MNP-ULAs are generally not aggregable in the BGP routing service and are represented as many more-specific prefixes instead of a smaller number of aggregated prefixes.

In addition, BGP routing service infrastructure nodes configure administratively-assigned ULAs ("ADM-ULA") that are statically-assigned and derived from a shorter ADM-ULA prefix assigned to their BGP network partitions. Unlike MNP-ULAs, the ADM-ULAs are persistently present and unchanging in the routing system. The BGP routing services therefore establish forwarding table entries based on these MNP-ULAs and ADM-ULAs instead of based on the GUA MNPs themselves. However, nodes set the 40-bit Global ID and 16-bit Subnet ID to 0 when they advertise MNP-ULAs in BGP routing exchanges and/or install MNP-ULAs in forwarding tables.

Both ADM-ULAs and MNP-ULAs are used by the OAL for nested encapsulation where the inner IPv6 packet is encapsulated in an IPv6 adaptation layer header with ULA source and destination addresses, which is then encapsulated in an IP header specific to the underlying Internetwork that will carry the actual packet transmission. A high level ATN/IPS network diagram is shown in Figure 1:

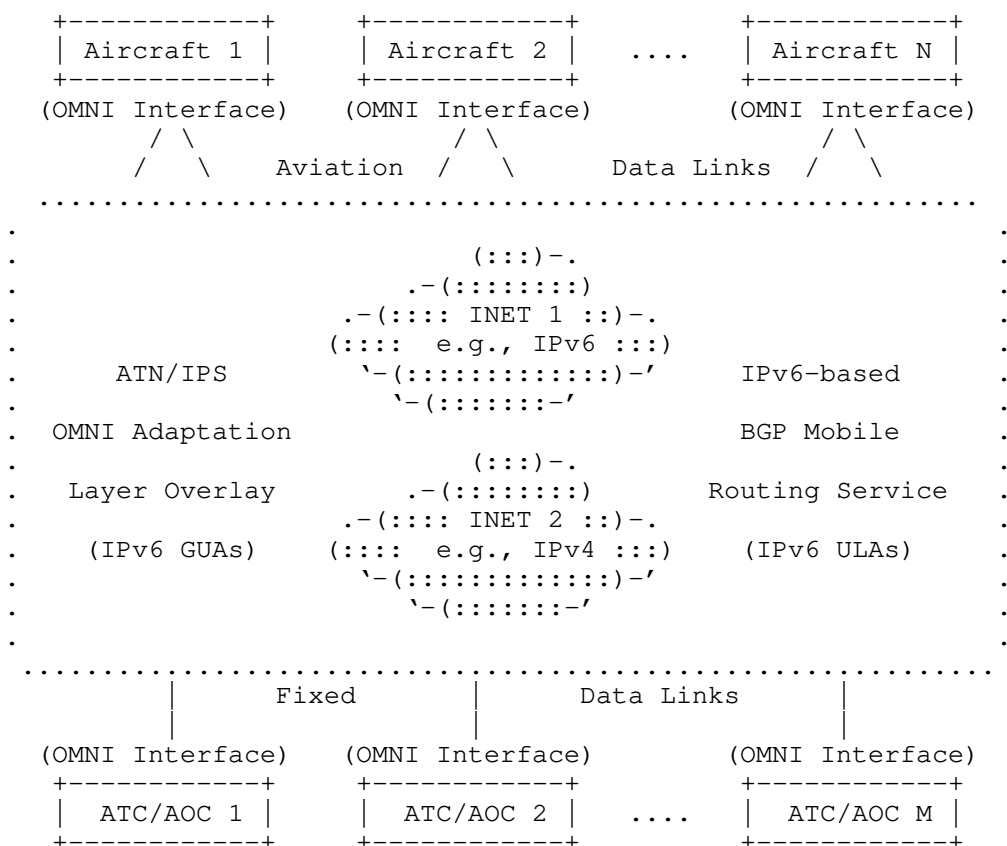


Figure 1: ATN/IPS Network Diagram

Connexion By Boeing [CBB] was an early aviation mobile routing service based on dynamic updates in the global public Internet BGP routing system. Practical experience with the approach has shown that frequent injections and withdrawals of prefixes in the Internet routing system can result in excessive BGP update messaging, slow routing table convergence times, and extended outages when no route is available. This is due to both conservative default BGP protocol timing parameters (see Section 6) and the complex peering interconnections of BGP routers within the global Internet infrastructure. The situation is further exacerbated by frequent aircraft mobility events that each result in BGP updates that must be propagated to all BGP routers in the Internet that carry a full routing table.

We therefore consider an approach using a BGP overlay network routing system where a private BGP routing protocol instance is maintained between ATN/IPS Autonomous System (AS) Border Routers (ASBRs). The private BGP instance does not interact with the native BGP routing systems in underlying INETs, and BGP updates are unidirectional from "stub" ASBRs (s-ASBRs) to a small set of "core" ASBRs (c-ASBRs) in a hub-and-spokes topology. No extensions to the BGP protocol are necessary, and BGP routing is based on (intermediate-layer) ULAs instead of upper- or lower-layer public/private IP prefixes. This allows ASBRs to perform adaptation layer forwarding based on intermediate layer IPv6 header information instead of network layer forwarding based on upper layer IP header information or link layer forwarding based on lower layer IP header information.

The s-ASBRs for each stub AS connect to a small number of c-ASBRs via dedicated high speed links and/or secured tunnels (e.g., IPsec [RFC4301], WireGuard [WG], etc.) over the underlying INET. Neighboring ASBRs should use also such IP layer security encapsulations over direct physical links to ensure INET layer security.

The s-ASBRs engage in external BGP (eBGP) peerings with their respective c-ASBRs, and only maintain routing table entries for the MNP-ULAs currently active within the stub AS. The s-ASBRs send BGP updates for MNP-ULA injections or withdrawals to c-ASBRs but do not receive any BGP updates from c-ASBRs. Instead, the s-ASBRs maintain default routes with their c-ASBRs as the next hop, and therefore hold only partial topology information.

The c-ASBRs connect to other c-ASBRs within the same partition using internal BGP (iBGP) peerings over which they collaboratively maintain a full routing table for all active MNP-ULAs currently in service within the partition. Therefore, only the c-ASBRs maintain a full BGP routing table and never send any BGP updates to s-ASBRs. This simple routing model therefore greatly reduces the number of BGP updates that need to be synchronized among peers, and the number is reduced further still when intradomain routing changes within stub ASes are processed within the AS instead of being propagated to the core. BGP Route Reflectors (RRs) [RFC4456] can also be used to support increased scaling properties.

When there are multiple INET partitions, the c-ASBRs of each partition use eBGP to peer with the c-ASBRs of other partitions so that the full set of ULAs for all partitions are known globally among all of the c-ASBRs. Each c/s-ASBR further configures an ADM-ULA which is taken from an ADM-ULA prefix assigned to each partition, as well as static forwarding table entries for all other OMNI link partition prefixes. Both ADM-ULAs and MNP-ULAs are used by the OAL

for nested encapsulation where the inner IPv6 packet is encapsulated in an IPv6 OAL header with ULA source and destination addresses, which is then encapsulated in an IP header specific to the INET partition.

With these intra- and inter-INET BGP peerings in place, a forwarding plane spanning tree is established that properly covers the entire operating domain. All nodes in the network can be visited using strict spanning tree hops, but in many instances this may result in longer paths than are necessary. AERO [I-D.templin-6man-aero] provides an example service for discovering and utilizing (route-optimized) shortcuts that do not always follow strict spanning tree paths.

The remainder of this document discusses the proposed BGP-based ATN/IPS mobile routing service.

2. Terminology

The terms Autonomous System (AS) and Autonomous System Border Router (ASBR) are the same as defined in [RFC4271].

The following terms are defined for the purposes of this document:

Air Traffic Management (ATM)

The worldwide service for coordinating safe aviation operations.

Air Traffic Controller (ATC)

A government agent responsible for coordinating with aircraft within a defined operational region via voice and/or data Command and Control messaging.

Airline Operations Controller (AOC)

An airline agent responsible for tracking and coordinating with aircraft within their fleet.

Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS)

A future aviation network for ATCs and AOCs to coordinate with all aircraft operating worldwide. The ATN/IPS will be an IPv6-based overlay network service that connects access networks via tunneling over one or more Internetworking underlays.

Internetworking underlay ("INET")

A wide-area network that supports overlay network tunneling and connects Radio Access Networks to the rest of the ATN/IPS. Example INET service providers for civil aviation include ARINC, SITA and Inmarsat.

(Radio) Access Network ("ANET")

An aviation radio data link service provider's network, including radio transmitters and receivers as well as supporting ground-domain infrastructure needed to convey a customer's data packets to outside INETs. The term ANET is intended in the same spirit as for radio-based Internet service provider networks (e.g., cellular operators), but can also refer to ground-domain networks that connect AOCs and ATCs.

partition (or "segment")

A fully-connected internal subnetwork of an INET in which all nodes can communicate with all other nodes within the same partition using the same IP protocol version and addressing plan. Each INET consists of one or more partitions.

Overlay Multilink Network Interface (OMNI)

A virtual layer 2 bridging service that presents an ATN/IPS overlay unified link view even though the underlay may consist of multiple INET partitions. The OMNI virtual link is manifested through nested encapsulation in which original IP packets from the ATN/IPS are first encapsulated in ULA-addressed IPv6 headers which are then forwarded to the next hop using INET encapsulation if necessary. Forwarding over the OMNI virtual link is therefore based on ULAs instead of the original IP addresses. In this way, packets sent from a source can be conveyed over the OMNI virtual link even though there may be many underlying INET partitions in the path to the destination.

OMNI Adaptation Layer (OAL)

A middle layer below the IP layer but above the INET layer that applies IP-in-IPv6 encapsulation prior to INET encapsulation. The IPv6 encapsulation header inserted by the OAL uses ULAs instead of GUAs. End systems that configure OMNI interfaces act as OAL ingress and egress points, while intermediate systems with OMNI interfaces act as OAL forwarding nodes. There may be zero, one or many intermediate nodes between the OAL ingress and egress, but the upper layer IPv6 Hop Limit is not decremented during (OAL layer) forwarding. Further details on OMNI and the OAL are found in [I-D.templin-6man-omni].

OAL Autonomous System (OAL AS)

A "hub-of-hubs" autonomous system maintained through peerings between the core autonomous systems of different OMNI virtual link partitions.

Core Autonomous System Border Router (c-ASBR)

A BGP router located in the hub of the INET partition hub-and-spokes overlay network topology.

Core Autonomous System (Core AS)

The "hub" autonomous system maintained by all c-ASBRs within the same partition.

Stub Autonomous System Border Router (s-ASBR)

A BGP router configured as a spoke in the INET partition hub-and-spokes overlay network topology.

Stub Autonomous System (Stub AS)

A logical grouping that includes all Clients currently associated with a given s-ASBR.

Client

An ATC, AOC or aircraft that connects to the ATN/IPS as a leaf node. The Client could be a singleton host, or a router that connects a mobile or fixed network.

Proxy/Server

An ANET/INET border node that acts as a transparent intermediary between Clients and s-ASBRs. From the Client's perspective, the Proxy/Server presents the appearance that the Client is communicating directly with the s-ASBR. From the s-ASBR's perspective, the Proxy/Server presents the appearance that the s-ASBR is communicating directly with the Client.

Mobile Network Prefix (MNP)

An IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft.

Mobility Service Prefix (MSP)

An aggregated IP prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated (e.g., up to 2^{32} IPv6 /56 MNPs could be delegated from a /24 MSP).

3. ATN/IPS Routing System

The ATN/IPS routing system comprises a private BGP instance coordinated in an overlay network via tunnels between neighboring ASBRs over one or more underlying INETs. The ATN/IPS routing system interacts with underlying INET BGP routing systems only through the static advertisement of a small and unchanging set of MSPs instead of the full dynamically changing set of MNPs.

Within each INET partition, each s-ASBR connects a stub AS to the INET partition core using a distinct stub AS Number (ASN). Each s-ASBR further uses eBGP to peer with one or more c-ASBRs. All c-ASBRs are members of the INET partition core AS, and use a shared

core ASN. Unique ASNs are assigned according to the standard 32-bit ASN format [RFC4271][RFC6793]. Since the BGP instance does not connect with any INET BGP routing systems, the ASNs can be assigned from the [RFC6996] 32-bit ASN space which reserves 94,967,295 numbers for private use. The ASNs must be allocated and managed by an ATN/IPS assigned numbers authority established by ICAO, which must ensure that ASNs are responsibly distributed without duplication and/or overlap.

The c-ASBRs use iBGP to maintain a synchronized consistent view of all active MNP-ULAs currently in service within the INET partition. Figure 2 below represents the reference INET partition deployment. (Note that the figure shows details for only two s-ASBRs (s-ASBR1 and s-ASBR2) due to space constraints, but the other s-ASBRs should be understood to have similar Stub AS, MNP and eBGP peering arrangements.) The solution described in this document is flexible enough to extend to these topologies.

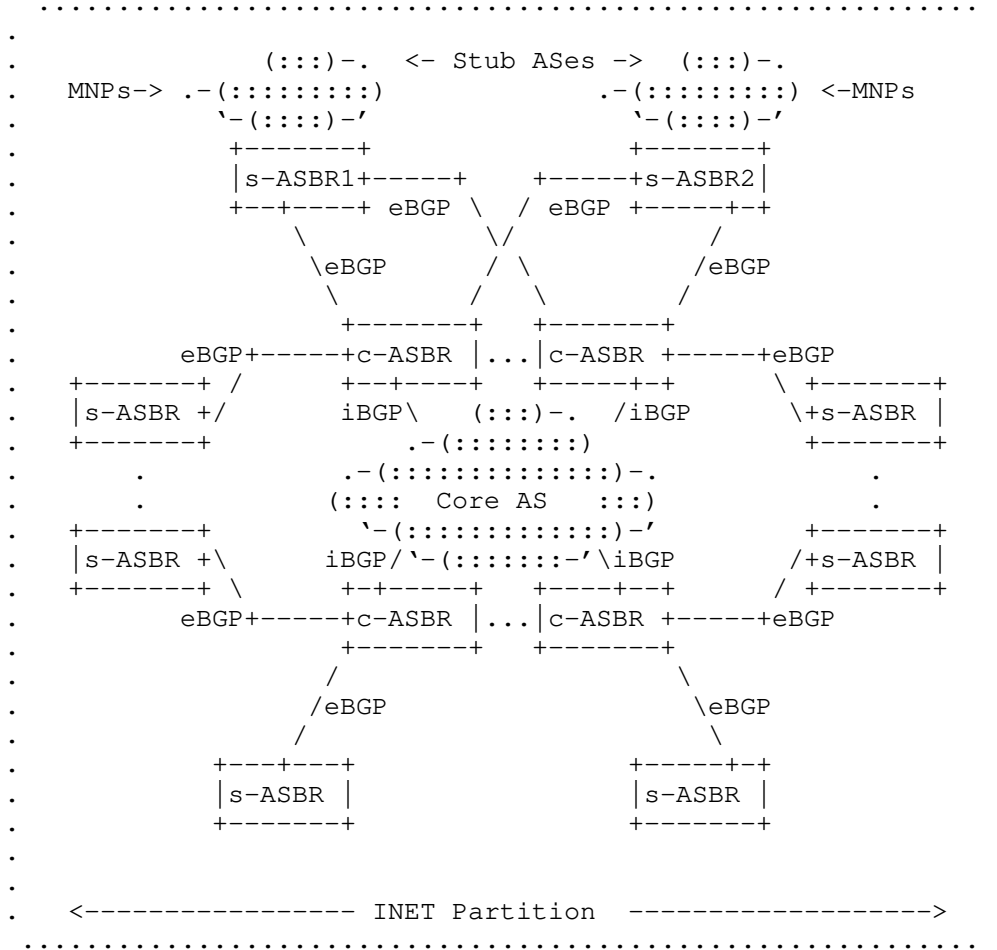


Figure 2: INET Partition Reference Deployment

In the reference deployment, each s-ASBR maintains routes for active MNP-ULAs that currently belong to its stub AS. In response to "Inter-domain" mobility events, each s-ASBR dynamically announces new MNP-ULAs and withdraws departed MNP-ULAs in its eBGP updates to c-ASBRs. Since ATN/IPS end systems are expected to remain within the same stub AS for extended timeframes, however, intra-domain mobility events (such as an aircraft handing off between cell towers) are handled within the stub AS instead of being propagated as inter-domain eBGP updates.

Each c-ASBR configures a black-hole route for each of its MSPs. By black-holing the MSPs, the c-ASBR maintains forwarding table entries only for the MNP-ULAs that are currently active. If an arriving packet matches a black-hole route without matching an MNP-ULA, the c-ASBR should drop the packet and may also generate an ICMPv6 Destination Unreachable message [RFC4443], i.e., without forwarding the packet outside of the ATN/IPS overlay based on a less-specific route.

The c-ASBRs do not send BGP updates for MNP-ULAs to s-ASBRs, but instead originate a default route. In this way, s-ASBRs have only partial topology knowledge (i.e., they know only about the active MNP-ULAs currently within their stub ASes) and they forward all other packets to c-ASBRs which have full topology knowledge.

Each s-ASBR and c-ASBR configures an ADM-ULA that is aggregable within an INET partition, and each partition configures a unique ADM-ULA prefix that is permanently announced into the routing system. The core ASes of each INET partition are joined together through external BGP peerings. The c-ASBRs of each partition establish external peerings with the c-ASBRs of other partitions to form a "core-of-cores" OMNI link AS. The OMNI link AS contains the global knowledge of all MNP-ULAs deployed worldwide, and supports ATN/IPS overlay communications between nodes located in different INET partitions by virtue of OAL encapsulation. OMNI link nodes can then navigate to ASBRs by including an ADM-ULA or directly to an end system by including an MNP-ULA in the destination address of an OAL-encapsulated packet (see: [I-D.templin-6man-aero]). Figure 3 shows a reference OAL topology.

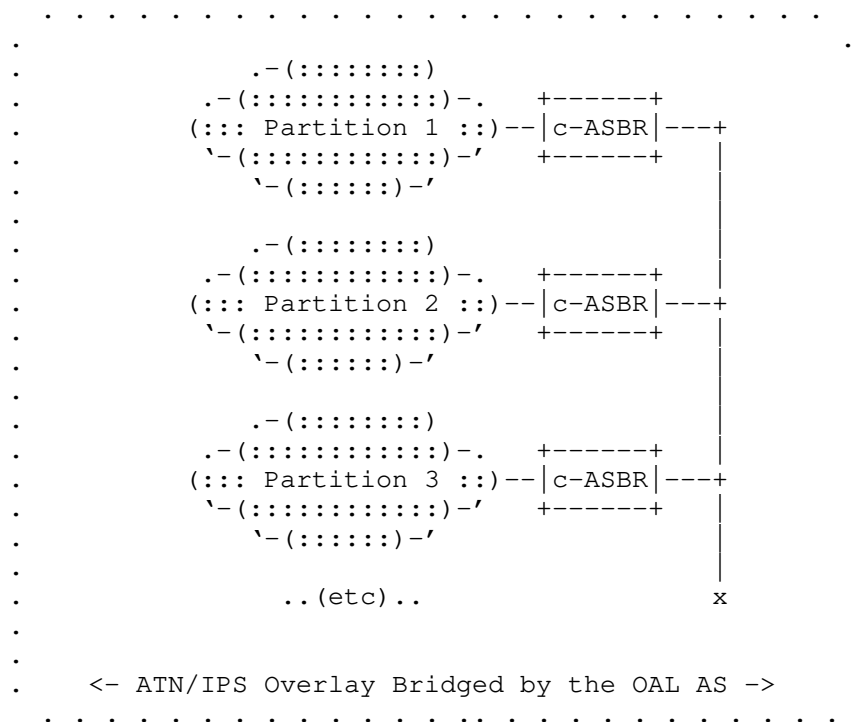


Figure 3: Spanning Partitions with the OAL

Scaling properties of this ATN/IPS routing system are limited by the number of BGP routes that can be carried by the c-ASBRs. A 2015 study showed that BGP routers in the global public Internet at that time carried more than 500K routes with linear growth and no signs of router resource exhaustion [BGP]. A more recent network emulation study also showed that a single c-ASBR can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine. Commercially-available high-performance dedicated router hardware can support many millions of routes.

Therefore, assuming each c-ASBR can carry 1M or more routes, this means that at least 1M ATN/IPS end system MNP-ULAs can be serviced by a single set of c-ASBRs and that number could be further increased by using RRs and/or more powerful routers. Another means of increasing scale would be to assign a different set of c-ASBRs for each set of MSPs. In that case, each s-ASBR still peers with one or more c-ASBRs from each set of c-ASBRs, but the s-ASBR institutes route filters so that it only sends BGP updates to the specific set of c-ASBRs that aggregate the MSP. In this way, each set of c-ASBRs maintains separate routing and forwarding tables so that scaling is distributed

across multiple c-ASBR sets instead of concentrated in a single c-ASBR set. For example, a first c-ASBR set could aggregate an MSP segment A::/32, a second set could aggregate B::/32, a third could aggregate C::/32, etc. The union of all MSP segments would then constitute the collective MSP(s) for the entire ATN/IPS, with potential for supporting many millions of mobile networks or more.

In this way, each set of c-ASBRs services a specific set of MSPs, and each s-ASBR configures MSP-specific routes that list the correct set of c-ASBRs as next hops. This design also allows for natural incremental deployment, and can support initial medium-scale deployments followed by dynamic deployment of additional ATN/IPS infrastructure elements without disturbing the already-deployed base. For example, a few more c-ASBRs could be added if the MNP service demand ever outgrows the initial deployment. For larger-scale applications (such as unmanned air vehicles and terrestrial vehicles) even larger scales can be accommodated by adding more c-ASBRs.

4. ATN/IPS (Radio) Access Network (ANET) Model

(Radio) Access Networks (ANETs) connect end system Clients such as aircraft, ATCs, AOCs etc. to the ATN/IPS routing system. Clients may connect to multiple ANETs at once, for example, when they have both satellite and cellular data links activated simultaneously. Clients configure an Overlay Multilink Network (OMNI) Interface [I-D.templin-6man-omni] over their underlying ANET interfaces as a connection to an NBMA virtual link (manifested by the OAL) that spans the entire ATN/IPS. Clients may further move between ANETs in a manner that is perceived as a network layer mobility event. Clients could therefore employ a multilink/mobility routing service such as those discussed in Section 7.

Clients register all of their active data link connections with their serving s-ASBRs as discussed in Section 3. Clients may connect to s-ASBRs either directly, or via a Proxy/Server at the ANET/INET boundary.

Figure 4 shows the ATN/IPS ANET model where Clients connect to ANETs via aviation data links. Clients register their ANET addresses with a nearby s-ASBR, where the registration process may be brokered by a Proxy/Server at the edge of the ANET.

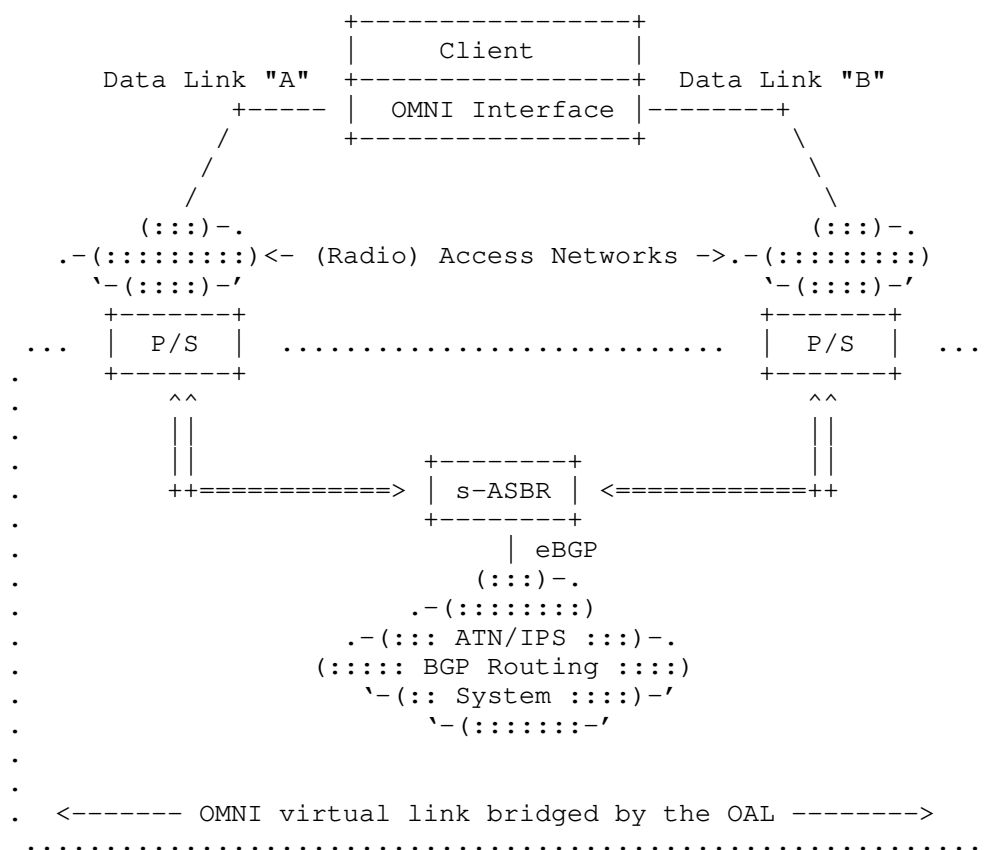


Figure 4: ATN/IPS ANET Architecture

When a Client connects to an ANET it specifies a nearby s-ASBR that it has selected to connect to the ATN/IPS. The login process is transparently brokered by a Proxy/Server at the border of the ANET which then conveys the connection request to the s-ASBR via tunneling across the OMNI virtual link. Each ANET border Proxy/Server is also equally capable of serving in the s-ASBR role so that a first on-link Proxy/Server can be selected as the s-ASBR while all others perform the Proxy/Server role in a hub-and-spokes arrangement. An on-link Proxy/Server is selected to serve the s-ASBR role when it receives a control message from a Client requesting that service.

The Client can coordinate with a network-based s-ASBR over additional ANETs after it has already coordinated with a first-hop Proxy/Server over a first ANET. If the Client connects to multiple ANETs, the s-ASBR will register the individual ANET Proxy/Servers as conduits through which the Client can be reached. The Client then sees the

s-ASBR as the "hub" in a "hub-and-spokes" arrangement with the first-hop Proxy/Servers as spokes. Selection of a network-based s-ASBR is through the discovery methods specified in relevant mobility and virtual link coordination specifications (e.g., see AERO [I-D.templin-6man-aero] and OMNI [I-D.templin-6man-omni]).

The s-ASBR represents all of its active Clients as MNP-ULA routes in the ATN/IPS BGP routing system. The s-ASBR's stub AS is therefore used only to advertise the set of MNPs of all its active Clients to its BGP peer c-ASBRs and not to peer with other s-ASBRs (i.e., the stub AS is a logical construct and not a physical one). The s-ASBR injects the MNP-ULAs of its active Clients and withdraws the MNP-ULAs of its departed Clients via BGP updates to c-ASBRs, which further propagate the MNP-ULAs to other c-ASBRs within the OAL AS. Since Clients are expected to remain associated with their current s-ASBR for extended periods, the level of MNP-ULA injections and withdrawals in the BGP routing system will be on the order of the numbers of network joins, leaves and s-ASBR handovers for aircraft operations (see: Section 6). It is important to observe that fine-grained events such as Client mobility and Quality of Service (QoS) signaling are coordinated only by Proxies and the Client's current s-ASBRs, and do not involve other ASBRs in the routing system. In this way, intradomain routing changes within the stub AS are not propagated into the rest of the ATN/IPS BGP routing system.

5. ATN/IPS Route Optimization

ATN/IPS end systems will frequently need to communicate with correspondents associated with other s-ASBRs. In the BGP peering topology discussed in Section 3, this can initially only be accommodated by including multiple extraneous hops and/or spanning tree segments in the forwarding path. In many cases, it would be desirable to establish a "short cut" around this "dogleg" route so that packets can traverse a minimum number of tunneling hops across the OMNI virtual link. ATN/IPS end systems could therefore employ a route optimization service according to the mobility service employed (see: Section 7).

Each s-ASBR provides designated routing services for only a subset of all active Clients, and instead acts as a simple Proxy/Server for other Clients. As a designated router, the s-ASBR advertises the MNPs of each of its active Clients into the ATN/IPS routing system and provides basic (unoptimized) forwarding services when necessary. An s-ASBR could be the first-hop ATN/IPS service access point for some, all or none of a Client's underlying interfaces, while the Client's other underlying interfaces employ the Proxy/Server function of other s-ASBRs. Route optimization allows Client-to-Client communications while bypassing s-ASBR designated routing services whenever possible.

A route optimization example is shown in Figure 5 and Figure 6 below. In the first figure, multiple spanning tree segments between Proxy/Servers and ASBRs are necessary to convey packets between Clients associated with different s-ASBRs. In the second figure, the optimized route tunnels packets directly between Proxy/Servers without involving the ASBRs.

These route optimized paths are established through secured control plane messaging (i.e., over secured tunnels and/or using higher-layer control message authentications) but do not provide lower-layer security for the data plane. Data communications over these route optimized paths should therefore employ higher-layer security.

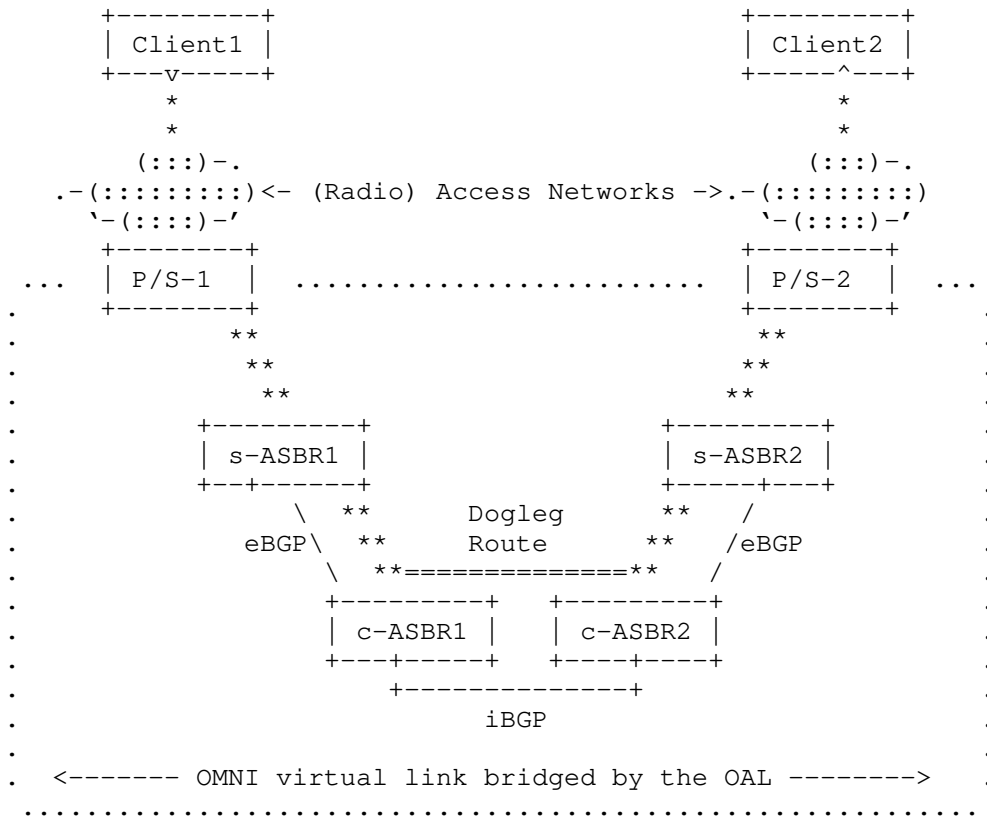


Figure 5: Dogleg Route Before Optimization

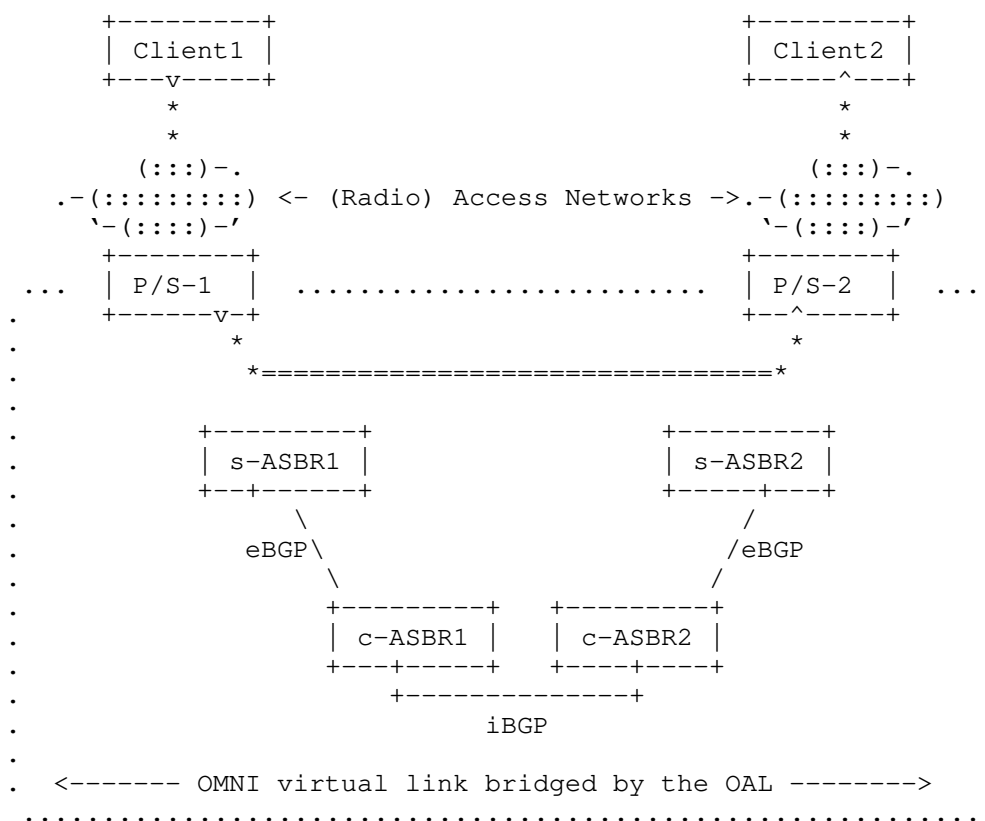


Figure 6: Optimized Route

6. BGP Protocol Considerations

The number of eBGP peering sessions that each c-ASBR must service is proportional to the number of s-ASBRs in its local partition. Network emulations with lightweight virtual machines have shown that a single c-ASBR can service at least 100 eBGP peerings from s-ASBRs that each advertise 10K MNP-ULA routes (i.e., 1M total). It is expected that robust c-ASBRs can service many more peerings than this - possibly by multiple orders of magnitude. But even assuming a conservative limit, the number of s-ASBRs could be increased by also increasing the number of c-ASBRs. Since c-ASBRs also peer with each other using iBGP, however, larger-scale c-ASBR deployments may need to employ an adjunct facility such as BGP Route Reflectors (RRs) [RFC4456].

The number of aircraft in operation at a given time worldwide is likely to be significantly less than 1M, but we will assume this number for a worst-case analysis. Assuming a worst-case average 1 hour flight profile from gate-to-gate with 10 service region transitions per flight, the entire system will need to service at most 10M BGP updates per hour (2778 updates per second). This number is within the realm of the peak BGP update messaging seen in the global public Internet today [BGP2]. Assuming a BGP update message size of 100 bytes (800bits), the total amount of BGP control message traffic to a single c-ASBR will be less than 2.5Mbps which is a nominal rate for modern data links.

Industry standard BGP routers provide configurable parameters with conservative default values. For example, the default hold time is 90 seconds, the default keepalive time is 1/3 of the hold time, and the default MinRouteAdvertisementInterval is 30 seconds for eBGP peers and 5 seconds for iBGP peers (see Section 10 of [RFC4271]). For the simple mobile routing system described herein, these parameters can be set to more aggressive values to support faster neighbor/link failure detection and faster routing protocol convergence times. For example, a hold time of 3 seconds and a MinRouteAdvertisementInterval of 0 seconds for both iBGP and eBGP.

Instead of adjusting BGP default time values, BGP routers can use the Bidirectional Forwarding Detection (BFD) protocol [RFC5880] to quickly detect link failures that don't result in interface state changes, BGP peer failures, and administrative state changes. BFD is important in environments where rapid response to failures is required for routing reconvergence and, hence, communications continuity.

Each c-ASBR will be using eBGP both in the ATN/IPS and the INET with the ATN/IPS unicast IPv6 routes resolving over INET routes. Consequently, c-ASBRs and potentially s-ASBRs will need to support separate local ASes for the two BGP routing domains and routing policy or assure routes are not propagated between the two BGP routing domains. From a conceptual, operational and correctness standpoint, the implementation should provide isolation between the two BGP routing domains (e.g., separate BGP instances).

ADM-ULAs and MNP-ULAs begin with fd00::/8 followed by a pseudo-random 40-bit global ID to form the prefix [ULA]::/48, along with a 16-bit Subnet ID '*' to form the prefix [ULA*]::/64. Each individual address taken from [ULA*]::/64 includes additional routing information in the interface identifier. For example, for the MNP 2001:db8:1:0::/56, the resulting MNP-ULA is [ULA*]:2001:db8:1:0/120, and for the administrative address 1001:2002 the ADM-ULA is [ULA*]:1001:2002/64 (see: [I-D.templin-6man-omni] for further

details). However, MNP-ULA prefixes installed in the BGP routing system always set the Global ID and Subnet ID to 0 (i.e., the "wildcard" subnet) since OMNI link forwarding decisions are based solely on the MNP found in the interface identifier independently of the Global/Subnet IDs.

This gives rise to a BGP routing system that must accommodate large numbers of long and non-aggregable MNP-ULA prefixes as well as moderate numbers of long and semi-aggregable ADM-ULA prefixes. The system is kept stable and scalable through the s-ASBR / c-ASBR hub-and-spokes topology which ensures that mobility-related churn is not exposed to the core.

7. Stub AS Mobile Routing Services

Stub ASes maintain intradomain routing information for mobile node clients, and are responsible for all localized mobility signaling without disturbing the BGP routing system. Clients can enlist the services of a candidate mobility service such as Mobile IPv6 (MIPv6) [RFC6275], LISP [I-D.ietf-lisp-rfc6830bis] or AERO [I-D.templin-6man-aero] according to the service offered by the stub AS. Further details of mobile routing services are out of scope for this document.

8. Implementation Status

The BGP routing topology described in this document has been modeled in realistic network emulations showing that at least 1 million MNP-ULAs can be propagated to each c-ASBR even on lightweight virtual machines. No BGP routing protocol extensions need to be adopted.

9. IANA Considerations

This document does not introduce any IANA considerations.

10. Security Considerations

ATN/IPS ASBRs on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, ASBRs should employ physical security and/or IP securing mechanisms such as IPsec [RFC4301], WireGuard [WG], etc.

ATN/IPS ASBRs present targets for Distributed Denial of Service (DDoS) attacks. This concern is no different than for any node on the open Internet, where attackers could send spoofed packets to the node at high data rates. This can be mitigated by connecting ATN/IPS ASBRs over dedicated links with no connections to the Internet and/or when ASBR connections to the Internet are only permitted through well-managed firewalls.

ATN/IPS s-ASBRs should institute rate limits to protect low data rate aviation data links from receiving DDoS packet floods.

BGP protocol message exchanges and control message exchanges used for route optimization must be secured to ensure the integrity of the system-wide routing information base. Security is based on IP layer security associations between peers which ensure confidentiality, integrity and authentication over secured tunnels (see above). Higher layer security protection such as TCP-AO [RFC5926] is therefore optional, since it would be redundant with the security provided at lower layers.

Data communications over route optimized paths should employ end-to-end higher-layer security since only the control plane and unoptimized paths are protected by lower-layer security. End-to-end higher-layer security mechanisms include QUIC-TLS [RFC9001], TLS [RFC8446], DTLS [RFC6347], SSH [RFC4251], etc. applied in a manner outside the scope of this document.

This document does not include any new specific requirements for mitigation of DDoS.

10.1. Public Key Infrastructure (PKI) Considerations

In development of the overall ATN/IPS operational concept, ICAO addressed the security concerns in multiple ways to ensure coordination and consistency across the various groups. This also avoided potential duplicative work. Technical provisions related specifically to the operation of ATN/IPS are specified in supporting ATN/IPS standards. However, other considerations such as the establishment of a PKI, were determined to have an impact beyond ATN/IPS. ICAO created a Trust Framework Study Group (TFSG) to define various governance, policy, procedures and overall technical performance requirements for system connectivity and interoperability.

As part of their charter, the TSFG is specifically developing a concept of operations for a common aviation digital trust framework and principles to facilitate an interoperable secure, cyber resilient and seamless exchange of information in a digitally connected

environment. They are also developing governance principles, policy, procedures and requirements for establishing digital identity for a global trust framework that will consider any exchange of information among users of the aviation ecosystem, and to promote these concepts with all relevant stakeholders.

ATN/IPS will take advantage of the developments of TFSG within the overall ATN/IPS operational concept. As such, this will include the usage of the PKI specification resulting from the TFSG.

11. Acknowledgements

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the Boeing Commercial Airplanes (BCA) Internet of Things (IoT) and autonomy programs.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

The following individuals contributed insights that have improved the document: Ahmad Amin, Mach Chen, Russ Housley, Erik Kline, Hubert Kuenig, Tony Li, Gyan Mishra, Alexandre Petrescu, Dave Thaler, Pascal Thubert, Michael Tuxen, Tony Whyman.

12. References

12.1. Normative References

- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.

- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

12.2. Informative References

- [ATN] Maiolla, V., "The OMNI Interface - An IPv6 Air/Ground Interface for Civil Aviation, IETF Liaison Statement #1676, <https://datatracker.ietf.org/liaison/1676/>", 3 March 2020.
- [ATN-IPS] WG-I, ICAO., "ICAO Document 9896 (Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocol), Draft Edition 3 (work-in-progress)", 10 December 2020.
- [BGP] Huston, G., "BGP in 2015, <http://potaroo.net>", January 2016.
- [BGP2] Huston, G., "BGP Instability Report, <http://bgpupdates.potaroo.net/instability/bgpupd.html>", May 2017.
- [CBB] Dul, A., "Global IP Network Mobility using Border Gateway Protocol (BGP), http://www.quark.net/docs/Global_IP_Network_Mobility_using_BGP.pdf", March 2006.

- [I-D.ietf-lisp-rfc6830bis]
Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, "The Locator/ID Separation Protocol (LISP)", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6830bis-36, 18 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-lisp-rfc6830bis-36.txt>>.
- [I-D.templin-6man-aero]
Templin, F. L., "Automatic Extended Route Optimization (AERO)", Work in Progress, Internet-Draft, draft-templin-6man-aero-42, 9 April 2022, <<https://www.ietf.org/archive/id/draft-templin-6man-aero-42.txt>>.
- [I-D.templin-6man-omni]
Templin, F. L., "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", Work in Progress, Internet-Draft, draft-templin-6man-omni-57, 9 April 2022, <<https://www.ietf.org/archive/id/draft-templin-6man-omni-57.txt>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <<https://www.rfc-editor.org/info/rfc4251>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5926] Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, DOI 10.17487/RFC5926, June 2010, <<https://www.rfc-editor.org/info/rfc5926>>.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<https://www.rfc-editor.org/info/rfc6275>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.

- [RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <<https://www.rfc-editor.org/info/rfc6793>>.
- [RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, DOI 10.17487/RFC6996, July 2013, <<https://www.rfc-editor.org/info/rfc6996>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <<https://www.rfc-editor.org/info/rfc9001>>.
- [WG] Donenfeld, J., "WireGuard: Fast, Modern, Secure VPN Tunnel, <https://www.wireguard.com/>", February 2022.

Appendix A. BGP Convergence Considerations

Experimental evidence has shown that BGP convergence time required after an MNP-ULA is asserted at a new location or withdrawn from an old location can be several hundred milliseconds even under optimal AS peering arrangements. This means that packets in flight destined to an MNP-ULA route that has recently been changed can be (mis)delivered to an old s-ASBR after a Client has moved to a new s-ASBR.

To address this issue, the old s-ASBR can maintain temporary state for a "departed" Client that includes an OAL address for the new s-ASBR. The OAL address never changes since ASBRs are fixed infrastructure elements that never move. Hence, packets arriving at the old s-ASBR can be forwarded to the new s-ASBR while the BGP routing system is still undergoing reconvergence. Therefore, as long as the Client associates with the new s-ASBR before it departs from the old s-ASBR (while informing the old s-ASBR of its new location) packets in flight during the BGP reconvergence window are accommodated without loss.

Appendix B. Change Log

<< RFC Editor - remove prior to publication >>

Differences from earlier versions:

* Submit for RFC publication.

Authors' Addresses

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
United States of America
Email: fltemplin@acm.org

Greg Saccone
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
United States of America
Email: gregory.t.saccone@boeing.com

Gaurav Dawra
LinkedIn
United States of America
Email: gdawra.ietf@gmail.com

Acee Lindem
Cisco Systems, Inc.
United States of America
Email: acee@cisco.com

Victor Moreno
Cisco Systems, Inc.
United States of America
Email: vimoreno@cisco.com

RTG Working Group
Internet Draft
Intended status: Informational
Expires: October 14, 2022

K. Majumdar
CommScope
U. Chunduri
Intel
L. Dunbar
Futurewei
April 14, 2022

Extension of Transport Aware Mobility in Data Network
draft-mcd-rtgwg-extension-tn-aware-mobility-04

Abstract

The existing Transport Network Aware Mobility for 5G [TN-AWARE-MOBILITY] draft specifies a framework for mapping the 5G mobile systems Slice and Service Types (SSTs) to corresponding underlying network paths in IP and Layer 2 Transport networks. The focus of that work is limited to the mobility domain and transport network characteristics till the UPF and doesn't go beyond the UPF to the Data Network.

To maintain E2E transport network characteristics the framework needs to be extended beyond UPF. This document describes a framework for extending the mobility aware transport network characteristics from the UPF through the Data Network.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 23, 2021.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction.....	3
2. Conventions used in this document.....	4
3. Framework for Extension of Transport Network Aware Mobility....	4
4. Mobility Packet Transition to the Data Network.....	5
5. Transport Network Characteristics Mapping to SR-TE Paths.....	7
5.1. Extend TN Aware Mobility for BGP SR-TE Policy.....	9
5.2. Extend TN Aware Mobility for SR-PCE Controller.....	13
5.3. Extend TN Aware Mobility for RestConf/gRPC based SR-TE Controller.....	16
5.4. Extend BGP FlowSpec for TN Aware Mobility.....	18
6. Mapping of TN Characteristics on SD-WAN Edge Node.....	21
6.1. SD-WAN Hybrid Use Case with SR-TE Integration.....	23
7. IANA Considerations.....	25
8. Security Considerations.....	25
9. Contributors.....	25
10. References.....	25
10.1. Normative References.....	25

10.2. Informative References.....	26
11. Acknowledgments.....	26
Authors' Addresses.....	28

1. Introduction

The [TN-AWARE-MOBILITY] draft defines the transport path characteristics in backhaul, midhaul, and fronthaul segments between the radio side network functions and user plane functions (UPF). It describes how various transport network underlay routing mechanisms apply to the framework laid out including RSVP-TE, SR, and also a data plane agnostic integrated routing and TE mechanism - Preferred Path Routing (PPR) to map the network slice properties into the IP/L2 transport network.

The current [TN-AWARE-MOBILITY] draft doesn't extend the transport network characteristics from the UPF through the Data Network. If the user service termination happens in the data network, the Transport Path Network characteristics through the Data Network would be lost.

This proposed Extension of Transport Aware Mobility in Data Network extends the mobility aware transport network characteristics from the UPF through the Data Network.

The UPF can be placed on the edge of the network where it can perform entry or exit point to the Data Network. It can connect to a Provider Edge node as well and bring all the mobile connections in a distributed way to the Data Network.

The UPF can as well connect to the SD-WAN edge node or L3 aggregator device and would try to bring all the 5G mobility connections for small, medium, and large enterprises. This would be a scenario for Enterprise 5G.

The current draft proposes mechanisms on how mobility aware transport network characteristics to be mapped into SR-TE paths or Un-secure, Secure, Secure SR-TE paths based in the Data Network on different use cases scenarios.

2. Conventions used in this document

BSID	- Binding SID
DC	- Data Center
DN	- Data Network (5G)
EMBB	- enhanced Mobile Broadband (5G)
gNB	- 5G NodeB
GTP-U	- GPRS Tunneling Protocol - Userplane (3GPP)
MIOT	- Massive IOT (5G)
PECP	- Path Computation Element (PCE) Communication Protocol
SD-WAN	- Software-Defined Wide Area Network
SID	- Segment Identifier
SLA	- Service Layer Agreement
SST	- Slice and Service Types (5G)
SR	- Segment Routing
SR-PCE	- SR Path Computation Element
UE	- User Equipment
UPF	- User Plane Function (5G)
URLLC	- Ultra reliable and low latency communications (5G)

3. Framework for Extension of Transport Network Aware Mobility

Architecture wise, the proposed Extension of Transport Aware Mobility in the Data Network solution focuses on the following areas:

- a) The Mobility packet transition in and out from the UPF to the C-PE Node maintaining the Transport Path Characteristics.
- b) On a PE node, based on the transport characteristics, use different methods of fetching SR-TE path segments from the SR-TE Controller and map the SR-TE segments with the mobility aware transport packets.
- c) On an SD-WAN CE Node, based on the transport characteristics, mapping of mobility aware transport packets to the secure and un-secure tunnel path.

Figure 1 captured under Section 4 provides the representation of a network on how UE could be connected to the UPF and C-PE nodes in the Data Network. The C-PE node represents a combined CE and PE node. In some cases, UPF would be connected to the pure PE or CE node.

4. Mobility Packet Transition to the Data Network

As the Transport Aware Mobility packets transition in and out from the UPF to the PE or C-PE (in SDWAN case) node, the Mobility Transport Characteristics need to be maintained in the Data Network. The current solution proposes a generic approach to how the mobility packet transition can happen in the Data Network maintaining the same transport characteristics.

There are two scenarios could happen here:

A) The UPF is not co-located with the C-PE in the same device. Based on the local policy the proposed new header format for the TN Aware Mobility Packets transitioning from the UPF to the C-PE device and vice-versa is proposed below:

. From the UPF to the C-PE Node:
Inner IP Hdr (UE Packet) + Transport Hdr (Carrying UDP Src Port) + Outer IP (C-PE Node Address)

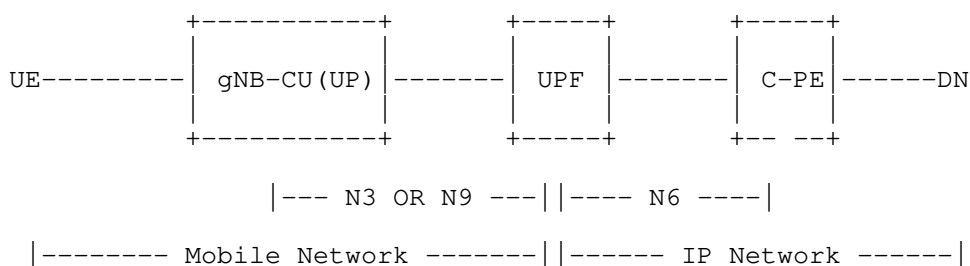
. From the C-PE to the UPF Node:
Outer IP (UPF Node Address) + Transport Hdr (Carrying UDP Src Port) + Inner IP Hdr (UE Packet)

B) The UPF is co-located with the C-PE in the same device. Based on the local policy the original UDP Source Port information can be passed to the local C-PE node and no new header is needed here.

The current draft proposes to create a new encapsulation header in scenario A. At the UPF node, the TN aware mobility UE packet carrying the original UDP header Source Port information along with the Inner IP packet to get encapsulated with the outer IP header of the outgoing C-PE node IP address.

In below Figure 1, both scenarios are captured. Scenario A captures the UPF is physically separated from the C-PE node over an IP network. Scenario B captures the edge networking deployment. In that case, the virtual UPF could be co-located with the physical C-PE node in the same device.

Scenario A:



Scenario B:

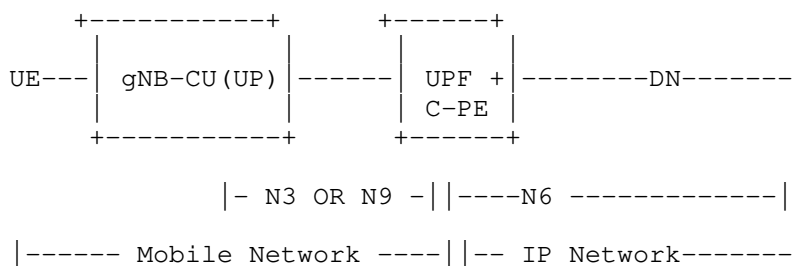
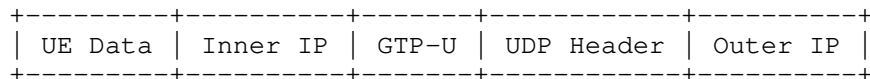


Figure 1: Mobile and IP Data Network for UE

The Figure 2 captures the TN Aware Mobility packet format under the scenario A.

1. UE Packet format in the Mobile Network to the UPF:



2. UE Packet format in the IP Network to the Ingress C-PE Node:

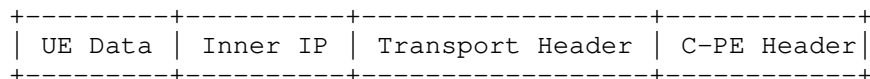


Figure 2: UE Packet Transition from Mobile to IP Network

The source port in the original UDP header indicates the TN Aware Mobility SST type.

5. Transport Network Characteristics Mapping to SR-TE Paths

With the 5G Mobile Networking, the UPF would be terminating the mobile connection from the UE. In some Edge Networking scenarios, the virtual UPF would be co-located with the C-PE or it would be connected to the C-PE node over IP Network.

The 5G UE traffic coming to the UPF might be carrying Transport Network Characteristics. In that scenario, there would be a need to maintain the Transport Path Characteristic through the core of the network so that end to end SLA can be maintained for the UE traffic.

In scenarios, where ingress PE acting as SR-TE node, the mapping of Transport Network Aware Mobility {5G UDP Src Port Range} to {BGP SR-TE Policy, BSID} to be done at the ingress PE. Once this mapping is done, the mobility Transport Path Characteristics can be maintained in the data network.

On a PE node, based on the transport characteristics, the current solution proposes different methods of applying SR-TE path segments:

Scenario 1: In this scenario, the assumption is that the Ingress PE node is connected to the BGP SR-TE Controller through the BGP SR-TE Policy SAFI Session. This solution defines a mechanism to

map the BGP SR-TE Underlay Path Segments based on the Mobility Transport Characteristics.

- . This mechanism would require a new BGP Sub-TLV as part of the existing SR Policy SAFI NLRI to download SR-TE Policies corresponds to the mobility Transport Path characteristics. If the TN aware mobility packet UDP Source Port value falls within the UDP Src Port range value of this Sub-TLV, then the pre-downloaded SR-TE Policy MUST be applied on the mobility traffic to map to the correct network slice in the Data Network. Once the Policy is fetched it would be cached by the PE node for operating in line with the subsequent mobility TN aware packets.

Scenario 2: In this scenario, the assumption is that the Ingress PE node is connected to the SR-PCE (Path Compute Element) Controller through the PCEP Session. This draft defines a mechanism to map the SR-TE Underlay Segments based on the Mobility Transport Path Characteristics.

- . Currently, this mechanism does not require new encoding in the PCEP based communication, though it needs local Configuration in the PE node to request the SR-TE Paths from the PCEP based Controller based on on-demand TN aware mobility traffic metric types.

Scenario 3: In this scenario, the assumption is that the Ingress PE node is connected to the SR-TE Controller over Restconf/Netconf or gRPC session. The existing mechanism would be used to download the SR-TE Underlay Path Segments to the PE node based on the Mobility Transport Path Characteristics.

- . The Yang Data Model or Protobuf definition is required to define a new Sub-TLV like Scenario 1. The SR-TE Controller would pre-download the SR-TE Policies with the new Sub-TLV in the Ingress PE using the existing session. Once the specific SR-TE Policy is fetched, it would be cached by the Ingress PE to apply for the mobility TN aware traffic in-line to maintain the network characteristics in the Data Network.

Scenario 4: In this scenario, the assumption is that the Ingress PE node is connected to the BGP SR FlowSpec Controller through the BGP FlowSpec Session. This draft defines a mechanism to map the

FlowSpec redirect to indirection-id community-based SR Traffic rules to the Mobility Transport Path Characteristics.

- . Currently, this mechanism does not require any new encoding in the BGP SR FlowSpec path redirect draft [FLOWSPEC-PATH-REDIRECT], though it needs local Configuration in the Ingress PE node that is acting as BGP FlowSpec Client to map the Mobility traffic based on the SR FlowSpec traffic re-direction rules.

5.1. Extend TN Aware Mobility for BGP SR-TE Policy

- 1) To integrate Transport Network Aware Mobility with BGP SR-TE Policy at the Ingress PE UPF, the Class-map needs to be defined to classify the incoming mobility traffic with different Transport Path Characteristic.
- 2) The Ingress PE UPF is assumed to have a BGP SR-TE Policy SAFI connection with the BGP SR-TE Controller. The Mobility traffic destination would resolve in the BGP Peer Next Hop for which SR-TE Policy to be applied to maintain the same network characteristics beyond the mobility domain.
- 3) A new 5G Metadata Sub-TLV has been defined for existing SR-Policy SAFI with the UDP Source Port Range to identify the SR-TE path based on the Transport Path characteristics.
- 4) The BGP SR-TE Controller would be programmed with {5G UDP Src Port Range}. That would create internal mapping Table for {5G UDP Src Port Range} < -- > {BGP SR-TE Policy, BSID}.
- 5) The BGP SR-TE Controller would download the SR-TE Policy in the Ingress PE through the existing BGP SR-Policy SAFI session, and that the BGP update would include an additional 5G Metadata Sub-TLV. The UDP Src Port range in the 5G Metadata Sub-TLV MUST fall within the UDP Source Port range for the SSTs defined by the [TN-AWARE-MOBILITY] draft. If the UDP Src Port range falls outside the range defined by the [TN-AWARE-MOBILITY] draft, then the SR-TE Policy SHOULD be ignored by the Ingress PE.
- 6) The SR-TE Policy-based traffic steering would be applied in the Ingress PE and it would maintain the local mapping for the reverse Mobility traffic to the UE.

The following class-map definition needs to be applied in the headend PE for the incoming Transport Network aware mobility traffic path:

```
Class-map type traffic match MIOT
```

```
    Match UDP Src Port Range Xx - Xy
```

```
Class-map type traffic match URLLC
```

```
    Match UDP Src Port Range Yx - Yy
```

```
Class-map type traffic match EMBB
```

```
    Match UDP Src Port Range Zx - Zy
```

The class-map would help to identify the incoming mobility traffic characteristics. Based on these characteristics the headend PE would be able to map the Transport Network aware mobility traffic to the appropriate BGP SR-TE Policy path over the Data Network to reach the UE's destination.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the Ingress PE having BGP SR-Policy SAFI connection with the BGP SR-TE Controller:

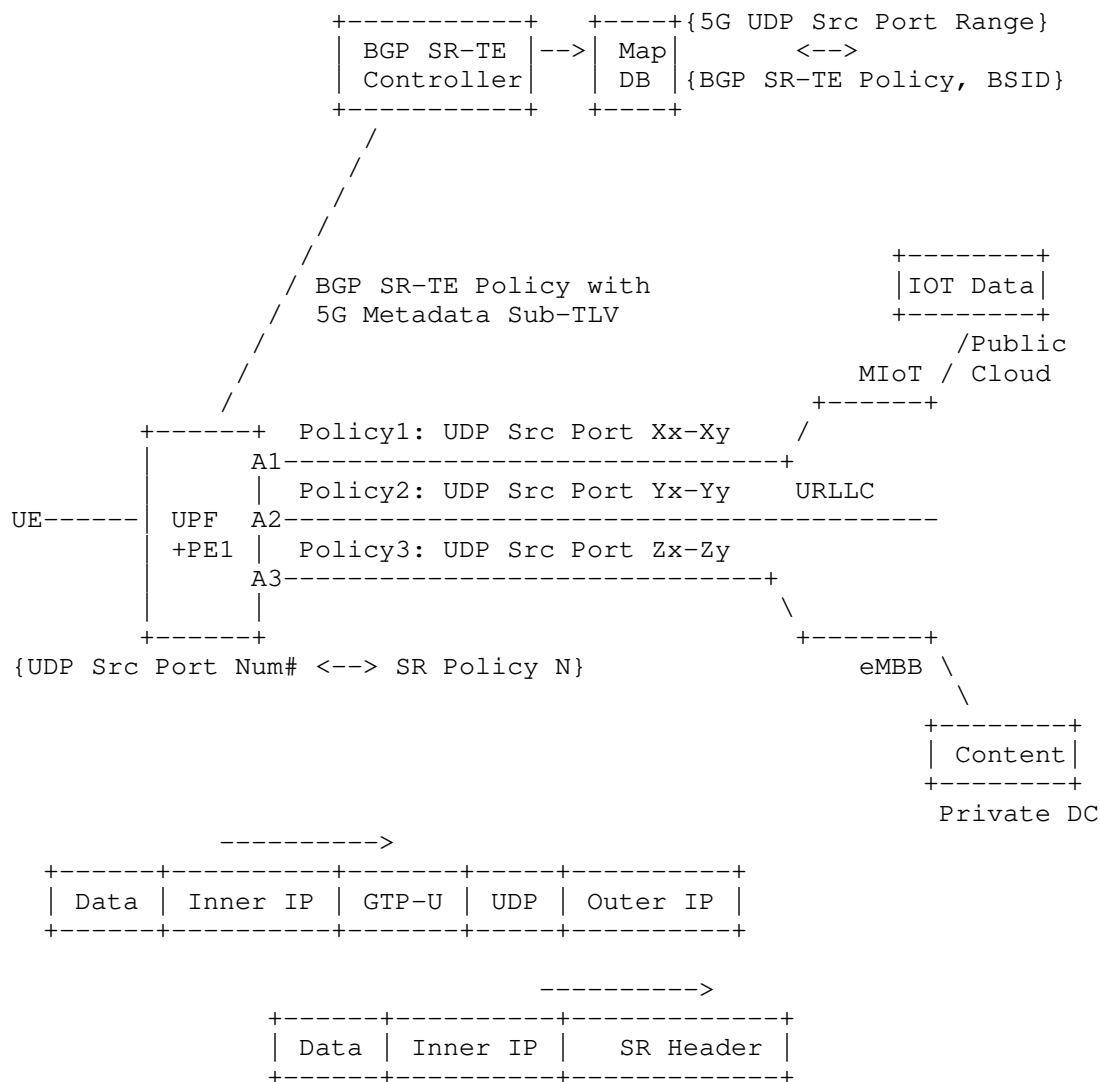


Figure 3: TN Aware Mobility Traffic Mapping to BGP SR-TE Policy Path

Note that, in the above figure SR Header is shown as an illustrative purpose and the actual outgoing packet format is based on the BGP SR-TE mechanism (SR-MPLS or SRv6) on the Ingress PE. That could be SR-MPLS or SRv6 Header. Though if the BSID is not present with the BGP SR-TE Policy, the local Ingress PE would map the incoming traffic to the best effort policy map path in the underlay.

To support the Transport Network Mobility Traffic Mapping to BGP SR-TE Policy Path in the headend PE, a new 5G Metadata Sub-TLV needs to be supported. The proposed BGP SR Policy Encoding from the BGP SR-TE Policy Controller to the headend PE node is defined below:

SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>

Attributes:

Tunnel Encap Attr (23)

Tunnel Type: SR Policy

Existing Policy Sub-TLV

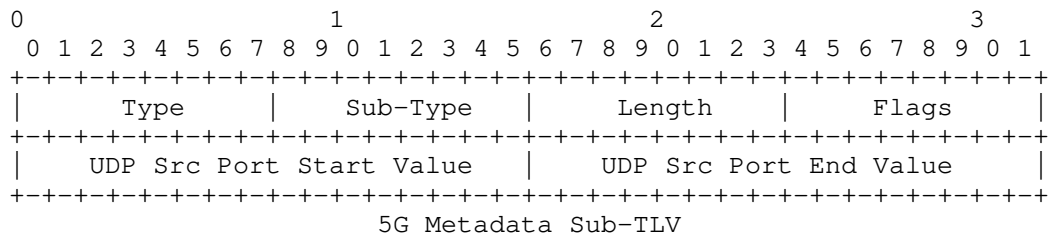
5G Metadata Sub-TLV

The draft [BGP-SR-TE-POLICY] defines BGP SR-TE Policy encodings. There is no change in the existing encoding that is being used from the BGP SR-TE Controller to the headend PE node. The current solution proposes the new 5G Metadata Sub-TLV for BGP SR-TE Controller to download the SR Policies to the headend PE and to apply the SR-TE Policy-driven path for the Transport Network aware mobility traffic.

The incoming TN aware mobility traffic with UDP Src port and BGP NH to the traffic destination would be used as a key to find the BGP SR-TE Policy. If the BGP Next Hop of the traffic matches with the SR Policy SAFI NLRI Endpoint, and UDP Src Port value falls within the UDP Src Port range defined by the 5G Metadata Sub-TLV, the SR Policy would be applied to the mobility traffic to maintain the traffic characteristics in the data network. The BGP SR-TE Controller would be pre-provisioned with the 5G UDP SRC Port Range based on the [TN-AWARE-MOBILITY] draft, and their corresponding BGP SR-TE Policy.

The 5G Metadata sub-TLV is optional and it MUST NOT appear more than once in the SR-TE Policy.

The format of the new SR-TE 5G Metadata Sub-TLV is captured below:



where:

- o Type: To be defined by IANA.
- o Sub-Type: This field has one of the following values:
 - 0: Reserved.
 - 1: UDP Source Port Range.
 - 2 - 255: Reserved for future use.
- o Length: 6 octets.
- o Flags: 1 octet of flags. None are defined at this stage. Flags SHOULD be set to zero on transmission and MUST be ignored on receipt.
- o UDP Src Port Start Value: 2 octets value to define the starting of the value of the UDP Src Port range.
- o UDP Src Port End Value: 2 octets value to define the end value of the UDP Src Port range.

5.2. Extend TN Aware Mobility for SR-PCE Controller

- 1) To integrate Transport Network Aware Mobility with SR-TE ODN based PCE Controller at the Ingress PE UPF, the Class-map needs to be defined to classify the incoming mobility traffic with different Transport Path Characteristic.
- 2) The Ingress PE UPF is assumed to have PCEP based communication with the SR-PCE Controller.

- 3) The Ingress PE would define the Policy-map to map the Transport Path characteristics into SR-TE Color.
- 4) The Segment Routing TE Configuration for different Metric types will associate the SR-TE Colors with their corresponding TE metric type.
- 5) The existing SR-TE ODN based PCEP messages with TE metric type and value MUST be used to associate the SR-TE Path corresponding to the 5G UDP Src Port.
- 6) In this case, the mapping between {5G UDP Src Port} and {SR-TE Policy} would be maintained by the Ingress PE.
- 7) Once the TN aware mobility traffic destination resolves into a destination of BGP Peer Next Hop, the SR-TE ODN based traffic steering MUST be applied based on the UDP Src Port value of the incoming traffic.

The class-map definition to identify the incoming mobility traffic characteristics is already defined in Section 5.1. The same class-map definition applicable here as well.

The policy-map definition to associate SR-TE color with Transport Path characteristics is defined below:

Policy-map type Transport-Network-Aware-Mobility

```
Class type traffic MIOT
    Set color <MIOT-10>

Class type traffic URLLC
    Set color <URLLC-20>

Class type traffic EMBB
    Set color <EMBB-30>
```

The Segment Routing TE Configuration mechanism can associate the SR-TE Colors with their corresponding metric type. That exists today, and there is no change there. It is captured here to show how TN

aware mobility network characteristics get mapped to different TE metrics through this mechanism.

Segment-routing traffic-eng

On-demand color <MIOT-10> dynamic

Metric

Type <MIOT>

On-demand color <URLLC-20> dynamic

Metric

Type <URLLC>

On-demand color <EMBB-30> dynamic

Metric

Type <EMBB>

As a result, mobility Transport Network aware of different traffic characteristics like MIOT, URLLC, or EMBB get to assigned corresponding "te" metric types. To fetch the corresponding SR-TE dynamic path from the SR-PCE Controller based on the newly defined "te" metric types <MIOT>, <URLLC> or <EMBB> needs to be extended in the PCEP RFC [RFC5440].

The below figure tries to capture the overall topology, and how to map the mobility traffic in the headend PE having PCEP connection with the SR-PCE Controller:

- 3) A new 5G Metadata Yang data model and Protobuf to be defined for SR-Policy SAFI with UDP Source Port Range to identify the SR-TE path based on the Transport Path characteristics.
- 4) The SR-TE Controller would be programmed with {5G UDP Src Port Range}. That would create internal mapping Table for {5G UDP Src Port Range} < -- > {BGP SR-TE Policy, BSID}.
- 5) As the Headend PE sends the 5G metadata Yang data model or Protobuf, the Controller will find a matching SR-TE Policy based on the UDP Source Port.
- 6) The SR-TE Controller would download the SR-TE Policy in the Ingress PE through the existing Restconf or gRPC session, and that BGP update would include an additional 5G Metadata Sub-TLV. The UDP Src Port range in the 5G Metadata Sub-TLV MUST fall within the UDP Source Port range for the SSTs defined by the [TN-AWARE-MOBILITY] draft. If the UDP Src Port range falls outside the range defined by the [TN-AWARE-MOBILITY] draft, then the SR-TE Policy SHOULD be ignored by the Ingress PE.
- 7) The SR-TE Policy-based traffic steering would be applied in the Ingress PE UPF and it would maintain the local mapping for the reverse Mobility traffic to the UE.

The class-map definition to identify the incoming mobility traffic characteristics is already defined in Section 5.1. The same class-map definition works here as well.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the headend PE having BGP SR-Policy SAFI connection with the BGP SR-PCE Controller:

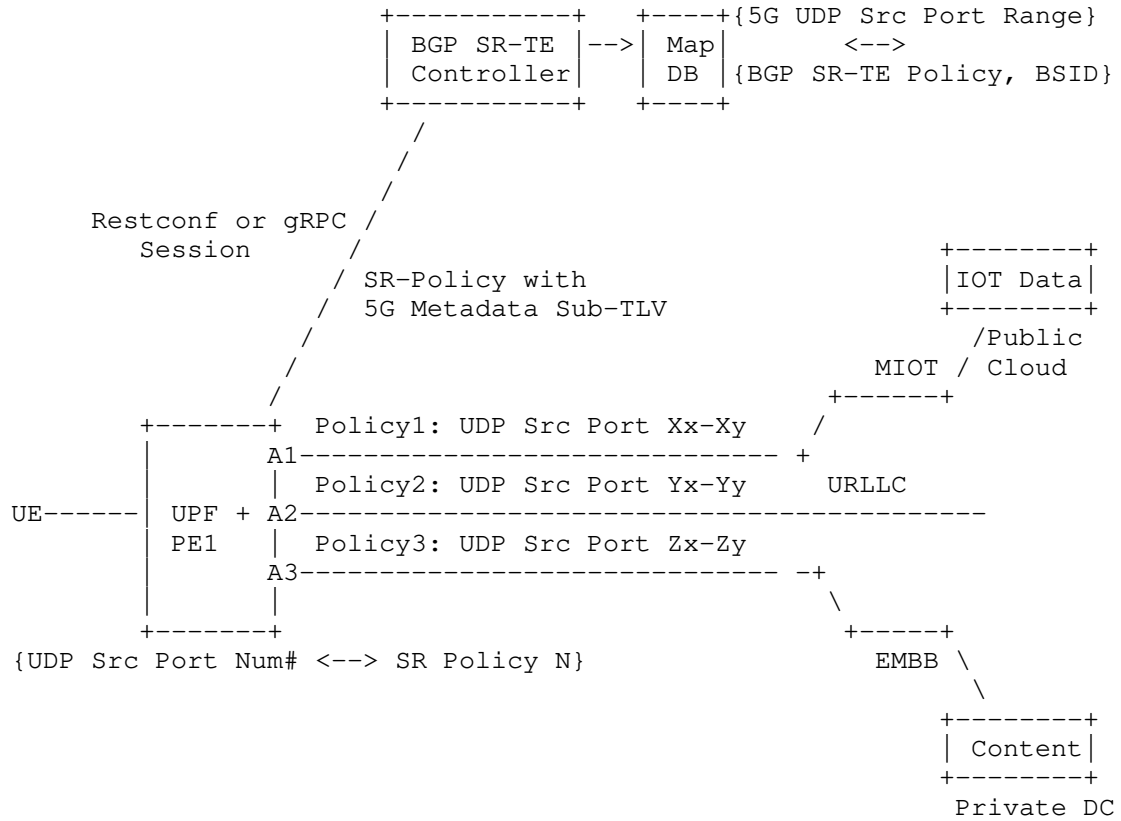


Figure 5: TN Aware Mobility Traffic Mapping to SR-TE Path

5.4. Extend BGP FlowSpec for TN Aware Mobility

- 1) To integrate Transport Network Aware Mobility with SR-TE Policy at the Ingress PE UPF, the Class-map needs to be defined to classify the incoming mobility traffic with different Transport Path Characteristic.
- 2) The Ingress PE UPF that is acting as BGP FlowSpec Client is assumed to have a BGP FlowSpec session with the FlowSpec Controller. The Mobility traffic destination would resolve in the BGP Peer Next Hop for which SR FlowSpec traffic re-direct policy to be applied to maintain the same network characteristics in the data network.

- 3) The current proposal tries to integrate FlowSpec Redirect to Indirection ID [FLOWSPEC-PATH-REDIRECT] based traffic rules with the TN aware mobility traffic based on the UDP Source Port range at the FlowSpec Client Router/Ingress PE.
- 4) Based on the BGP FlowSpec RFC 8955 the BGP FlowSpec NLRI can carry out the UDP Source Port range. The 5G SST specific UDP Source Port range values can be pushed over a BGP FlowSpec session between the FlowSpec Controller and the Ingress PE node.
- 5) There are no additional changes required on the BGP FlowSpec side other than provisioning 5G SST specific UDP Source Port range at the FlowSpec Controller along with the corresponding FlowSpec Redirect to indirection-id.
- 6) The BGP FlowSpec Controller would be programmed with {5G UDP Src Port Range} to map different SSTs defined in [TN-AWARE-MOBILITY] draft to map the corresponding FlowSpec Redirect to Indirection-id. That would create internal mapping Table for {5G UDP Src Port Range} < -- > {BGP FlowSpec Generalized Indirection-ID}.
- 7) The BGP FlowSpec NLRI carrying 5G UDP Source Port Range along with the corresponding Redirect to indirection-id Extended Community can be pushed to the Ingress PE node.
- 8) The Mobility traffic coming from the UPF to the Ingress PE in the Data Network carrying specific UDP Source Port from UE can be classified based on the local Policy and apply the BGP FlowSpec based re-direction rule based on the matching FlowSpec policy.

The class-map definition to identify the incoming mobility traffic characteristics is already defined in Section 5.1. The same class-map definition works here as well.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the Ingress PE acting as FlowSpec Client having BGP FlowSpec SAFI connection with the BGP FlowSpec Controller:

6. Mapping of TN Characteristics on SD-WAN Edge Node

On an SD-WAN CE Node, based on the mobility Transport Network characteristics, mapping of mobility aware transport packets to the secure and un-secure tunnel path needs to be achieved.

The [BGP-IPSEC-Discover] draft defines how SD-WAN Edge Node maps the overlay/client routes to the underlay secure tunnel routes.

The current proposal specifies a generic approach on how SD-WAN Edge Node maps the Mobility Transport Network aware traffic to the Secure Tunnels, or Un-Secure TE Paths, or Secure SR-TE Tunnel Paths.

The [SDWAN-BGP-USAGE] draft describes how BGP can be used as a Control Plane for the SD-WAN network and defines the use case for the Hybrid SD-WAN network.

In the case of a hybrid SD-WAN use case, UPF can run part of the SD-WAN edge node or it could be connected to it over an IP network. This would be a use case scenario for Enterprise 5G.

In that scenario, the Transport Path Characteristic for the 5G mobile traffic need to be mapped to Secure (IPSec Tunnel) or Un-secure path (could be MPLS based).

The existing [TN-AWARE-MOBILITY] draft is extended to support new Transport Path Characteristics "Security" for the mobile traffic where security is important for certain mobile traffic.

Based on the UDP Src Port characteristics coming from the mobile network, the SD-WAN edge node would be able to decide what traffic it needs to put in the secure tunnel vs. an un-secure tunnel where low latency more important than security.

The SD-WAN Edge Node can map the URLLC traffic without any security characteristics to the underlay MPLS path, whereas MIOT, and EMBB traffic with security characteristics gets mapped to the underlay Secure IPsec Tunnel path. The mapping between SD-WAN overlay and underlay routes are described in the [BGP-IPSEC-Discovery] draft.

This solution extends it for Transport Network aware mobility traffic. The SD-WAN Edge Node here identifies the incoming mobility traffic characteristics using the class-map definition, and that is already defined under Section 5.1. Based on the incoming traffic characteristics, the Edge Node will be able to map the mobility overlay traffic to the respective SD-WAN underlay tunnel.

6.1. SD-WAN Hybrid Use Case with SR-TE Integration

- 1) In the case of SD-WAN hybrid use cases, UPF can run part of the SD-WAN edge node, or it could be connected to it over an IP network. This would be a use case scenario for Enterprise 5G.
- 2) The SD-WAN edge node can act as an SR-TE Headend PE in some use case scenarios that are described in [SDWAN-BGP-USAGE] draft.
- 3) In that case, the Headend PE could be connected with SR-TE Policy Controller over the BGP SR-Policy SAFI session, or SR-PCE Controller over the PCEP session, or SR-TE Controller over Netconf/ Restconf, or GRPC session, or even SR FlowSpec Controller over BGP FlowSpec session.
- 4) The SD-WAN edge node can map the "Un-secure" mobility traffic to the SR-TE path the same way as described under PE acting as ingress SR-TE headend.
- 5) Though the mapping for "Secure" mobility traffic to the SR-TE path would be slightly different than "Un-secure" mobility traffic.
- 6) The mobility 5G UE client traffic with the Transport Path Characteristics "Security" would be encapsulated with Tunnel mode IPsec header between the two SD-WAN SAFI underlay endpoints (belong to the same BGP AS domain). This encapsulated secure traffic will become the new overlay for the SR-TE traffic.
- 7) The rest of the mechanism for the secure mobility traffic with SR-TE traffic forwarding is the same as un-secure SR-TE based traffic forwarding.

In Figure 8, the traffic coming from the mobility side with Transport Network characteristics gets mapped to the underlay un-secure or secure SR-TE path to maintain the traffic network characteristics in the Data Network.

The SD-WAN Edge Node can map the URLLC traffic without any security characteristics to the underlay SR-TE path without any IPSec encapsulation. Whereas MIOT and EMBB traffic with the security characteristics can be mapped to the underlay Secure IPSec Tunnel path with the SR-TE encapsulation to the SD-WAN endpoints.

7. IANA Considerations

The newly defined 5G Metadata Sub-TLV would need an IANA code point allocation for the Type field. A request for any IANA code point allocation would be submitted.

8. Security Considerations

This document does not introduce any new security issues.

9. Contributors

The following people have contributed to this document.

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[RFC5440] JP. Vasseur, Ed., JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", March 2009

[TN-AWARE-MOBILITY] U. Chunduri, et al, "Transport Network aware Mobility for 5G", draft-ietf-dmm-tn-aware-mobility-03, March 2022

[BGP-SR-TE-POLICY] S. Previdi, et al, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-16, March 2022

[SDWAN-BGP-USAGE] L. Dunber, et al, "BGP Usage for SDWAN Overlay Networks", draft-ietf-bess-bgp-sdwan-usage-04, October 2021

[BGP-IPSEC-Discover] L. Dunber, et al, "BGP UPDATE for SDWAN Edge Discovery", draft-ietf-idr-sdwan-edge-discovery-01, March 2022

[RFC9012] K. Patel, et al, "The BGP Tunnel Encapsulation Attribute", April 2021.

11. Acknowledgments

TBD.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Kausik Majumdar
CommScope

Email: kausik.majumdar@commscope.com

Uma Chunduri
Intel Corporation

Email: umac.ietf@gmail.com

Linda Dunbar
Futurewei

Email: linda.dunbar@futurewei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 27 October 2022

F. L. Templin, Ed.
Boeing Research & Technology
25 April 2022

Automatic Extended Route Optimization (AERO)
draft-templin-6man-aero-46

Abstract

This document specifies an Automatic Extended Route Optimization (AERO) service for IP internetworking over Overlay Multilink Network (OMNI) interfaces. AERO/OMNI use an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (IPv6 ND) protocol. Prefix delegation/registration services are employed for network admission and to manage the IP forwarding and routing systems. Secure multilink operation, mobility management, multicast, traffic path selection and route optimization are naturally supported through dynamic neighbor cache updates. AERO is a widely-applicable mobile internetworking service especially well-suited to aviation services, intelligent transportation systems, mobile end user devices and many other applications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Terminology	6
3. Automatic Extended Route Optimization (AERO)	15
3.1. AERO Node Types	15
3.2. The AERO Service over OMNI Links	16
3.2.1. AERO/OMNI Reference Model	16
3.2.2. Addressing and Node Identification	20
3.2.3. AERO Routing System	21
3.2.4. Segment Routing Topologies (SRTs)	23
3.2.5. Segment Routing For OMNI Link Selection	23
3.3. OMNI Interface Characteristics	24
3.4. OMNI Interface Initialization	26
3.4.1. AERO Proxy/Server and Relay Behavior	27
3.4.2. AERO Client Behavior	27
3.4.3. AERO Host Behavior	28
3.4.4. AERO Gateway Behavior	28
3.5. OMNI Interface Neighbor Cache Maintenance	28
3.5.1. OMNI ND Messages	30
3.5.2. OMNI Neighbor Advertisement Message Flags	32
3.5.3. OMNI Neighbor Window Synchronization	33
3.6. OMNI Interface Encapsulation and Fragmentation	33
3.7. OMNI Interface Decapsulation	36
3.8. OMNI Interface Data Origin Authentication	36
3.9. OMNI Interface MTU	37
3.10. OMNI Interface Forwarding Algorithm	37
3.10.1. Host Forwarding Algorithm	39
3.10.2. Client Forwarding Algorithm	39
3.10.3. Proxy/Server and Relay Forwarding Algorithm	40
3.10.4. Gateway Forwarding Algorithm	43
3.11. OMNI Interface Error Handling	44
3.12. AERO Mobility Service Coordination	47
3.12.1. AERO Service Model	47
3.12.2. AERO Host and Client Behavior	48
3.12.3. AERO Proxy/Server Behavior	49
3.13. AERO Route Optimization	56
3.13.1. Multilink Address Resolution	57
3.13.2. Multilink Route Optimization	61

3.13.3.	Rapid Commit Route Optimization	73
3.13.4.	Client/Gateway Route Optimization	73
3.13.5.	Client/Client Route Optimization	75
3.13.6.	Client-to-Client OMNI Link Extension	77
3.13.7.	Intra-ANET/ENET Route Optimization for AERO Peers .	78
3.14.	Neighbor Unreachability Detection (NUD)	78
3.15.	Mobility Management and Quality of Service (QoS)	80
3.15.1.	Mobility Update Messaging	80
3.15.2.	Announcing Link-Layer Information Changes	81
3.15.3.	Bringing New Links Into Service	82
3.15.4.	Deactivating Existing Links	82
3.15.5.	Moving Between Proxy/Servers	82
3.16.	Multicast	83
3.16.1.	Source-Specific Multicast (SSM)	84
3.16.2.	Any-Source Multicast (ASM)	85
3.16.3.	Bi-Directional PIM (BIDIR-PIM)	86
3.17.	Operation over Multiple OMNI Links	86
3.18.	DNS Considerations	87
3.19.	Transition/Coexistence Considerations	87
3.20.	Proxy/Server-Gateway Bidirectional Forwarding Detection	88
3.21.	Time-Varying MNPs	88
4.	Implementation Status	88
5.	IANA Considerations	89
6.	Security Considerations	89
7.	Acknowledgements	91
8.	References	93
8.1.	Normative References	93
8.2.	Informative References	95
Appendix A.	Non-Normative Considerations	102
A.1.	Implementation Strategies for Route Optimization	103
A.2.	Implicit Mobility Management	103
A.3.	Direct Underlying Interfaces	104
A.4.	AERO Critical Infrastructure Considerations	104
A.5.	AERO Server Failure Implications	105
A.6.	AERO Client / Server Architecture	105
Appendix B.	Change Log	107
Author's Address	108

1. Introduction

Automatic Extended Route Optimization (AERO) fulfills the requirements of Distributed Mobility Management (DMM) [RFC7333] and route optimization [RFC5522] for aeronautical networking and other network mobility use cases including intelligent transportation systems and enterprise mobile device users. AERO is a secure internetworking and mobility management service that employs the Overlay Multilink Network Interface (OMNI) [I-D.templin-6man-omni]

Non-Broadcast, Multiple Access (NBMA) virtual link model. The OMNI link is a virtual overlay configured over one or more concatenated underlay Internetworks, and nodes on the link can exchange original IP packets as single-hop neighbors. The OMNI Adaptation Layer (OAL) supports multilink operation for increased reliability and path optimization while providing fragmentation and reassembly services to support improved performance and Maximum Transmission Unit (MTU) diversity. This specification provides a mobility service architecture companion to the OMNI specification.

The AERO service connects Hosts and Clients over Proxy/Servers and Relays that are seen as OMNI link neighbors; AERO further includes Gateways that interconnect diverse Internetworks as OMNI link segments through OAL forwarding at a layer below IP. Each node's OMNI interface uses an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (IPv6 ND) protocol [RFC4861]. A Client's OMNI interface can be configured over multiple underlay interfaces, and therefore appears as a single interface with multiple link-layer addresses. Each link-layer address is subject to change due to mobility and/or multilink fluctuations, and link-layer address changes are signaled by ND messaging the same as for any IPv6 link.

AERO provides a secure cloud-based service where mobile node Clients may use Proxy/Servers acting as proxys and/or designated routers while fixed nodes may use any Relay on the link for efficient communications. Fixed nodes forward original IP packets destined to other AERO nodes via the nearest Relay, which forwards them through the cloud. Mobile node Clients discover shortest paths to OMNI link neighbors through AERO route optimization. Both unicast and multicast communications are supported, and Clients may efficiently move between locations while maintaining continuous communications with correspondents and without changing their IP Address.

AERO Gateways peer with Proxy/Servers in a secured private BGP overlay routing instance to establish a Segment Routing Topology (SRT) spanning tree over the underlay Internetworks of one or more disjoint administrative domains concatenated as a single unified OMNI link. Each OMNI link instance is characterized by the set of Mobility Service Prefixes (MSPs) common to all mobile nodes. Relays provide an optimal route from (fixed) correspondent nodes on underlay Internetworks to (mobile or fixed) nodes on the OMNI link. To the underlay Internetwork, the Relay is the source of a route to the MSP; hence uplink traffic to mobile nodes is naturally routed to the nearest Relay.

AERO can be used with OMNI links that span private-use Internetworks and/or public Internetworks such as the global Internet. In both cases, Clients may be located behind Network Address Translators (NATs) on the path to their associated Proxy/Servers. A means for robust traversal of NATs while avoiding "triangle routing" and critical infrastructure traffic concentration is therefore provided.

AERO assumes the use of PIM Sparse Mode in support of multicast communication. In support of Source Specific Multicast (SSM) when a Mobile Node is the source, AERO route optimization ensures that a shortest-path multicast tree is established with provisions for mobility and multilink operation. In all other multicast scenarios there are no AERO dependencies.

AERO provides a secure aeronautical internetworking service for both manned and unmanned aircraft, where the aircraft is treated as a mobile node that can connect an Internet of Things (IoT). AERO is also applicable to a wide variety of other use cases. For example, it can be used to coordinate the links of mobile nodes (e.g., cellphones, tablets, laptop computers, etc.) that connect into a home enterprise network via public access networks with VPN or non-VPN services enabled according to the appropriate security model. AERO can also be used to facilitate terrestrial vehicular and urban air mobility (as well as pedestrian communication services) for future intelligent transportation systems [I-D.ietf-ipwave-vehicular-networking][I-D.templin-ipwave-uam-its]. Other applicable use cases are also in scope.

Along with OMNI, AERO provides secured optimal routing support for the "6M's" of modern Internetworking, including:

1. Multilink - a mobile node's ability to coordinate multiple diverse underlay data links as a single logical unit (i.e., the OMNI interface) to achieve the required communications performance and reliability objectives.
2. Multinet - the ability to span the OMNI link over a segment routing topology with multiple diverse administrative domain network segments while maintaining seamless end-to-end communications between mobile Clients and correspondents such as air traffic controllers, fleet administrators, other mobile Clients, etc.
3. Mobility - a mobile node's ability to change network points of attachment (e.g., moving between wireless base stations) which may result in an underlay interface address change, but without disruptions to ongoing communication sessions with peers over the OMNI link.

4. Multicast - the ability to send a single network transmission that reaches multiple nodes belonging to the same interest group, but without disturbing other nodes not subscribed to the interest group.
5. Multihop - a mobile node vehicle-to-vehicle relaying capability useful when multiple forwarding hops between vehicles may be necessary to "reach back" to an infrastructure access point connection to the OMNI link.
6. MTU assurance - the ability to deliver packets of various robust sizes between peers without loss due to a link size restriction, and to dynamically adjust packets sizes to achieve the optimal performance for each independent traffic flow.

The following numbered sections present the AERO specification. The appendices at the end of the document are non-normative.

2. Terminology

The terminology in the normative references applies; especially, the terminology in the OMNI specification [I-D.templin-6man-omni] is used extensively throughout. The following terms are defined within the scope of this document:

IPv6 Neighbor Discovery (IPv6 ND)

a control message service for coordinating neighbor relationships between nodes connected to a common link. AERO uses the IPv6 ND messaging service specified in [RFC4861] in conjunction with the OMNI extensions specified in [I-D.templin-6man-omni].

IPv6 Prefix Delegation

a networking service for delegating IPv6 prefixes to nodes on the link. The nominal service is DHCPv6 [RFC8415], however alternate services (e.g., based on IPv6 ND messaging) are also in scope. A minimal form of prefix delegation known as "prefix registration" can be used if the Client knows its prefix in advance and can represent it in the source address of an IPv6 ND message.

L2

The Data Link layer in the OSI network model. Also known as "layer-2", "link-layer", "sub-IP layer", etc.

L3

The Network layer in the OSI network model. Also known as "layer-3", "IP layer", etc.

Adaptation layer

A mid-layer that adapts L3 to a diverse collection of L2 underlay interfaces and their encapsulations. (No layer number is assigned, since numbering was an artifact of the legacy reference model that need not carry forward in the modern architecture.) The adaptation layer sees the upper layer as "L3" and sees all lower layer encapsulations as "L2 encapsulations", which may include UDP, IP and true link-layer (e.g., Ethernet, etc.) headers.

Access Network (ANET)

a connected network region (e.g., an aviation radio access network, satellite service provider network, cellular operator network, WiFi network, etc.) that joins Clients to the Mobility Service. Physical and/or data link level security is assumed, and sometimes referred to as "protected spectrum". Private enterprise networks and ground domain aviation service networks may provide multiple secured IP hops between the Client's point of connection and the nearest Proxy/Server.

Internetwork (INET)

a connected network region with a coherent IP addressing plan that provides transit forwarding services between ANETs and OMNI nodes that coordinate with the Mobility Service over unprotected media. No physical and/or data link level security is assumed, therefore security must be applied by upper layers. The global public Internet itself is an example.

End-user Network (ENET)

a simple or complex "downstream" network that travels with the Client as a single logical unit. The ENET could be as simple as a single link connecting a single Host, or as complex as a large network with many links, routers, bridges and Hosts. The ENET could also provide an "upstream" link in a recursively-descending chain of additional Clients and ENETs. In this way, an ENET of an upstream Client is seen as the ANET of a downstream Client.

{A,I,E}NET interface

a node's attachment to a link in an {A,I,E}NET.

underlay network/interface

an ANET/INET/ENET network/interface over which an OMNI interface is configured. The OMNI interface is seen as a L3 interface by the IP layer, and the OMNI adaptation layer sees the underlay interface as an L2 interface. The underlay interface either connects directly to the physical communications media or coordinates with another node where the physical media is hosted.

OMNI link

the same as defined in [I-D.templin-6man-omni]. The OMNI link employs IPv6 encapsulation [RFC2473] to traverse intermediate nodes in a spanning tree over underlay network segments the same as a bridged campus LAN. AERO nodes on the OMNI link appear as single-hop neighbors at the network layer even though they may be separated by many underlay network hops; AERO nodes can employ Segment Routing [RFC8402] to navigate between different OMNI links, and/or to cause packets to visit selected waypoints within the same OMNI link.

OMNI Adaptation Layer (OAL)

an OMNI interface sublayer service that encapsulates original IP packets admitted into the interface in an IPv6 header and/or subjects them to fragmentation and reassembly. The OAL is also responsible for generating MTU-related control messages as necessary, and for providing addressing context for spanning multiple segments of an L2-extended OMNI link.

OMNI Interface

a node's attachment to an OMNI link (i.e., the same as defined in [I-D.templin-6man-omni]). Since OMNI interface addresses are managed for uniqueness, OMNI interfaces do not require Duplicate Address Detection (DAD) and therefore set the administrative variable 'DupAddrDetectTransmits' to zero [RFC4862].

(network) partition

frequently, underlay networks such as large corporate enterprise networks are sub-divided internally into separate isolated partitions (a technique also known as "network segmentation"). Each partition is fully connected internally but disconnected from other partitions, and there is no requirement that separate partitions maintain consistent Internet Protocol and/or addressing plans. (Each partition is seen as a separate OMNI link segment as discussed throughout this document.)

L2 encapsulation

the OAL encapsulation of a packet in an outer header or headers that can be routed within the scope of the local {A,I,E}NET partition. Common L2 encapsulation combinations include UDP/IP/Ethernet, etc.

L2 address(es)

the addresses that appear in the OAL L2 encapsulations for an underlay interface.

INADDR

the UDP/IP addresses that appear in an L2 address.

original IP packet

a whole IP packet or fragment admitted into the OMNI interface by the network layer prior to OAL encapsulation and fragmentation, or an IP packet delivered to the network layer by the OMNI interface following OAL decapsulation and reassembly.

OAL packet

an original IP packet encapsulated in an OAL IPv6 header before OAL fragmentation, or following OAL reassembly.

OAL fragment

a portion of an OAL packet following fragmentation but prior to L2 encapsulation, or following L2 decapsulation but prior to OAL reassembly.

(OAL) atomic fragment

an OAL packet that can be forwarded without fragmentation, but still includes a Fragment Header with a valid Identification value and with Fragment Offset and More Fragments both set to 0.

(OAL) carrier packet

an encapsulated OAL fragment following L2 encapsulation or prior to L2 decapsulation. OAL sources and destinations exchange carrier packets over underlay interfaces, and may be separated by one or more OAL intermediate nodes. OAL intermediate nodes re-encapsulate carrier packets during forwarding by removing the L2 headers of the previous hop underlay network and replacing them with new L2 headers for the next hop underlay network.

OAL source

an OMNI interface acts as an OAL source when it encapsulates original IP packets to form OAL packets, then performs OAL fragmentation and L2 encapsulation to create carrier packets.

OAL destination

an OMNI interface acts as an OAL destination when it decapsulates carrier packets, then performs OAL reassembly and decapsulation to derive the original IP packet.

OAL intermediate node

an OMNI interface acts as an OAL intermediate node when it removes the L2 headers of carrier packets received from a previous hop, then re-encapsulates the carrier packets in new L2 headers and forwards them to the next hop. OAL intermediate nodes decrement the Hop Limit of the OAL IPv6 header during re-encapsulation, and discard the packet if the Hop Limit reaches 0. OAL intermediate nodes do not decrement the Hop Limit/TTL of the original IP packet.

Mobility Service Prefix (MSP)

an aggregated IP Global Unicast Address (GUA) prefix (e.g., 2001:db8::/32, 192.0.2.0/24, etc.) assigned to the OMNI link and from which more-specific Mobile Network Prefixes (MNPs) are delegated. OMNI link administrators typically obtain MSPs from an Internet address registry, however private-use prefixes can alternatively be used subject to certain limitations (see: [I-D.templin-6man-omni]). OMNI links that connect to the global Internet advertise their MSPs to their interdomain routing peers.

Mobile Network Prefix (MNP)

a longer IP prefix delegated from an MSP (e.g., 2001:db8:1000:2000::/56, 192.0.2.8/30, etc.) and delegated to an AERO Client or Relay.

Interface Identifier (IID)

the least significant 64 bits of an IPv6 address, as specified in the IPv6 addressing architecture [RFC4291].

Link Local Address (LLA)

an IPv6 address beginning with fe80::/64 per the IPv6 addressing architecture [RFC4291] and with either a 64-bit MNP (LLA-MNP) or a 56-bit random value (LLA-RND) encoded in the IID as specified in [I-D.templin-6man-omni].

Unique Local Address (ULA)

an IPv6 address beginning with fd00::/8 followed by a 40-bit Global ID followed by a 16-bit Subnet ID per [RFC4193] and with either a 64-bit MNP (ULA-MNP) or a 56-bit random value (ULA-RND) encoded in the IID as specified in [I-D.templin-6man-omni]. (Note that [RFC4193] specifies a second form of ULAs based on the prefix fc00::/8, which are referred to as "ULA-C" throughout this document to distinguish them from the ULAs defined here.)

Temporary Local Address (TLA)

a ULA beginning with fd00::/16 followed by a 48-bit randomly-initialized value followed by an MNP-based (TLA-MNP) or random (TLA-RND) IID as specified in [I-D.templin-6man-omni]. Clients use TLAs to bootstrap autoconfiguration in the presence of OMNI link infrastructure or for sustained communications in the absence of infrastructure. (Note that in some environments Clients can instead use a (Hierarchical) Host Identity Tag ((H)HIT) instead of a TLA - see: [I-D.templin-6man-omni].)

eXtended Local Address (XLA)

a TLA beginning with fd00::/64 followed by an MNP-based (XLA-MNP) or random (XLA-RND) IID as specified in [I-D.templin-6man-omni]. An XLA is simply a TLA with an all-0 48-bit value following

fd00::/16, and can be used to supply a "wildcard match" for IPv6 ND cache entries, a routing table entry for the OMNI link routing system, etc. (Note that XLAs can also be statelessly formed from LLAs (and vice-versa) simply by inverting prefix bits 7 and 8.)

AERO node

a node that is connected to an OMNI link and participates in the AERO internetworking and mobility service.

AERO Host ("Host")

an AERO node that configures an OMNI interface over an ENET underlying interface serviced by an upstream Client. The Host does not assign an LLA or ULA to the OMNI interface, but instead assigns the address taken from the ENET underlying interface. (As an implementation matter, the Host may instead configure the "OMNI interface" as a virtual sublayer of the underlay interface itself.) When an AERO host forwards an original IP packet to another AERO node on the same ENET, it uses simple IP-in-IP encapsulation without including an OAL encapsulation header. The Host is therefore an OMNI link termination endpoint.

AERO Client ("Client")

an AERO node that configures an OMNI interface over one or more underlay interfaces and requests MNP delegation/registration service from AERO Proxy/Servers. The Client assigns an XLA-MNP (as well as Proxy/Server-specific ULA-MNPs) to the OMNI interface for use in IPv6 ND exchanges with other AERO nodes and forwards original IP packets to correspondents according to OMNI interface neighbor cache state. The Client coordinates with Proxy/Servers and/or other Clients over upstream ANET/INET interfaces and may also provide Proxy/Server services for Hosts and/or other Clients over downstream ENET interfaces.

AERO Proxy/Server ("Proxy/Server")

a node that provides a proxying service between AERO Clients and external peers on its Client-facing ANET interfaces (i.e., in the same fashion as for an enterprise network proxy) as well as designated router services for coordination with correspondents on its INET-facing interfaces. (Proxy/Servers in the open INET instead configure only a single INET interface and no ANET interfaces.) The Proxy/Server configures an OMNI interface and assigns a ULA-RND to support the operation of IPv6 ND services, while advertising any associated MNPs for which it is acting as a hub via BGP peerings with AERO Gateways.

AERO Relay ("Relay")

a Proxy/Server that provides forwarding services between nodes reached via the OMNI link and correspondents on other links/

networks. AERO Relays configure an OMNI interface, assign a ULA-RND and maintain BGP peerings with Gateways the same as Proxy/Servers and run a dynamic routing protocol to discover any non-MNP IP GUA routes in service on other links/networks. The Relay advertises the MSP(s) to its other links/networks, and redistributes routes discovered on other links/networks into the OMNI link BGP routing system the same as for Proxy/Servers. (Relays that connect to major Internetworks such as the global IPv6 or IPv4 Internet can also be configured to advertise "default" routes into the OMNI link BGP routing system.)

AERO Gateway ("Gateway")

a BGP hub autonomous system node that also provides OAL forwarding services for nodes on an OMNI link. Gateways forward carrier packets between OMNI link segments as OAL intermediate nodes while decrementing the OAL IPv6 header Hop Limit but without decrementing the network layer IP TTL/Hop Limit. Gateways peer with Proxy/Servers and other Gateways to form an IPv6-based OAL spanning tree over all OMNI link segments and to discover the set of all MNP and non-MNP prefixes in service. Gateways process carrier packets received over the secured spanning tree that are addressed to themselves, while forwarding all other carrier packets to the next hop also via the secured spanning tree. Gateways forward carrier packets received over the unsecured spanning tree to the next hop either via the unsecured spanning tree or via direct encapsulation if the next hop is on the same OMNI link segment.

First-Hop Segment (FHS) Proxy/Server

a Proxy/Server for a source Client's underlay interface that forwards the Client's packets into the segment routing topology. FHS Proxy/Servers also act as intermediate forwarding nodes to facilitate RS/RA exchanges between a Client and its Hub Proxy/Server.

Hub Proxy/Server

a single Proxy/Server selected by a Client that injects the Client's MNP into the BGP routing system and provides a designated router service for all of the Client's underlay interfaces. Clients often select the first FHS Proxy/Server they coordinate with to serve in the Hub role (as all FHS Proxy/Servers are equally capable candidates to serve in that capacity), however the Client can also select any available Proxy/Server for the OMNI link (as there is no requirement that the Hub must also be one of the Client's FHS Proxy/Servers).

Last-Hop Segment (LHS) Proxy/Server

a Proxy/Server for an underlay interface of the target Client that forwards packets received from the segment routing topology to the target Client over that interface.

Segment Routing Topology (SRT)

a Multinet OMNI link forwarding region between FHS and LHS Proxy/Servers. FHS/LHS Proxy/Servers and SRT Gateways span the OMNI link on behalf of source/target Client pairs. The SRT maintains a spanning tree established through BGP peerings between Gateways and Proxy/Servers. Each SRT segment includes Gateways in a "hub" and Proxy/Servers in "spokes", while adjacent segments are interconnected by Gateway-Gateway peerings. The BGP peerings are configured over both secured and unsecured underlay network paths such that a secured spanning tree is available for critical control messages while other messages can use the unsecured spanning tree.

Mobile Node (MN)

an AERO Client and all of its downstream-attached networks that move together as a single unit, i.e., an end system that connects an Internet of Things.

Mobile Router (MR)

a MN's on-board router that forwards original IP packets between any downstream-attached networks and the OMNI link. The MR is the MN entity that hosts the AERO Client.

Route Optimization Source (ROS)

the AERO node nearest the source that initiates route optimization. The ROS may be a FHS Proxy/Server or Relay for the source, or may be the source Client itself.

Route Optimization responder (ROR)

the AERO node that responds to route optimization requests on behalf of the target. The ROR may be either the target MNP Client itself, the Client's current Hub Proxy/Server or a Relay for a non-MNP target.

Potential Router List (PRL)

a geographically and/or topologically referenced list of addresses of all Proxy/Servers within the same OMNI link. Each OMNI link has its own PRL.

Distributed Mobility Management (DMM)

a BGP-based overlay routing service coordinated by Proxy/Servers and Gateways that tracks all Proxy/Server-to-Client associations.

Mobility Service (MS)

the collective set of all Proxy/Servers, Gateways and Relays that provide the AERO Service to Clients.

Multilink Forwarding Information Base (MFIB)

A forwarding table on each AERO/OMNI source, destination and intermediate node that includes Multilink Forwarding Vectors (MFV) with both next hop forwarding instructions and context for reconstructing compressed headers for specific underlay interface pairs used to communicate with peers.

Multilink Forwarding Vector (MFV)

An MFIB entry that includes soft state for each underlay interface pairwise communication session between peer OMNI nodes. MFVs are identified by both a next-hop and previous-hop MFV Index (MFVI), with the next-hop established based on an IPv6 ND solicitation and the previous hop established based on the solicited IPv6 ND advertisement response.

Multilink Forwarding Vector Index (MFVI)

A 4 octet value selected by an AERO/OMNI node when it creates an MFV, then advertises to either a next-hop or previous-hop. AERO/OMNI intermediate nodes assign two distinct local MFVIs for each MFV and advertise one to the next-hop and the other to the previous-hop. AERO/OMNI end systems assign and advertise a single MFVI. AERO/OMNI nodes also discover the remote MFVIs advertised by other nodes that indicate a value the remote node is willing to accept.

Throughout the document, the simple terms "Host", "Client", "Proxy/Server", "Gateway" and "Relay" refer to "AERO Host", "AERO Client", "AERO Proxy/Server", "AERO Gateway" and "AERO Relay", respectively. Capitalization is used to distinguish these terms from other common Internetworking uses in which they appear without capitalization.

The terminology of IPv6 ND [RFC4861], DHCPv6 [RFC8415] and OMNI [I-D.templin-6man-omni] (including the names of node variables, messages and protocol constants) is used throughout this document. The terms "All-Routers multicast", "All-Nodes multicast", "Solicited-Node multicast" and "Subnet-Router anycast" are defined in [RFC4291]. Also, the term "IP" is used to generically refer to either Internet Protocol version, i.e., IPv4 [RFC0791] or IPv6 [RFC8200].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Automatic Extended Route Optimization (AERO)

The following sections specify the operation of IP over OMNI links using the AERO service:

3.1. AERO Node Types

AERO Hosts configure an OMNI interface over an underlay interface connected to a Client's ENET and coordinate with both other AERO Hosts and Clients over the ENET. As an implementation matter, the Host either assigns the same (MNP-based) IP address from the underlay interface to the OMNI interface, or configures the "OMNI interface" as a virtual sublayer of the underlay interface itself. AERO Hosts treat the ENET as an ANET, and treat the upstream Client for the ENET as a Proxy/Server. AERO Hosts are seen as OMNI link termination endpoints.

AERO Clients can be deployed as fixed infrastructure nodes close to end systems, or as Mobile Nodes (MNs) that can change their network attachment points dynamically. AERO Clients configure OMNI interfaces over underlay interfaces with addresses that may change due to mobility. AERO Clients register their Mobile Network Prefixes (MNPs) with the AERO service, and distribute the MNPs to ENETs (which may connect AERO Hosts and other Clients). AERO Clients provide Proxy/Server-like services for Hosts and other Clients on downstream-attached ENETs.

AERO Gateways, Proxy/Servers and Relays are critical infrastructure elements in fixed (i.e., non-mobile) INET deployments and hence have permanent and unchanging INET addresses. Together, they constitute the AERO service which provides an OMNI link virtual overlay for connecting AERO Clients and Hosts. AERO Gateways (together with Proxy/Servers) provide the secured backbone supporting infrastructure for a Segment Routing Topology (SRT) spanning tree for the OMNI link.

AERO Gateways forward carrier packets both within the same SRT segment and between disjoint SRT segments based on an IPv6 encapsulation mid-layer known as the OMNI Adaptation Layer (OAL) [I-D.templin-6man-omni]. The OMNI interface and OAL provide a virtual bridging service, since the inner IP TTL/Hop Limit is not decremented. Each Gateway also peers with Proxy/Servers and other Gateways in a dynamic routing protocol instance to provide a Distributed Mobility Management (DMM) service for the list of active MNPs (see Section 3.2.3). Gateways assign one or more Mobility Service Prefixes (MSPs) to the OMNI link and configure secured tunnels with Proxy/Servers, Relays and other Gateways; they further maintain forwarding table entries for each MNP or non-MNP prefix in service on the OMNI link.

AERO Proxy/Servers distributed across one or more SRT segments provide default forwarding and mobility/multilink services for AERO Client mobile nodes. Each Proxy/Server also peers with Gateways in a dynamic routing protocol instance to advertise its list of associated MNPs (see Section 3.2.3). Hub Proxy/Servers provide prefix delegation/registration services and track the mobility/multilink profiles of each of their associated Clients, where each delegated prefix becomes an MNP taken from an MSP. Proxy/Servers at ANET/INET boundaries provide a forwarding service for ANET Clients and Hosts to communicate with peers in external INETs, while Proxy/Servers in the open INET provide an authentication service for INET Client IPv6 ND messages but only a secondary forwarding service when the Client cannot forward directly to a peer or Gateway. Source Clients securely coordinate with target Clients by sending control messages via a First-Hop Segment (FHS) Proxy/Server which forwards them over the SRT spanning tree to a Last-Hop Segment (LHS) Proxy/Server which finally forwards them to the target.

AERO Relays are Proxy/Servers that provide forwarding services to exchange original IP packets between the OMNI link and nodes on other links/networks. Relays run a dynamic routing protocol to discover any non-MNP prefixes in service on other links/networks, and Relays that connect to larger Internetworks (such as the Internet) may originate default routes. The Relay redistributes OMNI link MSP(s) into other links/networks, and redistributes non-MNP prefixes via OMNI link Gateway BGP peerings.

3.2. The AERO Service over OMNI Links

3.2.1. AERO/OMNI Reference Model

Figure 1 presents the basic OMNI link reference model:

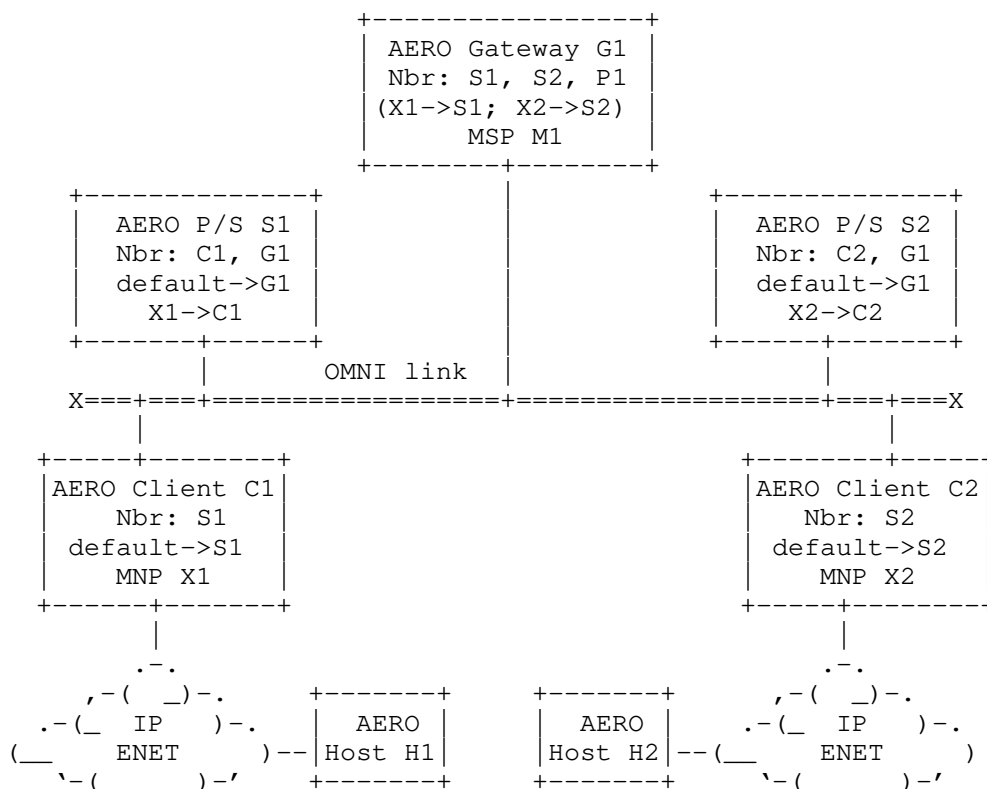


Figure 1: AERO/OMNI Reference Model

In this model:

- * the OMNI link is an overlay network service configured over one or more underlay SRT segments which may be managed by different administrative authorities and have incompatible protocols and/or addressing plans.
- * AERO Gateway G1 aggregates Mobility Service Prefix (MSP) M1, discovers Mobile Network Prefixes (MNPs) X* and advertises the MSP via BGP peerings over secured tunnels to Proxy/Servers (S1, S2). Gateways provide the backbone for an SRT spanning tree for the OMNI link.

- * AERO Proxy/Servers S1 and S2 configure secured tunnels with Gateway G1 and also provide mobility, multilink, multicast and default router services for the MNPs of their associated Clients C1 and C2. (Proxy/Servers that act as Relays can also advertise non-MNP routes for non-mobile correspondent nodes the same as for MNP Clients.)
- * AERO Clients C1 and C2 associate with Proxy/Servers S1 and S2, respectively. They receive MNP delegations X1 and X2, and also act as default routers for their associated physical or internal virtual ENETs.
- * AERO Hosts H1 and H2 attach to the ENETs served by Clients C1 and C2, respectively.

An OMNI link configured over a single underlay network appears as a single unified link with a consistent addressing plan; all nodes on the link can exchange carrier packets via simple L2 encapsulation (i.e., following any necessary NAT traversal) since the underlay is connected. In common practice, however, OMNI links are often configured over an SRT spanning tree that bridges multiple distinct underlay network segments managed under different administrative authorities (e.g., as for worldwide aviation service providers such as ARINC, SITA, Inmarsat, etc.). Individual underlay networks may also be partitioned internally, in which case each internal partition appears as a separate segment.

The addressing plan of each SRT segment is consistent internally but will often bear no relation to the addressing plans of other segments. Each segment is also likely to be separated from others by network security devices (e.g., firewalls, proxys, packet filtering gateways, etc.), and disjoint segments often have no common physical link connections. Therefore, nodes can only be assured of exchanging carrier packets directly with correspondents in the same segment, and not with those in other segments. The only means for joining the segments therefore is through inter-domain peerings between AERO Gateways.

The OMNI link spans multiple SRT segments using the OMNI Adaptation Layer (OAL) [I-D.templin-6man-omni] to provide the network layer with a virtual abstraction similar to a bridged campus LAN. The OAL is an OMNI interface sublayer that inserts a mid-layer IPv6 encapsulation header for inter-segment forwarding (i.e., bridging) without decrementing the network-layer TTL/Hop Limit of the original IP packet. An example OMNI link SRT is shown in Figure 2:

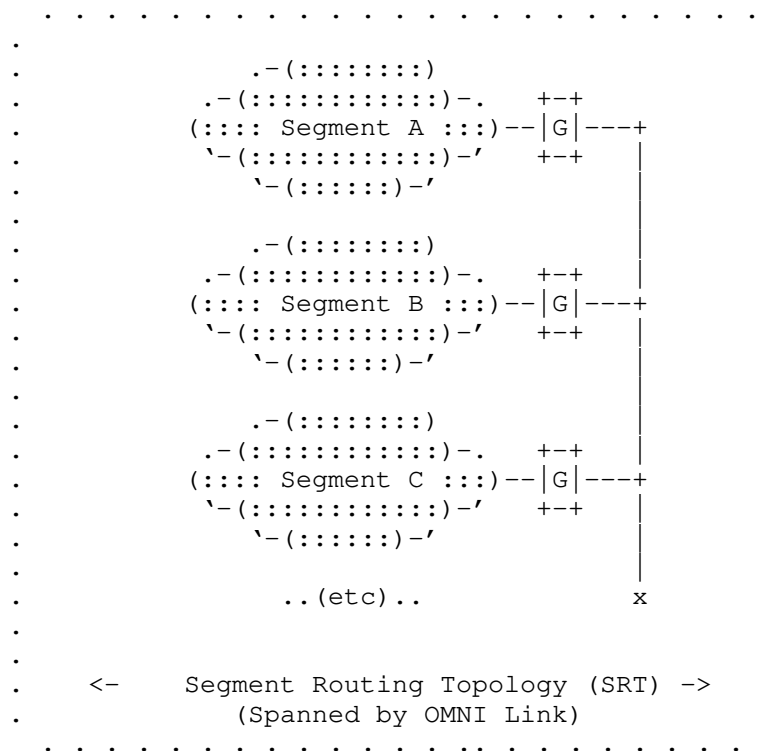


Figure 2: OMNI Link Segment Routing Topology (SRT)

Gateway, Proxy/Server and Relay OMNI interfaces are configured over both secured tunnels and open INET underlay interfaces within their respective SRT segments. Within each segment, Gateways configure "hub-and-spokes" BGP peerings with Proxy/Servers and Relays as "spokes". Adjacent SRT segments are joined by Gateway-to-Gateway peerings to collectively form a spanning tree over the entire SRT. The "secured" spanning tree supports authentication and integrity for critical control plane messages. The "unsecured" spanning tree conveys ordinary carrier packets without security codes and that must be treated by destinations according to data origin authentication procedures. AERO nodes can employ route optimization to cause carrier packets to take more direct paths between OMNI link neighbors without having to follow strict spanning tree paths.

The AERO Multinet service concatenates SRT segments to form larger networks through Gateway-to-Gateway peerings as originally described in the "Catenet Model for Internetworking" [IEN48]; especially Figure 2 follows directly from the illustrations in [IEN48-2]. The Catenet model inspired the global public Internet as it is known today, while AERO applies the Catenet concepts to provide true Multinet services for the future architecture.

3.2.2. Addressing and Node Identification

AERO nodes on OMNI links use the Link-Local Address (LLA) prefix `fe80::/64` [RFC4291] to assign LLAs to the OMNI interface in a latent state and do not employ LLAs for any operational purposes (instead, the LLAs are assigned solely to satisfy the requirements of [RFC4861]). AERO Clients configure LLAs constructed from MNPs (i.e., "LLA-MNPs") while AERO infrastructure nodes construct LLAs based on 56-bit random values ("LLA-RNDs") per [I-D.templin-6man-omni]. Non-MNP routes are also represented the same as for MNPs, but may include a prefix that is not properly covered by an MSP.

AERO nodes also use the Unique Local Address (ULA) prefix `fd00::/8` followed by a pseudo-random 40-bit Global ID to form the prefix `{ULA}::/48`, then include a 16-bit Subnet ID '*' to form the prefix `{ULA*}::/64` [RFC4291]. The AERO node then uses the prefix `{ULA*}::/64` to form "ULA-MNPs" or "ULA-RNDs" as specified in [I-D.templin-6man-omni] to support OAL addressing. (The prefix `{ULA*}::/64` appearing alone and with no suffix represents "default" for that prefix.)

AERO Clients also use Temporary Local Addresses (TLAs) and eXtended Local Addresses (XLAs) constructed per [I-D.templin-6man-omni], where TLAs are distinguished from ordinary ULAs based on the prefix `fd00::/16` and XLAs are distinguished from ULAs/TLAs based on the prefix `fd00::/64`. Clients use TLA-RNDs only in initial control message exchanges until a stable MNP is assigned, but may sometimes also use them for sustained communications within a local routing region. AERO nodes use XLA-MNPs to provide forwarding information for the global routing table as well as IPv6 ND message and OAL addressing information.

AERO MSPs, MNPs and non-MNP routes are typically based on Global Unicast Addresses (GUAs), but in some cases may be based on IPv4 private addresses [RFC1918] or IPv6 ULA-D's [RFC4193]. A GUA block is also reserved for OMNI link anycast purposes. See [I-D.templin-6man-omni] for a full specification of LLAs, ULAs, TLAs, XLAs, GUAs and anycast addresses used by AERO nodes on OMNI links.

Finally, AERO Clients and Proxy/Servers configure node identification values as specified in [I-D.templin-6man-omni].

3.2.3. AERO Routing System

The AERO routing system comprises a private Border Gateway Protocol (BGP) [RFC4271] service coordinated between Gateways and Proxy/Servers (Relays also engage in the routing system as simplified Proxy/Servers). The service supports carrier packet forwarding at a layer below IP and does not interact with the public Internet BGP routing system, but supports redistribution of information for other links and networks connected by Relays.

In a reference deployment, each Proxy/Server is configured as an Autonomous System Border Router (ASBR) for a stub Autonomous System (AS) using a 32-bit AS Number (ASN) [RFC4271] that is unique within the BGP instance, and each Proxy/Server further uses eBGP to peer with one or more Gateways but does not peer with other Proxy/Servers. Each SRT segment in the OMNI link must include one or more Gateways in a "hub" AS, which peer with the Proxy/Servers within that segment as "spoke" ASes. All Gateways within the same segment are members of the same hub AS, and use iBGP to maintain a consistent view of all active routes currently in service. The Gateways of different segments peer with one another using eBGP.

Gateways maintain forwarding table entries only for ULA prefixes for infrastrucutre elements and XLA-MNPs corresponding to MNP and non-MNP routes that are currently active; Gateways also maintain black-hole routes for the OMNI link MSPs so that carrier packets destined to non-existent more-specific routes are dropped with a Destination Unreachable message returned. In this way, Proxy/Servers and Relays have only partial topology knowledge (i.e., they only maintain routing information for their directly associated Clients and non-AERO links) and they forward all other carrier packets to Gateways which have full topology knowledge.

Each OMNI link segment assigns a unique sub-prefix of {ULA}::/48 known as the "SRT prefix". For example, a first segment could assign {ULA}:1000::/56, a second could assign {ULA}:2000::/56, a third could assign {ULA}:3000::/56, etc. Within each segment, each Proxy/Server configures a ULA-RND within the segment's SRT prefix with a 56-bit random value in the interface identifier as specified in [I-D.templin-6man-omni].

The administrative authorities for each segment must therefore coordinate to assure mutually-exclusive ULA prefix assignments, but internal provisioning of ULAs is an independent local consideration for each administrative authority. For each ULA prefix, the

Gateway(s) that connect that segment assign the all-zero's address of the prefix as a Subnet Router Anycast address. For example, the Subnet Router Anycast address for {ULA}:1023::/64 is simply {ULA}:1023::/64.

ULA prefixes are statically represented in Gateway forwarding tables. Gateways join multiple SRT segments into a unified OMNI link over multiple diverse network administrative domains. They support a virtual bridging service by first establishing forwarding table entries for their ULA prefixes either via standard BGP routing or static routes. For example, if three Gateways ('A', 'B' and 'C') from different segments serviced {ULA}:1000::/60, {ULA}:2000::/56 and {ULA}:3000::/56 respectively, then the forwarding tables in each Gateway appear as follows:

A: {ULA}:1000::/56->local, {ULA}:2000::/56->B, {ULA}:3000::/56->C

B: {ULA}:1000::/56->A, {ULA}:2000::/56->local, {ULA}:3000::/56->C

C: {ULA}:1000::/56->A, {ULA}:2000::/56->B, {ULA}:3000::/56->local

These forwarding table entries rarely change, since they correspond to fixed infrastructure elements in their respective segments.

MNP (and non-MNP) ULA routes are instead dynamically advertised in the AERO routing system by Proxy/Servers and Relays that provide service for their corresponding MNPs. The routes are advertised as XLA-MNP prefixes, i.e., as fd00::{MNP} (see: [I-D.templin-6man-omni]). For example, if three Proxy/Servers ('D', 'E' and 'F') service the MNPs 2001:db8:1000:2000::/56, 2001:db8:3000:4000::/56 and 2001:db8:5000:6000::/56 then the routing system would include:

D: fd00::2001:db8:1000:2000/120

E: fd00::2001:db8:3000:4000/120

F: fd00::2001:db8:5000:6000/120

A full discussion of the BGP-based routing system used by AERO is found in [I-D.ietf-rtgwg-atn-bgp].

3.2.4. Segment Routing Topologies (SRTs)

The distinct {ULA}::/48 prefixes in an OMNI link domain identify distinct Segment Routing Topologies (SRTs). Each SRT is a mutually-exclusive OMNI link overlay instance using a distinct set of ULAs, and emulates a bridged campus LAN service for the OMNI link. In some cases (e.g., when redundant topologies are needed for fault tolerance and reliability) it may be beneficial to deploy multiple SRTs that act as independent overlay instances. A communication failure in one instance therefore will not affect communications in other instances.

Each SRT is identified by a distinct value in the 40-bit ULA Global ID field and assigns an OMNI IPv6 anycast address used for OMNI interface determination in Safety-Based Multilink (SBM) as discussed in [I-D.templin-6man-omni]. Each OMNI interface further applies Performance-Based Multilink (PBM) internally.

The Gateways and Proxy/Servers of each independent SRT engage in BGP peerings to form a spanning tree with the Gateways in non-leaf nodes and the Proxy/Servers in leaf nodes. The spanning tree is configured over both secured and unsecured underlay network paths. The secured spanning tree is used to convey secured control messages between Proxy/Servers and Gateways, while the unsecured spanning tree forwards data messages and/or unsecured control messages.

Each SRT segment is identified by a unique ULA prefix used by all Proxy/Servers and Gateways in the segment. Each AERO node must therefore discover an SRT prefix that correspondents can use to determine the correct segment, and must publish the SRT prefix in IPv6 ND messages.

Note: The distinct ULA prefixes in an OMNI link domain can be carried either in a common BGP routing protocol instance for all OMNI links or in distinct BGP routing protocol instances for different OMNI links. In some SBM environments, such separation may be necessary to ensure that distinct OMNI links do not include any common infrastructure elements as single points of failure. In other environments, carrying the ULAs of multiple OMNI links within a common routing system may be acceptable.

3.2.5. Segment Routing For OMNI Link Selection

Original IPv6 sources can direct IPv6 packets to an AERO node by including a standard IPv6 Segment Routing Header (SRH) [RFC8754] with the OMNI IPv6 anycast address for the selected OMNI link as either the IPv6 destination or as an intermediate hop within the SRH. This allows the original source to determine the specific OMNI link SRT an original IPv6 packet will traverse when there may be multiple

alternatives.

When an AERO node processes the SRH and forwards the original IPv6 packet to the correct OMNI interface, the OMNI interface writes the next IPv6 address from the SRH into the IPv6 destination address and decrements Segments Left. If decrementing would cause Segments Left to become 0, the OMNI interface deletes the SRH before forwarding. This form of Segment Routing supports Safety-Based Multilink (SBM).

3.3. OMNI Interface Characteristics

OMNI interfaces are virtual interfaces configured over one or more underlay interfaces classified as follows:

- * ANET interfaces connect to a protected and secured ANET that is separated from the open INET by Proxy/Servers. The ANET interface may be either on the same L2 link segment as a Proxy/Server, or separated from a Proxy/Server by multiple IP hops. (Note that NATs may appear internally within an ANET and may require NAT traversal on the path to the Proxy/Server the same as for the INET case.)
- * INET interfaces connect to an INET either natively or through one or several IPv4 Network Address Translators (NATs). Native INET interfaces have global IP addresses that are reachable from any INET correspondent. NATed INET interfaces typically have private IP addresses and connect to a private network behind one or more NATs with the outermost NAT providing INET access.
- * ENET interfaces connect a Client's downstream-attached networks, where the Client provides forwarding services for ENET Host and Client communications to remote peers. An ENET be as simple as a small stub network that travels with a mobile Client (e.g., an Internet-of-Things) to as complex as a large private enterprise network that the Client connects to a larger ANET or INET.
- * VPned interfaces use security encapsulation over an underlay network to a Client or Proxy/Server acting as a Virtual Private Network (VPN) gateway. Other than the link-layer encapsulation format, VPned interfaces behave the same as for Direct interfaces.
- * Direct (aka "point-to-point") interfaces connect directly to a Client or Proxy/Server without crossing any networked paths. An example is a line-of-sight link between a remote pilot and an unmanned aircraft.

OMNI interfaces use OAL encapsulation and fragmentation as discussed in Section 3.6. OMNI interfaces use L2 encapsulation (see: Section 3.6) to exchange carrier packets with OMNI link neighbors over INET or VPNed interfaces as well as over ANET interfaces for which the Client and FHS Proxy/Server may be multiple IP hops away. OMNI interfaces use link-layer encapsulation only (i.e., and no other L2 encapsulations) over Direct underlay interfaces or ANET interfaces when the Client and FHS Proxy/Server are known to be on the same underlay link.

OMNI interfaces maintain a neighbor cache for tracking per-neighbor state the same as for any interface. OMNI interfaces use IPv6 ND messages including Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA) and Redirect for neighbor cache management. In environments where spoofing may be a threat, OMNI neighbors should invoke OAL Identification window synchronization in their IPv6 ND message exchanges.

OMNI interfaces send IPv6 ND messages with an OMNI option formatted as specified in [I-D.templin-6man-omni]. The OMNI option includes prefix registration information, Interface Attributes and/or Multilink Forwarding Parameters containing link information parameters for the OMNI interface's underlay interfaces and any other per-neighbor information.

A Host's OMNI interface is configured over an underlay interface connected to an ENET provided by an upstream Client. From the Host's perspective, the ENET appears as an ANET and the upstream Client appears as a Proxy/Server. The Host does not provide OMNI intermediate node services and is therefore a logical termination point for the OMNI link.

A Client's OMNI interface may be configured over multiple ANET/INET underlay interfaces. For example, common mobile handheld devices have both wireless local area network ("WLAN") and cellular wireless links. These links are often used "one at a time" with low-cost WLAN preferred and highly-available cellular wireless as a standby, but a simultaneous-use capability could provide benefits. In a more complex example, aircraft frequently have many wireless data link types (e.g. satellite-based, cellular, terrestrial, air-to-air directional, etc.) with diverse performance and cost properties.

If a Client's multiple ANET/INET underlay interfaces are used "one at a time" (i.e., all other interfaces are in standby mode while one interface is active), then successive IPv6 ND messages all include OMNI option Multilink Forwarding Parameters sub-options with the same underlay interface index. In that case, the Client would appear to have a single underlay interface but with a dynamically changing link-layer address.

If the Client has multiple active ANET/INET underlay interfaces, then from the perspective of IPv6 ND it would appear to have multiple link-layer addresses. In that case, IPv6 ND message OMNI options MAY include Interface Attributes and/or Multilink Forwarding Parameters sub-options with different underlay interface indexes.

Proxy/Servers on the open Internet include only a single INET underlay interface. INET Clients therefore discover only the INADDR information for the Proxy/Server's INET interface. Proxy/Servers on an ANET/INET boundary include both an ANET and INET underlay interface. ANET Clients therefore must discover both the ANET and INET INADDR information for the Proxy/Server.

Gateway and Proxy/Server OMNI interfaces are configured over underlay interfaces that provide both secured tunnels for carrying IPv6 ND and BGP protocol control plane messages and open INET access for carrying unsecured messages. The OMNI interface configures a ULA-RND and acts as an OAL source to encapsulate and fragment original IP packets while presenting the resulting carrier packets over the secured or unsecured underlay paths. Note that Gateway and Proxy/Server end-to-end transport protocol sessions used by the BGP are run directly over the OMNI interface and use ULA-RND source and destination addresses. The OMNI interface employs the OAL to encapsulate the original IP packets for these sessions as carrier packets (i.e., even though the OAL header may use the same ULAs as the original IP header) and forwards them over the secured underlay path.

3.4. OMNI Interface Initialization

AERO Proxy/Servers, Clients and Hosts configure OMNI interfaces as their point of attachment to the OMNI link. AERO nodes assign the MSPs for the link to their OMNI interfaces (i.e., as a "route-to-interface") to ensure that original IP packets with destination addresses covered by an MNP not explicitly associated with another interface are directed to an OMNI interface.

OMNI interface initialization procedures for Proxy/Servers, Clients Hosts and Gateways are discussed in the following sections.

3.4.1. AERO Proxy/Server and Relay Behavior

When a Proxy/Server enables an OMNI interface, it assigns a ULA-RND appropriate for the given OMNI link SRT segment. The Proxy/Server also configures secured tunnels and engages in BGP routing protocol sessions with one or more neighboring Gateways.

The OMNI interface provides a single interface abstraction to the IP layer, but internally includes an NBMA nexus for sending carrier packets to OMNI interface neighbors over underlay INET interfaces and secured tunnels. The Proxy/Server further configures a service to facilitate IPv6 ND exchanges with AERO Clients and manages per-Client neighbor cache entries and IP forwarding table entries based on control message exchanges.

Relays are simply Proxy/Servers that run a dynamic routing protocol to redistribute routes between the OMNI interface and INET/ENET interfaces (see: Section 3.2.3). The Relay provisions MNPs to networks on the INET/ENET interfaces (i.e., the same as a Client would do) and advertises the MSP(s) for the OMNI link over the INET/ENET interfaces. The Relay further provides an attachment point of the OMNI link to a non-MNP-based global topology.

3.4.2. AERO Client Behavior

When a Client enables an OMNI interface, it assigns either an XLA-MNP or a TLA and sends OMNI-encapsulated RS messages over its ANET/INET underlay interfaces to an FHS Proxy/Server, which coordinates with a Hub Proxy/Server that returns an RA message with corresponding parameters. The RS/RA messages may pass through one or more NATs in the path between the Client and FHS Proxy/Server. (Note: if the Client used a TLA in its initial RS messages, it may discover ULA-MNPs in the corresponding RAs that it receives from FHS Proxy/Servers and begin using these new addresses. If the Client is operating outside the context of AERO infrastructure such as in a Mobile Ad-hoc Network (MANET), however, it may continue using TLAs for Client-to-Client communications until it encounters an infrastructure element that can delegate MNPs.)

A Client can further extend the OMNI link over its (downstream) ENET interfaces where it provides a first-hop router for Hosts and other AERO Clients connected to the ENET. A downstream Client that connects via the ENET serviced by an upstream Client can in turn service further downstream ENETs that connect other Hosts and Clients. This OMNI link extension can be applied recursively over a "chain" of ENET Clients.

3.4.3. AERO Host Behavior

When a Host enables an OMNI interface, it assigns an address taken from the ENET underlay interface which may itself be a GUA delegated by the upstream Client. The Host does not assign a link-local address to the OMNI interface, since no autoconfiguration is necessary on that interface. (As an implementation matter, the Host could instead configure the "OMNI interface" as a virtual sublayer of the ENET underlay interface itself.)

The Host sends OMNI-encapsulated RS messages over its ENET underlay interface to the upstream Client, which returns encapsulated RAs and provides routing services in the same fashion that Proxy/Servers provides services for Clients. Hosts represent the leaf end systems in recursively-nested chain of concatenated ENETs, i.e., they represent terminating endpoints for the OMNI link.

3.4.4. AERO Gateway Behavior

AERO Gateways configure an OMNI interface and assign a ULA-RND and corresponding Subnet Router Anycast address for each OMNI link SRT segment they connect to. Gateways configure secured tunnels with Proxy/Servers in the same SRT segment and other Gateways in the same (or an adjacent) SRT segment. Gateways then engage in a BGP routing protocol session with neighbors over the secured spanning tree (see: Section 3.2.3).

3.5. OMNI Interface Neighbor Cache Maintenance

Each Client, Proxy/Server and Gateway OMNI interface maintains a conceptual neighbor cache that includes a Neighbor Cache Entry (NCE) for each of its active neighbors on the OMNI link per [RFC4861]. Each NCE is indexed by the network layer address of the neighbor and determines the context for Identification verification. Clients and Proxy/Servers maintain NCEs through RS/RA exchanges, and also maintain NCEs for any active correspondent peers through NS/NA exchanges.

Hosts also maintain NCEs for Clients and other Hosts through the exchange of RS/RA or NS/NA messages. Each NCE is indexed by the address assigned to the Host ENET interface, which is the same address used for OMNI L2 encapsulation (i.e., without the insertion of an OAL header). This encapsulation format identifies the NCE as a Host-based entry where the Host is a leaf end system in the recursively extended OMNI link.

Gateways also maintain NCEs for Clients within their local segments based on NS/NA route optimization messaging (see: Section 3.13.4). When a Gateway creates/updates a NCE for a local segment Client based on NS/NA route optimization, it also maintains MFVI and INADDR state for messages destined to this local segment Client.

Proxy/Servers add an additional state DEPARTED to the list of NCE states found in Section 7.3.2 of [RFC4861]. When a Client terminates its association, the Proxy/Server OMNI interface sets a "DepartTime" variable for the NCE to "DEPART_TIME" seconds. DepartTime is decremented unless a new IPv6 ND message causes the state to return to REACHABLE. While a NCE is in the DEPARTED state, the Proxy/Server forwards carrier packets destined to the target Client to the Client's new FHS/Hub Proxy/Server instead. It is RECOMMENDED that DEPART_TIME be set to the default constant value 10 seconds to accept any carrier packets that may be in flight. When DepartTime decrements to 0, the NCE is deleted.

Clients determine the service profiles for their FHS and Hub Proxy/Servers by setting the N/A/U flags in a Neighbor Coordination sub-option of the first OMNI option in RS messages. When the N/A/U flags are clear, Proxy/Servers forward all NS/NA messages to the Client, while the Client performs mobility update signaling through the transmission of uNA messages to all active neighbors following a mobility event. However, in some environments this may result in excessive NS/NA control message overhead especially for Clients connected to low-end data links.

To minimize NS/NA message overhead, Clients can set the N/A/U flags in the OMNI option Neighbor Coordination header of RS messages they send. If the N flag is set, the FHS Proxy/Server that forwards the RS message assumes the role of responding to NS(NUD) messages and maintains peer NCEs associated with the NCE for this Client. If the A flag is set, the Hub Proxy/Server that processes the RS message assumes the role of responding to NS(AR) messages on behalf of this Client NCE. If the U flag is set, the Hub Proxy/Server that processes the RS message becomes responsible for maintaining a "Report List" of sources from which it has received an NS(AR) for this Client NCE. The Hub Proxy/Server maintains each Report List entry for REPORT_TIME seconds, and sends uNA messages to each member of the Report List when it receives a Client mobility update indication (e.g., through receipt of an RS with updated Interface Attributes, Traffic Selectors, etc.).

Clients and their Hub Proxy/Servers have full knowledge of the Client's current underlay Interface Attributes, while FHS Proxy/Servers acting in "proxy" mode have knowledge of only the individual Client underlay interfaces they service. Clients determine their FHS and Hub Proxy/Server service models by setting the N/A/U flags in the RS messages they send as discussed above.

Clients act as RORs on their own behalf when they receive an NS(AR) from an ROS via their Hub Proxy/Server (Relays instead act as RORs on behalf of non-MNP targets specific to other links/networks that the Relay services and/or "default"). The ROR returns a NA(AR) response to the ROS, which creates or updates a NCE for the target network-layer and link-layer addresses. The ROS then (re)sets ReachableTime for the NCE to REACHABLE_TIME seconds and performs reachability tests over specific underlay interface pairs to determine paths for forwarding carrier packets directly to the target. The ROS otherwise decrements ReachableTime while no further solicited NA messages arrive. It is RECOMMENDED that REACHABLE_TIME be set to the default constant value 30 seconds as specified in [RFC4861].

AERO nodes also use the value MAX_UNICAST_SOLICIT to limit the number of NS messages sent when a correspondent may have gone unreachable, the value MAX_RTR_SOLICITATIONS to limit the number of RS messages sent without receiving an RA and the value MAX_NEIGHBOR_ADVERTISEMENT to limit the number of unsolicited NAs that can be sent based on a single event. It is RECOMMENDED that MAX_UNICAST_SOLICIT, MAX_RTR_SOLICITATIONS and MAX_NEIGHBOR_ADVERTISEMENT be set to 3 the same as specified in [RFC4861].

Different values for the above constants MAY be administratively set; however, if different values are chosen, all nodes on the link MUST consistently configure the same values. Most importantly, DEPART_TIME and REPORT_TIME SHOULD be set to a value that is sufficiently longer than REACHABLE_TIME to avoid packet loss due to stale route optimization state.

3.5.1. OMNI ND Messages

OMNI interfaces prepare IPv6 ND messages the same as for standard IPv6 ND, but also include a new option type termed the OMNI option [I-D.templin-6man-omni]. OMNI interfaces do not use LLAs as IPv6 ND message source and destination addresses, but instead use ULAs. This allows for multiple different OMNI links to be joined into a single link at some future time without requiring a global renumbering event.

For each IPv6 ND message, OMNI interfaces include one or more OMNI options (and any other ND message options) then completely populate all option information. If the OMNI interface includes an authentication signature, it sets the IPv6 ND message Checksum field to 0 and calculates the authentication signature over the entire length of the OAL packet or super-packet (beginning with a pseudo-header of the IPv6 header) but does not then proceed to calculate the IPv6 ND message checksum itself. Otherwise, the OMNI interface calculates the standard IPv6 ND message checksum over the OAL packet or super-packet and writes the value in the Checksum field. OMNI interfaces verify authentication and integrity of each IPv6 ND message received according to the specific check(s) included, and process the message further only following verification.

OMNI options include per-neighbor information that provides multilink forwarding, link-layer address and traffic selector information for the neighbor's underlay interfaces. This information is stored in the neighbor cache and provides the basis for the forwarding algorithm specified in Section 3.10. The information is cumulative and reflects the union of the OMNI information from the most recent IPv6 ND messages received from the neighbor; it is therefore not required that each IPv6 ND message contain all neighbor information.

The OMNI option is distinct from any Source/Target Link-Layer Address Options (S/TLLAOs) that may appear in an IPv6 ND message according to the appropriate IPv6 over specific link layer specification (e.g., [RFC2464]). If both an OMNI option and S/TLLAO appear, the former pertains to encapsulation addresses while the latter pertains to the native L2 address format of the underlay media.

OMNI interface IPv6 ND messages may also include other IPv6 ND options. In particular, solicitation messages may include a Nonce option if required for verification of advertisement replies. If an OMNI IPv6 ND solicitation message includes a Nonce option, the advertisement reply must echo the same Nonce. If an OMNI IPv6 ND advertisement message includes a Timestamp option, the recipient should check the Timestamp to determine if the message is current.

AERO Clients send RS messages to the link-scoped All-Routers multicast address or a ULA-RND while using unicast or anycast L2 addresses. AERO Proxy/Servers respond by returning unicast RA messages. During the RS/RA exchange, AERO Clients and Proxy/Servers include state synchronization parameters to establish Identification windows and other state.

AERO Hosts and Clients on ENET underlay networks send RS messages to the link-scoped All-Routers multicast address, a ULA-RND of a remote Hub Proxy/Server or the ULA-MNP of an upstream Client while using unicast or anycast L2 addresses. The upstream AERO Client responds by returning a unicast RA message.

AERO nodes use NS/NA messages for the following purposes:

- * NS/NA(AR) messages are used for address resolution and optionally to establish sequence number windows. The ROS sends an NS(AR) to the solicited-node multicast address of the target, and an ROR with addressing information for the target returns a unicast NA(AR) that contains current, consistent and authentic target address resolution information. NS/NA(AR) messages must be secured.
- * NS/NA(NUD) messages are used to establish multilink forwarding state and determine target reachability. The source sends an NS(NUD) to the unicast address of the target while naming a specific underlay interface pair, and the target returns a unicast NA(NUD). NS/NA(NUD) messages that use an in-window sequence number and do not update any other state need not include an authentication signature but instead must include an IPv6 ND message checksum. NS/NA(NUD) messages may also be used to establish window synchronization and/or MFIB state, in which case the messages must be secured.
- * Unsolicited NA (uNA) messages are used to signal addressing and/or other neighbor state changes (e.g., address changes due to mobility, signal degradation, traffic selector updates, etc.). uNA messages that update state information must be secured.
- * NS/NA(DAD) messages are not used in AERO, since Duplicate Address Detection is not required.

Additionally, nodes may set the OMNI option PNG flag in NA/RA messages to receive a uNA response from the neighbor. The uNA response MUST set the ACK flag (without also setting the SYN or PNG flags) with the Acknowledgement field set to the Identification used in the PNG message.

3.5.2. OMNI Neighbor Advertisement Message Flags

As discussed in Section 4.4 of [RFC4861] NA messages include three flag bits R, S and O. OMNI interface NA messages treat the flags as follows:

- * R: The R ("Router") flag is set to 1 in the NA messages sent by all AERO/OMNI node types. Simple hosts that would set R to 0 do not occur on the OMNI link itself, but may occur on the downstream links of Clients and Relays.
- * S: The S ("Solicited") flag is set exactly as specified in Section 4.4. of [RFC4861], i.e., it is set to 1 for Solicited NAs and set to 0 for uNAs (both unicast and multicast).
- * O: The O ("Override") flag is set to 0 for solicited NAs returned by a Proxy/Server ROR and set to 1 for all other solicited and unsolicited NAs. For further study is whether solicited NAs for anycast targets apply for OMNI links. Since XLA-MNPs must be uniquely assigned to Clients to support correct IPv6 ND protocol operation, however, no role is currently seen for assigning the same XLA-MNP to multiple Clients.

3.5.3. OMNI Neighbor Window Synchronization

In secured environments (e.g., between secured spanning tree neighbors, between neighbors on the same secured ANET, etc.), OMNI interface neighbors can exchange OAL packets using randomly-initialized and monotonically-increasing Identification values (modulo 2^{32}) without window synchronization. In environments where spoofing is considered a threat, OMNI interface neighbors instead invoke window synchronization in NS/NA message exchanges to maintain send/receive window state in their respective neighbor cache entries as specified in [I-D.templin-6man-omni].

3.6. OMNI Interface Encapsulation and Fragmentation

When the network layer forwards an original IP packet into an OMNI interface, the interface locates or creates a Neighbor Cache Entry (NCE) that matches the destination. The OMNI interface then invokes the OMNI Adaptation Layer (OAL) as discussed in [I-D.templin-6man-omni] which encapsulates the packet in an IPv6 header to produce an OAL packet. For example, an original IP packet with source address 2001:db8:1:2::1 and destination address 2001:db8:1234:5678::1 might cause the OAL encapsulation header to include source address {XLA*}::2001:db8:1:2 (i.e., an XLA-MNP) and destination address {ULA*}::0012:3456:789a:bcde (i.e., a ULA-RND).

Following encapsulation, the OAL source then calculates a 2-octet checksum and fragments the OAL packet while including an identical Identification value for each fragment that must be within the window for the LHS Proxy/Server or the target Client itself. The OAL source finally includes the checksum as the final 2 octets of the final fragment, i.e., as a "trailer".

The OAL source next includes an identical Compressed Routing Header with 32-bit ID fields (CRH-32) [I-D.bonica-6man-comp-rtg-hdr] with each fragment containing one or more Multilink Forwarding Vector Indices (MFVIs) if necessary as discussed in Section 3.13. The OAL source can instead invoke OAL header compression by replacing the OAL IPv6 header, CRH-32 and Fragment Header with an OAL Compressed Header (OCH).

The OAL source finally encapsulates each resulting OAL fragment in L2 headers to form an OAL carrier packet, with source address set to its own L2 address (e.g., 192.0.2.100) and destination set to the L2 address of the next hop OAL intermediate node or destination (e.g., 192.0.2.1). The carrier packet encapsulation format in the above example is shown in Figure 3:

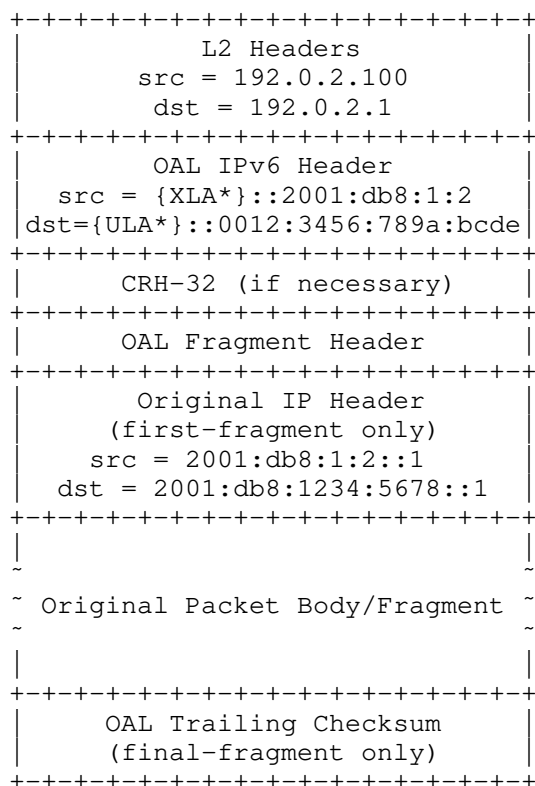


Figure 3: Carrier Packet Format

Note: the carrier packets exchanged by Hosts on ENETs do not include the OAL IPv6 or CRH-32 headers, i.e., the OAL encapsulation is NULL and only the Fragment Header and L2 encapsulations are included.

In this format, the OAL source encapsulates the original IP header and packet body/fragment in an OAL IPv6 header prepared according to [RFC2473], the CRH-32 is a Routing Header extension of the OAL header, the Fragment Header identifies each fragment, and the L2 headers are prepared as discussed in [I-D.templin-6man-omni]. The OAL source transmits each such carrier packet into the SRT spanning tree, where they are forwarded over possibly multiple OAL intermediate nodes until they arrive at the OAL destination.

The OMNI link control plane service distributes Client XLA-MNP prefix information that may change dynamically due to regional node mobility as well as XLA-MNP prefix information for Relay non-MNPs and per-segment ULA prefix information that rarely changes. OMNI link Gateways and Proxy/Servers use the information to establish and maintain a forwarding plane spanning tree that connects all nodes on the link. The spanning tree supports a carrier packet virtual bridging service according to link-layer (instead of network-layer) information, but may often include longer paths than necessary.

Each OMNI interface therefore also includes a Multilink Forwarding Information Base (MFIB) with Multilink Forwarding Vectors (MFVs) that can often provide more direct forwarding "shortcuts" that avoid strict spanning tree paths. As a result, the spanning tree is always available but OMNI interfaces can often use the MFIB to greatly improve performance and reduce load on critical infrastructure elements.

For carrier packets undergoing re-encapsulation at an OAL intermediate node, the OMNI interface decrements the OAL IPv6 header Hop Limit and discards the carrier packet if the Hop Limit reaches 0. The intermediate node next removes the L2 encapsulation headers from the first segment and re-encapsulates the packet in new L2 encapsulation headers for the next segment.

When an FHS Gateway receives a carrier packet with an OCH header that must be forwarded to an LHS Gateway over the unsecured spanning tree, it reconstructs the headers based on MFV state, inserts a CRH-32 immediately following the OAL header and adjusts the OAL payload length and destination address field. The FHS Gateway includes a single MFVI in the CRH-32 that will be meaningful to the LHS Gateway. When the LHS Gateway receives the carrier packet, it locates the MFV for the next hop based on the CRH-32 MFVI then re-applies header compression (resulting in the removal of the CRH-32) and forwards the carrier packet to the next hop.

3.7. OMNI Interface Decapsulation

OMNI interfaces (acting as OAL destinations) decapsulate and reassemble OAL packets into original IP packets destined either to the AERO node itself or to a destination reached via an interface other than the OMNI interface the original IP packet was received on. When carrier packets containing OAL fragments addressed to itself arrive, this OAL destination discards the NET encapsulation headers and reassembles to obtain the OAL packet or super-packet (see: [I-D.templin-6man-omni]). The OAL destination then verifies the OAL checksum, discards the OAL encapsulations to obtain the original IP packet(s) and finally forwards them to either the network layer or a next-hop on the OMNI link.

3.8. OMNI Interface Data Origin Authentication

AERO nodes employ simple data origin authentication procedures. In particular:

- * AERO Gateways and Proxy/Servers accept carrier packets received from the secured spanning tree.
- * AERO Proxy/Servers and Clients accept carrier packets and original IP packets that originate from within the same secured ANET.
- * AERO Clients and Relays accept original IP packets from downstream network correspondents based on ingress filtering.
- * AERO Hosts, Clients, Relays, Proxy/Servers and Gateways verify carrier packet L2 encapsulation addresses according to [I-D.templin-6man-omni].
- * AERO nodes accept carrier packets addressed to themselves with Identification values within the current window for the OAL source neighbor and drop any carrier packets with out-of-window Identification values. (AERO nodes may forward carrier packets not addressed to themselves without verifying the Identification value.)

AERO nodes silently drop any packets that do not satisfy the above data origin authentication procedures. Further security considerations are discussed in Section 6.

3.9. OMNI Interface MTU

The OMNI interface observes the link nature of tunnels, including the Maximum Transmission Unit (MTU), Maximum Reassembly Unit (MRU) and the role of fragmentation and reassembly [I-D.ietf-intarea-tunnels]. The OMNI interface employs an OMNI Adaptation Layer (OAL) that accommodates multiple underlay links with diverse MTUs while observing both a minimum and per-path Maximum Payload Size (MPS). The functions of the OAL and OMNI interface MTU/MRU/MPS considerations are specified in [I-D.templin-6man-omni]. (Note that the OMNI interface accommodates an assured MTU of 65535 octets due to the use of fragmentation, and can optionally expose larger MTUs to upper layers for best-effort Jumbogram services.)

When the network layer presents an original IP packet to the OMNI interface, the OAL source encapsulates and fragments the original IP packet if necessary. When the network layer presents the OMNI interface with multiple original IP packets bound to the same OAL destination, the OAL source can concatenate them as a single OAL super-packet as discussed in [I-D.templin-6man-omni] before applying fragmentation. The OAL source then encapsulates each OAL fragment in L2 headers for transmission as carrier packets over an underlay interface connected to either a physical link (e.g., Ethernet, WiFi, Cellular, etc.) or a virtual link such as an Internet or higher-layer tunnel (see the definition of link in [RFC8200]).

Note: Although a CRH-32 may be inserted or removed by a Gateway in the path (see: Section 3.10.4), this does not interfere with the destination's ability to reassemble since the CRH-32 is not included in the fragmentable part and its removal/transformation does not invalidate fragment header information.

3.10. OMNI Interface Forwarding Algorithm

Original IP packets enter a node's OMNI interface either from the network layer (i.e., from a local application or the IP forwarding system) while carrier packets enter from the link layer (i.e., from an OMNI interface neighbor). All original IP packets and carrier packets entering a node's OMNI interface first undergo data origin authentication as discussed in Section 3.8. Those that satisfy data origin authentication are processed further, while all others are dropped silently.

Original IP packets that enter the OMNI interface from the network layer are forwarded to an OMNI interface neighbor using OAL encapsulation and fragmentation to produce carrier packets for transmission over underlay interfaces. (If routing indicates that the original IP packet should instead be forwarded back to the

network layer, the packet is dropped to avoid looping). Carrier packets that enter the OMNI interface from the link layer are either re-encapsulated and re-admitted into the OMNI link, or reassembled and forwarded to the network layer where they are subject to either local delivery or IP forwarding. In all cases, the OAL MUST NOT decrement the original IP packet TTL/Hop-count since its forwarding actions occur below the network layer.

OMNI interfaces may have multiple underlay interfaces and/or neighbor cache entries for neighbors with multiple underlay interfaces (see Section 3.3). The OAL uses Interface Attributes and/or Traffic Selectors (e.g., port numbers, flow specifications, etc.) to select an outbound underlay interface for each OAL packet and also to select segment routing and/or link-layer destination addresses based on the neighbor's underlay interfaces. AERO implementations SHOULD permit network management to dynamically adjust Traffic Selector values at runtime.

If an OAL packet matches the Traffic Selectors of multiple outgoing interfaces and/or neighbor interfaces, the OMNI interface replicates the packet and sends one copy via each of the (outgoing / neighbor) interface pairs; otherwise, it sends a single copy of the OAL packet via an interface with the best matching Traffic Selector. (While not strictly required, the likelihood of successful reassembly may improve when the OMNI interface sends all fragments of the same fragmented OAL packet consecutively over the same underlay interface pair to avoid complicating factors such as delay variance and reordering.) AERO nodes keep track of which underlay interfaces are currently "reachable" or "unreachable", and only use "reachable" interfaces for forwarding purposes.

The Subnet ID value in ULAs is used only for subnet coordination within a local OMNI link segment. When a node forwards a packet with a ULA with a foreign Global and/or Subnet ID value it forwards the packet based solely on the OMNI link routing information. For this reason, OMNI link routing and forwarding table entries always include both ULAs with their associated prefix lengths and XLA-MNPs which encode an MNP while leaving the Global and Subnet ID values set to 0.

The following sections discuss the OMNI interface forwarding algorithms for Hosts, Clients, Proxy/Servers and Gateways. In the following discussion, an original IP packet's destination address is said to "match" if it is the same as a cached address, or if it is covered by a cached prefix (which may be encoded in a *LA-MNP).

3.10.1. Host Forwarding Algorithm

When an original IP packet enters a Host's OMNI interface from the network layer the Host searches for a NCE that matches the destination. If there is a matching NCE, the Host performs L2 encapsulation, fragments the encapsulated packet if necessary and forwards the packets into the ENET addressed to the L2 address of the neighbor.

After sending a packet, the Host may receive a Redirect message from its upstream Client to inform it of another AERO node on the same ENET that would provide a better first hop. The Host authenticates the Redirect message, then updates its neighbor cache accordingly.

3.10.2. Client Forwarding Algorithm

When an original IP packet enters a Client's OMNI interface from the network layer the Client searches for a NCE that matches the destination. If there is a matching NCE on an ANET/INET interface (i.e., an upstream interface), the Client selects one or more "reachable" neighbor interfaces in the entry for forwarding purposes. Otherwise, the Client invokes route optimization per Section 3.13 and follows the multilink forwarding procedures outlined there. If there is a matching NCE on an ENET interface (i.e., a downstream interface), the Client instead performs OAL and/or L2 encapsulation and forwards the packet to the downstream Host or Client.

When a carrier packet enters a Client's OMNI interface from the link-layer, if the OAL destination matches one of the Client's *LAs the Client (acting as an OAL destination) verifies that the Identification is in-window for this OAL source, then reassembles and decapsulates as necessary and delivers the original IP packet to the network layer. If the OAL destination matches a NCE for a Host or Client on an ENET interface, the Client instead forwards the carrier packet to the Host/Client. If the OAL destination does not match, the Client drops the original IP packet and MAY return a network-layer ICMP Destination Unreachable message subject to rate limiting (see: Section 3.11).

When a Client forwards a carrier packet from an ENET Host to a neighbor connected to the same ENET, it also returns a Redirect message to inform the source that it can reach the neighbor directly as an ENET peer.

Note: Clients and their FHS Proxy/Server (and other Client) peers can exchange original IP packets over ANET underlay interfaces without invoking the OAL, since the ANET is secured at the link and physical layers. By forwarding original IP packets without invoking the OAL,

however, the ANET peers can engage only in classical path MTU discovery since the packets are subject to loss and/or corruption due to the various per-link MTU limitations that may occur within the ANET. Moreover, the original IP packets do not include either the OAL integrity check or per-packet Identification values that can be used for data origin authentication and link-layer retransmissions. The tradeoff therefore involves an assessment of the per-packet encapsulation overhead saved by bypassing the OAL vs. inheritance of classical network "brittleness". (Note however that ANET peers can send small original IP packets without invoking the OAL, while invoking the OAL for larger packets. This presents the beneficial aspects of both small packet efficiency and large packet robustness, with delay variance and reordering as possible side effects.)

Note: The forwarding table entries established in peer Clients of a multihop forwarding region are based on ULA-MNPs and/or TLAs used to seed the multihop routing protocols. When ULA-MNPs are used, the ULA /64 prefix provides topological relevance for the multihop forwarding region, while the 64-bit Interface Identifier encodes the Client MNP. Therefore, Clients can forward atomic fragments with compressed OAL headers that do not include ULA or MFVI information by examining the MNP-based addresses in the actual IP packet header. In other words, each forwarding table entry contains two pieces of forwarding information - the ULA information in the prefix and the MNP information in the interface identifier.

3.10.3. Proxy/Server and Relay Forwarding Algorithm

When a Proxy/Server receives an original IP packet from the network layer, it drops the packet if routing indicates that it should be forwarded back to the network layer to avoid looping. Otherwise, the Proxy/Server regards the original IP packet the same as if it had arrived as carrier packets with OAL destination set to its own ULA. When the Proxy/Server receives carrier packets on underlay interfaces with OAL destination set to its own ULA, it performs OAL reassembly if necessary to obtain the original IP packet. The Proxy/Server then supports multilink forwarding procedures as specified in Section 3.13.2 and/or acts as an ROS to initiate route optimization as specified in Section 3.13.

When the Proxy/Server receives a carrier packet with OAL destination set to a *LA-MNP that does not match the MSP, it accepts the carrier packet only if data origin authentication succeeds and if there is a network layer routing table entry for a GUA route that matches the *LA-MNP. If there is no route, the Proxy/Server drops the carrier packet; otherwise, it reassembles and decapsulates to obtain the original IP packet then acts as a Relay to present it to the network layer where it will be delivered according to standard IP forwarding.

When a Proxy/Server receives a carrier packet from one of its Client neighbors with OAL destination set to another node, it forwards the packets via a matching NCE or via the spanning tree if there is no matching entry. When the Proxy/Server receives a carrier packet with OAL destination set to a *LA-MNP of one of its Client neighbors established through RS/RA exchanges, it accepts the carrier packet only if data origin authentication succeeds. If the NCE state is DEPARTED, the Proxy/Server changes the OAL destination address to the ULA of the new Proxy/Server, then re-encapsulates the carrier packet and forwards it to a Gateway which will eventually deliver it to the new Proxy/Server. If the neighbor cache state for the Client is REACHABLE, the Proxy/Server forwards the carrier packets to the Client which then must reassemble. (Note that the Proxy/Server does not reassemble carrier packets not explicitly addressed to its own ULA, since some of the carrier packets of the same original IP packet could be forwarded through a different Proxy/Server.) In that case, the Client may receive fragments that are smaller than its link MTU but that can still be reassembled.

Proxy/Servers process carrier packets with OAL destinations that do not match their ULA in the same manner as for traditional IP forwarding within the OAL, i.e., nodes use IP forwarding to forward packets not explicitly addressed to themselves. (Proxy/Servers include a special case that accepts and reassembles carrier packets destined to a *LA-MNP of one of their Clients received over the secured spanning tree.) Proxy/Servers process carrier packets with their ULA as the destination by first examining the packet for a CRH-32 header or an OCH header. In that case, the Proxy/Server examines the next MFVI in the carrier packet to locate the MFV entry in the MFIB for next hop forwarding (i.e., without examining IP addresses). When the Proxy/Server forwards the carrier packet, it changes the destination address according to the MFVI value for the next hop found either in the CRH-32 header or in the node's own MFIB. Proxy/Servers must verify that the L2 addresses of carrier packets not received from the secured spanning tree are "trusted" before forwarding according to an MFV (otherwise, the carrier packet must be dropped).

Note: Proxy/Servers may receive carrier packets addressed to their own ULA with CRH-32s that include additional forwarding information. Proxy/Servers use the forwarding information to determine the correct NCE and underlay interface for forwarding to the target Client, then remove the CRH-32 and forward the carrier packet. If necessary, the Proxy/Server reassembles first before re-encapsulating (and possibly also re-fragmenting) then forwards to the target Client.

Note: Clients and their FHS Proxy/Server peers can exchange original IP packets over ANET underlay interfaces without invoking the OAL, since the ANET is secured at the link and physical layers. By forwarding original IP packets without invoking the OAL, however, the Client and Proxy/Server can engage only in classical path MTU discovery since the packets are subject to loss and/or corruption due to the various per-link MTU limitations that may occur within the ANET. Moreover, the original IP packets do not include either the OAL integrity check or per-packet Identification values that can be used for data origin authentication and link-layer retransmissions. The tradeoff therefore involves an assessment of the per-packet encapsulation overhead saved by bypassing the OAL vs. inheritance of classical network "brittleness". (Note however that ANET peers can send small original IP packets without invoking the OAL, while invoking the OAL for larger packets. This presents the beneficial aspects of both small packet efficiency and large packet robustness.)

Note: When a Proxy/Server receives a (non-OAL) original IP packet from an ANET Client, or a carrier packet with OAL destination set to its own ULA from any Client, the Proxy/Server reassembles if necessary then performs ROS functions on behalf of the Client. The Client may at some later time begin sending carrier packets to the OAL address of the actual target instead of the Proxy/Server, at which point it may begin functioning as an ROS on its own behalf and thereby "override" the Proxy/Server's ROS role.

Note; Proxy/Servers drop any original IP packets (received either directly from an ANET Client or following reassembly of carrier packets received from an ANET/INET Client) with a destination that corresponds to the Client's delegated MNP. Similarly, Proxy/Servers drop any carrier packet received with both a source and destination that correspond to the Client's delegated MNP regardless of their OMNI link point of origin. These checks are necessary to prevent Clients from either accidentally or intentionally establishing endless loops that could congest Proxy/Servers and/or ANET/INET links.

Note: Proxy/Servers forward secure control plane carrier packets via the SRT secured spanning tree and forward other carrier packets via the unsecured spanning tree. When a Proxy/Server receives a carrier packet from the secured spanning tree, it considers the message as authentic without having to verify upper layer authentication signatures. When a Proxy/Server receives a carrier packet from the unsecured spanning tree, it applies data origin authentication itself and/or forwards the unsecured message toward the destination which must apply data origin authentication on its own behalf.

Note: If the Proxy/Server has multiple original IP packets to send to the same neighbor, it can concatenate them in a single OAL super-packet [I-D.templin-6man-omni].

3.10.4. Gateway Forwarding Algorithm

Gateways forward spanning tree carrier packets while decrementing the OAL header Hop Count but not the original IP header Hop Count/TTL. Gateways convey carrier packets that encapsulate critical IPv6 ND control messages or routing protocol control messages via the SRT secured spanning tree, and may convey other carrier packets via the secured/unsecured spanning tree or via more direct paths according to MFIB information. When the Gateway receives a carrier packet, it removes the L2 headers and searches for an MFIB entry that matches an MFVI or an IP forwarding table entry that matches the OAL destination address.

Gateways process carrier packets with OAL destinations that do not match their ULA or the SRT Subnet Router Anycast address in the same manner as for traditional IP forwarding within the OAL, i.e., nodes use IP forwarding to forward packets not explicitly addressed to themselves. Gateways process carrier packets with their ULA or the SRT Subnet Router Anycast address as the destination by first examining the packet for a full OAL header with a CRH-32 extension or an OCH header. In that case, the Gateway examines the next MFVI in the carrier packet to locate the MFV entry in the MFIB for next hop forwarding (i.e., without examining IP addresses). When the Gateway forwards the carrier packet, it changes the destination address according to the MFVI value for the next hop found either in the CRH-32 header or in the node's own MFIB. If the Gateway has a NCE for the target Client with an entry for the target underlay interface and current L2 addresses, the Gateway instead forwards directly to the target Client while using the final hop MFVI instead of the next hop (see: Section 3.13.4).

Gateways forward carrier packets received from a first segment via the secured spanning tree to the next segment also via the secured spanning tree. Gateways forward carrier packets received from a first segment via the unsecured spanning tree to the next segment also via the unsecured spanning tree. Gateways use a single IPv6 routing table that always determines the same next hop for a given OAL destination, where the secured/unsecured spanning tree is determined through the selection of the underlay interface to be used for transmission (i.e., a secured tunnel or an open INET interface).

As for Proxy/Servers, Gateways must verify that the L2 addresses of carrier packets not received from the secured spanning tree are "trusted" before forwarding according to an MFV (otherwise, the carrier packet must be dropped).

3.11. OMNI Interface Error Handling

When an AERO node admits an original IP packet into the OMNI interface, it may receive link-layer or network-layer error indications. The AERO node may also receive OMNI link error indications in OAL-encapsulated uNA messages that include authentication signatures.

A link-layer error indication is an ICMP error message generated by a router in an underlay network on the path to the neighbor or by the neighbor itself. The message includes an IP header with the address of the node that generated the error as the source address and with the link-layer address of the AERO node as the destination address.

The IP header is followed by an ICMP header that includes an error Type, Code and Checksum. Valid type values include "Destination Unreachable", "Time Exceeded" and "Parameter Problem" [RFC0792][RFC4443]. (OMNI interfaces ignore link-layer IPv4 "Fragmentation Needed" and IPv6 "Packet Too Big" messages for carrier packets that are no larger than the minimum/path MPS as discussed in Section 3.9, however these messages may provide useful hints of probe failures during path MPS probing.)

The ICMP header is followed by the leading portion of the carrier packet that generated the error, also known as the "packet-in-error". For ICMPv6, [RFC4443] specifies that the packet-in-error includes: "As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU" (i.e., no more than 1280 bytes). For ICMPv4, [RFC0792] specifies that the packet-in-error includes: "Internet Header + 64 bits of Original Data Datagram", however [RFC1812] Section 4.3.2.3 updates this specification by stating: "the ICMP datagram SHOULD contain as much of the original datagram as possible without the length of the ICMP datagram exceeding 576 bytes".

The link-layer error message format is shown in Figure 4:

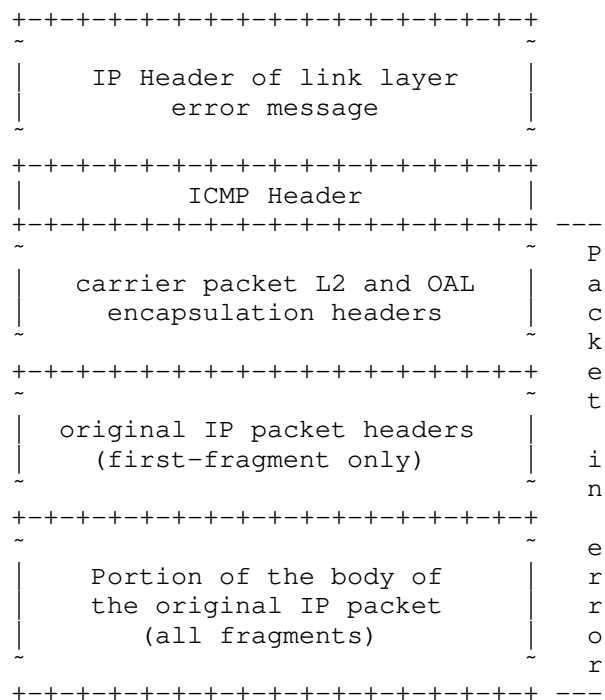


Figure 4: OMNI Interface Link-Layer Error Message Format

The AERO node rules for processing these link-layer error messages are as follows:

- * When an AERO node receives a link-layer Parameter Problem message, it processes the message the same as described as for ordinary ICMP errors in the normative references [RFC0792][RFC4443].
- * When an AERO node receives persistent link-layer Time Exceeded messages, the IP ID field may be wrapping before earlier fragments awaiting reassembly have been processed. In that case, the node should begin including integrity checks and/or institute rate limits for subsequent packets.

- * When an AERO node receives persistent link-layer Destination Unreachable messages in response to carrier packets that it sends to one of its neighbor correspondents, the node should process the message as an indication that a path may be failing, and optionally initiate NUD over that path. If it receives Destination Unreachable messages over multiple paths, the node should allow future carrier packets destined to the correspondent to flow through a default route and re-initiate route optimization.
- * When an AERO Client receives persistent link-layer Destination Unreachable messages in response to carrier packets that it sends to one of its neighbor Proxy/Servers, the Client should mark the path as unusable and use another path. If it receives Destination Unreachable messages on many or all paths, the Client should associate with a new Proxy/Server and release its association with the old Proxy/Server as specified in Section 3.15.5.
- * When an AERO Proxy/Server receives persistent link-layer Destination Unreachable messages in response to carrier packets that it sends to one of its neighbor Clients, the Proxy/Server should mark the underlay path as unusable and use another underlay path.
- * When an AERO Proxy/Server receives link-layer Destination Unreachable messages in response to a carrier packet that it sends to one of its permanent neighbors, it treats the messages as an indication that the path to the neighbor may be failing. However, the dynamic routing protocol should soon reconverge and correct the temporary outage.

When an AERO Gateway receives a carrier packet for which the network-layer destination address is covered by an MSP assigned to a black-hole route, the Gateway drops the packet if there is no more-specific routing information for the destination and returns an OMNI interface Destination Unreachable message subject to rate limiting.

When an AERO node receives a carrier packet for which reassembly is currently congested, it returns an OMNI interface Packet Too Big (PTB) message as discussed in [I-D.templin-6man-omni] (note that the PTB messages could indicate either "hard" or "soft" errors).

AERO nodes include ICMPv6 error messages intended for an OAL source as sub-options in the OMNI option of secured uNA messages. When the OAL source receives the uNA message, it can extract the ICMPv6 error message enclosed in the OMNI option and either process it locally or translate it into a network-layer error to return to the original source.

3.12. AERO Mobility Service Coordination

AERO nodes observe the Router Discovery and Prefix Registration specifications found in Section 15 of [I-D.templin-6man-omni]. AERO nodes further coordinate their autoconfiguration actions with the mobility service as discussed in the following sections.

3.12.1. AERO Service Model

Each AERO Proxy/Server on the OMNI link is configured to facilitate Client prefix delegation/registration requests. Each Proxy/Server is provisioned with a database of MNP-to-Client ID mappings for all Clients enrolled in the AERO service, as well as any information necessary to authenticate each Client. The Client database is maintained by a central administrative authority for the OMNI link and securely distributed to all Proxy/Servers, e.g., via the Lightweight Directory Access Protocol (LDAP) [RFC4511], via static configuration, etc. Clients receive the same service regardless of the Proxy/Servers they select.

Clients associate each of their ANET/INET underlay interfaces with a FHS Proxy/Server. Each FHS Proxy/Server locally services one or more of the Client's underlay interfaces, and the Client typically selects one among them to serve as the Hub Proxy/Server (the Client may instead select a "third-party" Hub Proxy/Server that does not directly service any of its underlay interfaces). All of the Client's other FHS Proxy/Servers forward proxied copies of RS/RA messages between the Hub Proxy/Server and Client without assuming the Hub role functions themselves.

Each Client associates with a single Hub Proxy/Server at a time, while all other Proxy/Servers are candidates for providing the Hub role for other Clients. An FHS Proxy/Server assumes the Hub role when it receives an RS message with its own ULA or link-scoped All-Routers multicast as the destination. An FHS Proxy/Server assumes the proxy role when it receives an RS message with the ULA of another Proxy/Server as the destination. (An FHS Proxy/Server can also assume the proxy role when it receives an RS message addressed to link-scoped All-Routers multicast if it can determine the ULA of another Proxy/Server to serve as a Hub.)

Hosts and Clients on ENET interfaces associate with an upstream Client on the ENET the same as a Client would associate with an ANET Proxy/Server. In particular, the Host/Client sends an RS message via the ENET which directs the message to the upstream Client. The upstream Client returns an RA message. In this way, the downstream nodes see the ENET as an ANET and see the upstream Client as a Proxy/Server for that ANET.

AERO Hosts, Clients and Proxy/Servers use IPv6 ND messages to maintain neighbor cache entries. AERO Proxy/Servers configure their OMNI interfaces as advertising NBMA interfaces, and therefore send unicast RA messages with a short Router Lifetime value (e.g., ReachableTime seconds) in response to a Client's RS message. Thereafter, Clients send additional RS messages to keep Proxy/Server state alive.

AERO Clients and Hub Proxy/Servers include prefix delegation and/or registration parameters in RS/RA messages. The IPv6 ND messages are exchanged between the Client and Hub Proxy/Server (via any FHS Proxy/Servers acting as proxys) according to the prefix management schedule required by the service. If the Client knows its MNP in advance, it can employ prefix registration by including its XLA-MNP as the source address of an RS message and with an OMNI option with valid prefix registration information for the MNP. If the Hub Proxy/Server accepts the Client's MNP assertion, it injects the MNP into the routing system and establishes the necessary neighbor cache state. If the Client does not have a pre-assigned MNP, it can instead employ prefix delegation by including a TLA as the source address of an RS message and with an OMNI option with prefix delegation parameters to request an MNP.

The following sections outlines Host, Client and Proxy/Server behaviors based on the Router Discovery and Prefix Registration specifications found in Section 15 of [I-D.templin-6man-omni]. These sections observe all of the OMNI specifications, and include additional specifications of the interactions of Client-Proxy/Server RS/RA exchanges with the AERO mobility service.

3.12.2. AERO Host and Client Behavior

AERO Hosts and Clients discover the addresses of candidate Proxy/Servers by resolving the Potential Router List (PRL) in a similar manner as described in [RFC5214]. Discovery methods include static configuration (e.g., a flat-file map of Proxy/Server addresses and locations), or through an automated means such as Domain Name System (DNS) name resolution [RFC1035]. Alternatively, the Host/Client can discover Proxy/Server addresses through a layer 2 data link login exchange, or through an RA response to a multicast/anycast RS as described below. In the absence of other information, the Host/Client can resolve the DNS Fully-Qualified Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the OMNI link (e.g., "example.com"). The name resolution returns a set of resource records with Proxy/Server address information.

The Host/Client then performs RS/RA exchanges over each of its underlay interfaces to associate with (possibly multiple) FHS Proxy/Serves and a single Hub Proxy/Server as specified in Section 15 of [I-D.templin-6man-omni]. The Host/Client then sends each RS (either directly via Direct interfaces, via a VPN for VPned interfaces, via an access router for ANET interfaces or via INET encapsulation for INET interfaces) and waits up to RetransTimer milliseconds for an RA message reply (see Section 3.12.3) while retrying up to MAX_RTR_SOLICITATIONS if necessary. If the Host/Client receives no RAs, or if it receives an RA with Router Lifetime set to 0, the Client SHOULD abandon attempts through the first candidate Proxy/Server and try another Proxy/Server.

After the Host/Client registers its underlay interfaces, it may wish to change one or more registrations, e.g., if an interface changes address or becomes unavailable, if traffic selectors change, etc. To do so, the Host/Client prepares an RS message to send over any available underlay interface as above. The RS includes an OMNI option with prefix registration/delegation information and with an Interface Attributes sub-option specific to the selected underlay interface. When the Host/Client receives the Hub Proxy/Server's RA response, it has assurance that both the Hub and FHS Proxy/Servers have been updated with the new information.

If the Host/Client wishes to discontinue use of a Hub Proxy/Server it issues an RS message over any underlay interface with an OMNI option with a prefix release indication (i.e., by setting the OMNI Neighbor Coordination header Preflen to 0). When the Hub Proxy/Server processes the message, it releases the MNP, sets the NCE state for the Host/Client to DEPARTED and returns an RA reply with Router Lifetime set to 0. After a short delay (e.g., 2 seconds), the Hub Proxy/Server withdraws the MNP from the routing system. (Alternatively, when the Host/Client associates with a new FHS/Hub Proxy/Server it can include an OMNI "Proxy/Server Departure" sub-option in RS messages with the ULAs of the Old FHS/Hub Proxy/Servers.)

3.12.3. AERO Proxy/Server Behavior

AERO Proxy/Servers act as both IP routers and IPv6 ND proxys, and support a prefix delegation/registration service for Clients. Proxy/Servers arrange to add their ULAs to the PRL maintained in a static map of Proxy/Server addresses for the link, the DNS resource records for the FQDN "linkupnetworks.[domainname]", etc. before entering service. The PRL should be arranged such that Clients can discover the addresses of Proxy/Servers that are geographically and/or topologically "close" to their underlay network connections.

When a FHS/Hub Proxy/Server receives a prospective Client's RS message, it SHOULD return an immediate RA reply with Router Lifetime set to 0 if it is currently too busy or otherwise unable to service the Client; otherwise, it processes the RS as specified in Section 15 of [I-D.templin-6man-omni]. When the Hub Proxy/Server receives the RS, it determines the correct MNPs to provide to the Client by processing the XLA-MNP prefix parameters and/or the DHCPv6 OMNI sub-option. When the Hub Proxy/Server returns the MNPs, it also creates an XLA-MNP forwarding table entry for the MNP resulting in a BGP update (see: Section 3.2.3). The Hub Proxy/Server then returns an RA to the Client with destination set to the source of the RS (if an FHS Proxy/Server on the return path proxys the RA, it changes the destination to the Client's ULA-MNP).

After the initial RS/RA exchange, the Hub Proxy/Server maintains a ReachableTime timer for each of the Client's underlay interfaces individually (and for the Client's NCE collectively) set to expire after ReachableTime seconds. If the Client (or an FHS Proxy/Server) issues additional RS messages, the Hub Proxy/Server sends an RA response and resets ReachableTime. If the Hub Proxy/Server receives an IPv6 ND message with a prefix release indication it sets the Client's NCE to the DEPARTED state and withdraws the MNP route from the routing system after a short delay (e.g., 2 seconds). If ReachableTime expires before a new RS is received on an individual underlay interface, the Hub Proxy/Server marks the interface as DOWN. If ReachableTime expires before any new RS is received on any individual underlay interface, the Hub Proxy/Server sets the NCE state to STALE and sets a 10 second timer. If the Hub Proxy/Server has not received a new RS or uNA message with a prefix release indication before the 10 second timer expires, it deletes the NCE and withdraws the XLA-MNP from the routing system.

The Hub Proxy/Server processes any IPv6 ND messages pertaining to the Client while forwarding to the Client or responding on the Client's behalf as necessary. The Hub Proxy/Server may also issue unsolicited RA messages, e.g., with reconfigure parameters to cause the Client to renegotiate its prefix delegation/registrations, with Router Lifetime set to 0 if it can no longer service this Client, etc. The Hub Proxy/Server may also receive carrier packets via the secured spanning tree that contain initial data packets sent while route optimization is in progress. The Hub Proxy/Server reassembles, then re-encapsulates/re-fragments and forwards the packets to the target Client via an FHS Proxy/Server if necessary. Finally, If the NCE is in the DEPARTED state, the old Hub Proxy/Server forwards any carrier packets it receives from the secured spanning tree and destined to the Client to the new Hub Proxy/Server, then deletes the entry after DepartTime expires.

Note: Clients SHOULD arrange to notify former Hub Proxy/Servers of their departures, but Hub Proxy/Servers are responsible for expiring neighbor cache entries and withdrawing XLA-MNP routes even if no departure notification is received (e.g., if the Client leaves the network unexpectedly). Hub Proxy/Servers SHOULD therefore set Router Lifetime to ReachableTime seconds in solicited RA messages to minimize persistent stale cache information in the absence of Client departure notifications. A short Router Lifetime also ensures that proactive RS/RA messaging between Clients and FHS Proxy/Servers will keep any NAT state alive (see above).

Note: All Proxy/Servers on an OMNI link MUST advertise consistent values in the RA Cur Hop Limit, M and O flags, Reachable Time and Retrans Timer fields the same as for any link, since unpredictable behavior could result if different Proxy/Servers on the same link advertised different values.

3.12.3.1. Additional Proxy/Server Considerations

AERO Clients register with FHS Proxy/Servers for each underlay interface. Each of the Client's FHS Proxy/Servers must inform a single Hub Proxy/Server of the Client's underlay interface(s) that it services. For Clients on Direct and VPned underlay interfaces, the FHS Proxy/Server for each interface is directly connected, for Clients on ANET underlay interfaces the FHS Proxy/Server is located on the ANET/INET boundary, and for Clients on INET underlay interfaces the FHS Proxy/Server is located somewhere in the connected Internetwork. When FHS Proxy/Server "B" processes a Client registration, it must either assume the Hub role or forward a proxied registration to another Proxy/Server "A" acting as the Hub. Proxy/Servers satisfy these requirements as follows:

- * when FHS Proxy/Server "B" receives a Client RS message, it first verifies that the OAL Identification is within the window for the NCE that matches the *LA-MNP for this Client neighbor and authenticates the message. If no NCE was found, Proxy/Server "B" instead creates one in the STALE state and caches the Client-supplied Interface Attributes, Origin Indication and OMNI Neighbor Coordination header window synchronization parameters as well as the Client's observed L2 addresses (noting that they may differ from the Origin addresses if there were NATs on the path). Proxy/Server "B" then examines the network-layer destination address. If the destination address is the ULA of a different Proxy/Server "A", Proxy/Server "B" prepares a separate proxied version of the RS message with an OAL header with source set to its own ULA and destination set to Proxy/Server B's ULA. Proxy/Server "B" also writes its own information over the Interface Attributes sub-option supplied by the Client, omits or zeros the Origin Indication sub-option then forwards the message into the OMNI link secured spanning tree.
- * when Hub Proxy/Server "A" receives the RS, it assume the Hub role and creates/updates a NCE for the Client with FHS Proxy/Server "B"'s Interface Attributes as the link-layer address information for this FHS omIndex. Hub Proxy/Server "A" then prepares an RA message with source set to its own ULA and destination set to the source of the RS message, then encapsulates the RA in an OAL header with source set to its own ULA and destination set to the ULA of FHS Proxy/Server "B". Hub Proxy/Server "A" then performs fragmentation if necessary and sends the resulting carrier packets into the secured spanning tree.
- * when FHS Proxy/Server "B" reassembles the RA, it locates the Client NCE based on the RA destination. If the RA message includes an OMNI "Proxy/Server Departure" sub-option, Proxy/Server "B" first sends a uNA to the old FHS/Hub Proxy/Servers named in the sub-option. Proxy/Server "B" then changes the RA destination address to the ULA-MNP of the Client, then re-encapsulates the message with OAL source set to its own ULA and OAL destination set to ULA that appeared in the Client's RS source, with an appropriate Identification value, with an authentication signature if necessary, with the Client's Interface Attributes sub-option echoed and with the cached observed L2 addresses written into an Origin Indication sub-option. Proxy/Server "B" sets the P flag in the RA flags field to indicate that the message has passed through a proxy [RFC4389], includes responsive window synchronization parameters, then fragments the RA if necessary and returns the fragments to the Client.

- * The Client repeats this process over each of its additional underlay interfaces while treating each additional FHS Proxy/Server "C", "D", "E", etc. as a proxy to facilitate RS/RA exchanges between the Hub and the Client. The Client creates/updates NCEs for each such FHS Proxy/Server as well as the Hub Proxy/Server in the process.

After the initial RS/RA exchanges each FHS Proxy/Server forwards any of the Client's carrier packets with OAL destinations for which there is no matching NCE to a Gateway using OAL encapsulation with its own ULA as the source and with destination determined by the Client. The Proxy/Server instead forwards any carrier packets destined to a neighbor cache target directly to the target according to the OAL/link-layer information - the process of establishing neighbor cache entries is specified in Section 3.13.

While the Client is still associated with FHS Proxy/Servers "B", "C", "D", etc., each FHS Proxy/Server can send NS, RS and/or unsolicited NA messages to update the neighbor cache entries of other AERO nodes on behalf of the Client based on changes in Interface Attributes, Traffic Selectors, etc. This allows for higher-frequency Proxy-initiated RS/RA messaging over well-connected INET infrastructure supplemented by lower-frequency Client-initiated RS/RA messaging over constrained ANET data links.

If the Hub Proxy/Server "A" ceases to send solicited RAs, FHS Proxy/Servers "B", "C", "D" can send unsolicited RAs over the Client's underlay interface with destination set to (link-local) All-Nodes multicast and with Router Lifetime set to zero to inform Clients that the Hub Proxy/Server has failed. Although FHS Proxy/Servers "B", "C" and "D" can engage in IPv6 ND exchanges on behalf of the Client, the Client can also send IPv6 ND messages on its own behalf, e.g., if it is in a better position to convey state changes. The IPv6 ND messages sent by the Client include the Client's XLA-MNP as the source in order to differentiate them from the IPv6 ND messages sent by a FHS Proxy/Server.

If the Client becomes unreachable over all underlay interface it serves, the Hub Proxy/Server sets the NCE state to DEPARTED and retains the entry for DepartTime seconds. While the state is DEPARTED, the Hub Proxy/Server forwards any carrier packets destined to the Client to a Gateway via OAL encapsulation. When DepartTime expires, the Hub Proxy/Server deletes the NCE, withdraws the XLA-MNP route and discards any further carrier packets destined to the former Client.

In some ANETs that employ a Proxy/Server, the Client's MNP can be injected into the ANET routing system. In that case, the Client can send original IP packets without invoking the OAL so that the ANET routing system transports the original IP packets to the Proxy/Server. This can be beneficial, e.g., if the Client connects to the ANET via low-end data links such as some aviation wireless links.

If the ANET first-hop access router is on the same underlay link as the Client and recognizes the AERO/OMNI protocol, the Client can avoid OAL encapsulation for both its control and data messages. When the Client connects to the link, it can send an unencapsulated RS message with source address set to its own XLA-MNP (or to a TLA), and with destination address set to the ULA of the Client's selected Proxy/Server or to link-scoped All-Routers multicast. The Client includes an OMNI option formatted as specified in [I-D.templin-6man-omni]. The Client then sends the unencapsulated RS message, which will be intercepted by the AERO-aware ANET access router.

The ANET access router then performs OAL encapsulation on the RS message and forwards it to a Proxy/Server at the ANET/INET boundary. When the access router and Proxy/Server are one and the same node, the Proxy/Server would share an underlay link with the Client but its message exchanges with outside correspondents would need to pass through a security gateway at the ANET/INET border. The method for deploying access routers and Proxys (i.e. as a single node or multiple nodes) is an ANET-local administrative consideration.

Note: When a Proxy/Server alters the IPv6 ND message contents before forwarding (e.g., such as altering the OMNI option contents), the original IPv6 ND message checksum or authentication signature is invalidated, and a new checksum or authentication signature must be calculated and included.

Note: When a Proxy/Server receives a secured Client NS message, it performs the same proxying procedures as for described for RS messages above. The proxying procedures for NS/NA message exchanges is specified in Section 3.13.

3.12.3.2. Detecting and Responding to Proxy/Server Failures

In environments where fast recovery from Proxy/Server failure is required, FHS Proxy/Servers SHOULD use proactive Neighbor Unreachability Detection (NUD) to track Hub Proxy/Server reachability in a similar fashion as for Bidirectional Forwarding Detection (BFD) [RFC5880]. Each FHS Proxy/Server can then quickly detect and react to failures so that cached information is re-established through alternate paths. The NS/NA(NUD) control messaging is carried only

over well-connected ground domain networks (i.e., and not low-end aeronautical radio links) and can therefore be tuned for rapid response.

FHS Proxy/Servers perform continuous NS/NA(NUD) exchanges with the Hub Proxy/Server, e.g., one exchange per second. The FHS Proxy/Server sends the NS(NUD) message via the spanning tree with its own ULA as the source and the ULA of the Hub Proxy/Server as the destination, and the Hub Proxy/Server responds with an NA(NUD). When the FHS Proxy/Server is also sending RS messages to a Hub Proxy/Server on behalf of Clients, the resulting RA responses can be considered as equivalent hints of forward progress. This means that the FHS Proxy/Server need not also send a periodic NS(NUD) if it has already sent an RS within the same period. If the Hub Proxy/Server fails (i.e., if the FHS Proxy/Server ceases to receive advertisements), the FHS Proxy/Server can quickly inform Clients by sending unsolicited RA messages

The FHS Proxy/Server sends unsolicited RA messages with source address set to the Hub Proxy/Server's address, destination address set to (link-local) All-Nodes multicast, and Router Lifetime set to 0. The FHS Proxy/Server SHOULD send MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by small delays [RFC4861]. Any Clients that had been using the failed Hub Proxy/Server will receive the RA messages and select one of its other FHS Proxy/Servers to assume the Hub role (i.e., by sending an RS with destination set to the ULA of the new Hub).

3.12.3.3. DHCPv6-Based Prefix Registration

When a Client is not pre-provisioned with an MNP, it will need for the Hub Proxy/Server to select one or more MNPs on its behalf and set up the correct state in the AERO routing service. (A Client with a pre-provisioned MNP may also request the Hub Proxy/Server to select additional MNPs.) The DHCPv6 service [RFC8415] is used to support this requirement.

When a Client needs to have the Hub Proxy/Server select MNPs, it sends an RS message with source address set to a TLA and with an OMNI option that includes a DHCPv6 message sub-option with DHCPv6 Prefix Delegation (DHCPv6-PD) parameters. When the Hub Proxy/Server receives the RS message, it extracts the DHCPv6-PD message from the OMNI option.

The Hub Proxy/Server then acts as a "Proxy DHCPv6 Client" in a message exchange with the locally-resident DHCPv6 server, which delegates MNPs and returns a DHCPv6-PD Reply message. (If the Hub Proxy/Server wishes to defer creation of MN state until the DHCPv6-PD

Reply is received, it can instead act as a Lightweight DHCPv6 Relay Agent per [RFC6221] by encapsulating the DHCPv6-PD message in a Relay-forward/reply exchange with Relay Message and Interface ID options.)

When the Hub Proxy/Server receives the DHCPv6-PD Reply, it creates an XLA based on the delegated MNP adds an XLA-MNP route to the routing system. The Hub Proxy/Server then sends an RA back to the Client either directly or via an FHS Proxy/Server acting as a proxy. The Proxy/Server that returns the RA directly to the Client sets the (newly-created) ULA-MNP as the destination address and with the DHCPv6-PD Reply message and OMNI Neighbor Coordination header Preflen coded in the OMNI option. When the Client receives the RA, it creates a default route, assigns the Subnet Router Anycast address and sets its {ULA,XLA}-MNP based on the delegated MNP.

Note: Further details of the DHCPv6-PD based MNP registration (as well as a minimal MNP delegation alternative that avoids including a DHCPv6 message sub-option in the RS) are found in [I-D.templin-6man-omni].

Note: when the Hub Proxy/Server forwards an RA to the Client via a different node acting as a FHS Proxy/Server, the Hub sets the RA destination to the same address that appeared in the RS source. The FHS Proxy/Server then subsequently sets the RA destination to the ULA-MNP when it forwards the Proxyed version of the RA to the Client - see [I-D.templin-6man-omni] for further details.

3.13. AERO Route Optimization

AERO nodes invoke route optimization when they need to forward initial packets to new target destinations over ANET/INET interfaces and for ongoing multilink forwarding for current destinations. Route optimization is based on IPv6 ND Address Resolution messaging between a Route Optimization Source (ROS) and a Relay or the target Client itself (reached via the current Hub Proxy/Server) acting as a Route Optimization Responder (ROR). Route optimization is initiated by the first eligible ROS closest to the source as follows:

- * For Clients on VPNed and Direct interfaces, the Client's FHS Proxy/Server is the ROS.
- * For Clients on ANET interfaces, either the Client or the FHS Proxy/Server may be the ROS.
- * For Clients on INET interfaces, the Client itself is the ROS.

- * For correspondent nodes on INET/ENET interfaces serviced by a Relay, the Relay is the ROS.
- * For Clients that engage the Hub Proxy/Server in "mobility anchor" mode, the Hub Proxy/Server is the ROS.
- * For peers within the same ANET/ENET, route optimization is through receipt of Redirect messages from a Proxy/Server.

The AERO routing system directs a route optimization request sent by the ROS to the ROR, which returns a route optimization reply which must include information that is current, consistent and authentic. The ROS is responsible for periodically refreshing the route optimization, and the ROR is responsible for quickly informing the ROS of any changes. Following address resolution, the ROS and ROR perform ongoing multilink route optimizations to maintain optimal forwarding profiles.

The route optimization procedures are specified in the following sections.

3.13.1. Multilink Address Resolution

When one or more original IP packets from a source node destined to a target node arrives, the ROS checks for a NCE with an XLA that matches the target destination. If there is a NCE in the REACHABLE state, the ROS invokes the OAL and forwards the resulting carrier packets according to the cached state then returns from processing. Otherwise, if there is no NCE the ROS creates one in the INCOMPLETE state.

The ROS next prepares an NS message for Address Resolution (NS(AR)) to send toward an ROR while including the original IP packet(s) as trailing data following the NS(AR) in an OAL super-packet [I-D.templin-6man-omni]. The resulting NS(AR) message must be sent securely, and includes:

- * the ULA of the ROS as the source address.
- * the XLA corresponding to the original IP packet's destination as the Target Address, e.g., for 2001:db8:1:2::10:2000 the Target Address is fd00::2001:db8:1:2.
- * the Solicited-Node multicast address [RFC4291] formed from the lower 24 bits of the original IP packet's destination as the destination address, e.g., for 2001:db8:1:2::10:2000 the NS(AR) destination address is ff02:0:0:0:0:1:ff10:2000.

The NS(AR) message also includes an OMNI option with an authentication sub-option if necessary and with OMNI extension header Preflen set to the prefix length associated with the NS(AR) source. The ROS also includes Interface Attributes and Traffic Selectors for all of the source Client's underlay interfaces, calculates the authentication signature or checksum, then selects an Identification value and submits the NS(AR) message for OAL encapsulation with OAL source set to its own ULA and OAL destination set to the XLA corresponding to the target and with window synchronization parameters. The ROS then inserts a fragment header, performs fragmentation and L2 encapsulation, then sends the resulting carrier packets into the SRT secured spanning tree without decrementing the network-layer TTL/Hop Limit field.

When the ROS is a Client, it must instead use the ULA of one of its FHS Proxy/Servers as the OAL destination. The ROS Client then fragments, performs L2 encapsulation and forwards the carrier packets to the FHS Proxy/Server. The FHS Proxy/Server then reassembles, verifies the NS(AR) authentication signature or checksum, changes the OAL source to its own ULA, changes the OAL destination to the XLA corresponding to the target, selects an appropriate Identification, then re-fragments and forwards the resulting carrier packets into the secured spanning tree on behalf of the Client.

Note: both the target Client and its Hub Proxy/Server include current and accurate information for the Client's multilink Interface Attributes profile. The Hub Proxy/Server can be trusted to provide an authoritative response on behalf of the Client should the need arise. While the Client has no such trust basis, any attempt by the Client to mount an attack by providing false Interface Attributes information would only result in black-holing of return traffic, i.e., the "attack" could only result in denial of service to the Client itself. Therefore, the Client's asserted Interface Attributes need not be validated by the Hub Proxy/Server.

3.13.1.1. Relaying the NS(AR) *NET Packet(s)

When the Gateway receives carrier packets containing the NS(AR), it discards the L2 headers and determines the next hop by consulting its standard IPv6 forwarding table for the OAL header destination address. The Gateway then decrements the OAL header Hop-Limit, then re-encapsulates and forwards the carrier packet(s) via the secured spanning tree the same as for any IPv6 router, where they may traverse multiple OMNI link segments. The final-hop Gateway will deliver the carrier packet via the secured spanning tree to the Hub Proxy/Server (or Relay) that services the target.

3.13.1.2. Processing and Responding to the NS(AR)

When the Hub Proxy/Server for the target receives the NS(AR) secured carrier packets with the XLA of the target as the OAL destination, it reassembles then forwards the message to the target Client (while including an authentication signature and encapsulation if necessary) or processes the NS(AR) locally if it is acting as a Relay/IP router or the Client's designated ROR. The Hub Proxy/Server processes the message as follows:

- * if the NS(AR) target matches a Client NCE in the DEPARTED state, the (old) Hub Proxy/Server re-encapsulates by setting the OAL destination address to the ULA of the Client's new Hub Proxy/Server. The old Hub Proxy/Server then re-fragments and re-encapsulates, then forwards the resulting carrier packets over the secured spanning tree.
- * If the NS(AR) target matches a Client NCE in the REACHABLE state, the Hub Proxy/Server notes whether the NS(AR) arrived from the secured spanning tree then sets the OAL destination address to the ULA-MNP of the Client or the ULA of the selected FHS Proxy/Server for the Client (i.e., if the Hub is not also the FHS Proxy/Server for the selected underlay interface). If the message arrived via the secured spanning tree the Hub Proxy/Server verifies the checksum; otherwise, it must verify the message authentication signature before forwarding. When the Hub Proxy/Server determines the underlay interface for the target Client, it then changes the OAL destination to the ULA of the target Client's FHS Proxy/Server, re-fragments and forwards the resulting carrier packets into the secured spanning tree. When the FHS Proxy/Server receives the carrier packets, it reassembles and verifies the checksum, then includes an authentication signature if necessary, changes the OAL source to its own ULA and destination to the ULA-MNP of the target Client, includes an Identification value within the current window, then re-fragments and forwards the resulting carrier packets to the target Client ROR. (Note that if the Hub and FHS Proxy/Server are one and the same the Hub itself will perform the FHS procedures.)
- * If the NS(AR) target matches one of its non-MNP routes, the Hub Proxy/Server serves as both a Relay and a ROR, since the Relay forwards IP packets toward the (fixed network) target at the network layer.

The ROR then creates a NCE for the NS(AR) source address if necessary, processes the window synchronization parameters, caches all Interface Attributes and Traffic Selector information, and prepares a (solicited) NA(AR) message to return to the ROS with the

source address set to its own ULA-MNP, the destination address set to the NS(AR) ULA source address and the Target Address set to the same value that appeared in the NS(AR) Target Address. The ROR includes an OMNI option with OMNI Neighbor Coordination header Preflen set to the prefix length associated with the NA(AR) source address.

The ROR then sets the NA(AR) message R flag to 1 (as a router) and S flag to 1 (as a response to a solicitation) and sets the O flag to 1 (as an authoritative responder). The ROR finally submits the NA(AR) for OAL encapsulation with source set to its own ULA and destination set to either the ULA corresponding to the NS(AR) source or the ULA of its FHS Proxy/Server, selects an appropriate Identification, and includes window synchronization parameters and authentication signature or checksum. The ROR then includes Interface Attributes and Traffic Selector sub-options for all of the target's underlay interfaces with current information for each interface, fragments and encapsulates each fragment in appropriate L2 headers, then forwards the resulting (L2-encapsulated) carrier packets to the FHS Proxy/Server.

When the FHS Proxy/Server receives the carrier packets, it reassembles if necessary and verifies the authentication signature or checksum. The FHS Proxy/Server then changes the OAL source address to its own ULA, changes the destination to the ULA or XLA corresponding to the NA(AR) destination, includes an appropriate Identification, then fragments and forwards the carrier packets into the secured spanning tree.

Note: If the Hub Proxy/Server is acting as the Client's ROR but not as a Relay/IP router (i.e., by virtue of receipt of an RS message with the A flag set), it prepares the NS(AR) with the R flag set to 0 but without setting the SYN flag in the OMNI Neighbor Coordination header window synchronization parameters. This informs the ROS that it must initiate multilink route optimization to synchronize with the Client either directly or via a FHS Proxy/Server (see: Section 3.13.2).

3.13.1.3. Relaying the NA(AR)

When the Gateway receives NA(AR) carrier packets, it discards the L2 headers and determines the next hop by consulting its standard IPv6 forwarding table for the OAL header destination address. The Gateway then decrements the OAL header Hop-Limit, re-encapsulates the carrier packet and forwards it via the SRT secured spanning tree, where it may traverse multiple OMNI link segments. The final-hop Gateway will deliver the carrier packet via the secured spanning tree to a Proxy/Server for the ROS.

3.13.1.4. Processing the NA(AR)

When the ROS receives the NA(AR) message, it first searches for a NCE that matches the NA(AR) target address. The ROS then processes the message the same as for standard IPv6 Address Resolution [RFC4861]. In the process, it caches all OMNI option information in the target NCE (including all Interface Attributes), and caches the NA(AR) XLA source address as the address of the target Client.

When the ROS is a Client, the SRT secured spanning tree will first deliver the solicited NA(AR) message to the FHS Proxy/Server, which re-encapsulates and forwards the message to the Client. If the Client is on a well-managed ANET, physical security and protected spectrum ensures security for the NA(AR) without needing an additional authentication signature; if the Client is on the open INET the Proxy/Server must instead include an authentication signature (while adjusting the OMNI option size, if necessary). The Proxy/Server uses its own ULA as the OAL source and the ULA-MNP of the Client as the OAL destination.

3.13.2. Multilink Route Optimization

Following address resolution, the ROS and ROR can assert multilink paths through underlay interface pairs serviced by the same source/destination ULAs by sending unicast NS/NA messages with Multilink Forwarding Parameters and OMNI Neighbor Coordination window synchronization parameters when necessary. The unicast NS/NA messages establish multilink forwarding state in intermediate nodes in the path between the ROS and ROR.

To support multilink route optimization, OMNI interfaces include an additional forwarding table termed the Multilink Forwarding Information Base (MFIB) that supports carrier packet forwarding based on OMNI neighbor underlay interface pairs. The MFIB contains Multilink Forwarding Vectors (MFVs) indexed by 4-octet values known as MFV Indexes (MFVIs).

OAL source, intermediate and destination nodes create MFVs/MFVIs when they process an NS message with a Multilink Forwarding Parameters sub-option with Job code '00' (Initialize; Build B) or a solicited NA with Job code '01' (Follow B; Build A) (see: [I-D.templin-6man-omni]). The OAL source of the NS (and OAL destination of the solicited NA) are considered to reside in the "First Hop Segment (FHS)", while the OAL destination of the NS (and OAL source of the solicited NA) are considered to reside in the "Last Hop Segment (LHS)".

When an OAL node processes an NS with Job code '00', it creates an MFV, records the NS source and destination ULAs and assigns a "B" MFVI. When the "B" MFVI is referenced, the MFV retains the ULAs in (dst,src) order the opposite of how they appeared in the original NS to support full header reconstruction. (If the NS message included a nested OAL encapsulation, the ULAs of both OAL headers are retained.)

When an OAL node processes a solicited NA with Job code '01', it locates the MFV created by the NS and assigns an "A" MFVI. When the "A" MFVI is referenced, the MFV retains the ULAs in (src,dst) order the same as they appeared in the original NS to support full header reconstruction. (If the NS message included a nested OAL encapsulation, the ULAs of both OAL headers are retained.)

OAL nodes generate random 32-bit values as candidate A/B MFVIs which must first be tested for local uniqueness. If a candidate MFVI is already in use (or if the value is 0), the OAL node repeats the random generation process until it obtains a unique non-zero value. (Since the number of MFVs in service at each OAL node is likely to be much smaller than 2^{32} , the process will generate a unique value after a small number of tries; also, an MFVI generated by a first OAL node is never tested for uniqueness on other OAL nodes, since the uniqueness property is node-local only.)

OAL nodes maintain A/B MFVIs as follows:

- * "B1" - a locally-unique MFVI maintained independently by each OAL node on the path from the FHS OAL source to the last OAL intermediate node before the LHS OAL destination. The OAL node generates and assigns a "B1" MFVI to a newly-created MFV when it processes an NS message with Job code '00'. When the OAL node receives future carrier packets that include this value, it can unambiguously locate the correct MFV and determine directionality without examining addresses.
- * "A1" - a locally unique MFVI maintained independently by each OAL node on the path from the LHS OAL source to the last OAL intermediate node before the FHS OAL destination. The OAL node generates and assigns an "A1" MFVI to the MFV that configures the corresponding "B1" MFVI when it processes a solicited NA message with Job code '01'. When the OAL node receives future carrier packets that include this value, it can unambiguously locate the correct MFV and determine directionality without examining addresses.

- * "A2" - the A1 MFVI of a remote OAL node discovered by an FHS OAL source or OAL intermediate node when it processes an NA message with Job code '01' that originated from an LHS OAL source. A2 values MUST NOT be tested for uniqueness within the OAL node's local context.
- * "B2" - the B1 MFVI of a remote OAL node discovered by an LHS OAL source or OAL intermediate node when it processes an NS message with Job code '00' that originated from an FHS OAL source. B2 values MUST NOT be tested for uniqueness within the OAL node's local context.

When an FHS OAL source has an original IP packet to send to an LHS OAL destination discovered via multilink address resolution, it first selects a source and target underlay interface pair. The OAL source uses its cached information for the target underlay interface as LHS information then prepares an NS message with an OMNI Multilink Forwarding Parameters sub-option with Job code '00' and with source set to its own ULA or XLA. If the LHS FMT-Forward and FMT-Mode bits are both clear, the OAL source sets the destination to the ULA of the LHS Proxy/Server; otherwise, it sets the destination to the XLA of the target Client. The OAL source then sets window synchronization information in the OMNI Neighbor Coordination header and creates a NCE for the selected destination ULA or XLA in the INCOMPLETE state. The OAL source next creates an MFV based on the NS source and destination ULAs, then generates a "B1" MFVI and assigns it to the MFV while also including it as the first B entry in the MFVI List. The OAL source then populates the NS Multilink Forwarding Parameters based on any FHS/LHS information it knows locally. OAL intermediate nodes on the path to the OAL destination may populate additional FHS/LHS information on a hop-by-hop basis.

If the OAL source is the FHS Proxy/Server, it then performs OAL encapsulation/fragmentation while setting the source to its own ULA and setting the destination to the FHS Subnet Router Anycast ULA determined by applying the FHS SRT prefix length to its ULA. The FHS Proxy/Server next examines the LHS FMT code. If FMT-Forward is clear and FMT-Mode is set, the FHS Proxy/Server checks for a NCE for the ULA of the LHS Proxy/Server. If there is no NCE, the FHS Proxy/Server creates one in the INCOMPLETE state. If a new NCE was created (or if the existing NCE requires fresh window synchronization), the FHS Proxy/Server then writes window synchronization parameters into the OMNI Multilink Forwarding Parameters Tunnel Window Synchronization fields. The FHS Proxy/Server then selects an appropriate Identification value and L2 headers and forwards the resulting carrier packets into the secured spanning tree which will deliver them to a Gateway interface that assigns the FHS Subnet Router Anycast ULA.

If the OAL source is the FHS Client, it instead includes an authentication signature if necessary, performs OAL encapsulation, sets the source to its own ULA-MNP, sets the destination to the ULA of the FHS Proxy/Server and selects an appropriate Identification value for the FHS Proxy/Server. If FHS FMT-Forward is set and LHS FMT-Forward is clear, the FHS Client creates/updates a NCE for the ULA of the LHS Proxy/Server as above and includes Tunnel Window Synchronization parameters. The FHS Client then fragments and encapsulates in appropriate L2 headers then forwards the carrier packets to the FHS Proxy/Server. When the FHS Proxy/Server receives the carrier packets, it verifies the Identification, reassembles/decapsulates to obtain the NS then verifies the authentication signature or checksum. The FHS Proxy/Server then creates an MFV (i.e., the same as the FHS Client had done) while assigning the current B entry in the MFVI List (i.e., the one included by the FHS Client) as the "B2" MFVI for this MFV. The FHS Proxy/Server next generates a new unique "B1" MFVI, then both assigns it to the MFV and writes it as the next B entry in the OMNI Multilink Forwarding Parameters MFVI List (while also writing any FHS Client and Proxy/Server addressing information). The FHS Proxy/Server then checks FHS/LHS FMT-Forward/Mode to determine whether to create a NCE for the LHS Proxy/Server ULA and include Tunnel Window Synchronization parameters the same as above. The FHS Proxy/Server then calculates the checksum, re-fragments while setting the OAL source address to its own ULA and destination address to the FHS Subnet Router Anycast ULA, and includes an Identification appropriate for the secured spanning tree. The FHS Proxy/Server finally includes appropriate L2 headers and forwards the carrier packets into the secured spanning tree the same as above.

Gateways in the spanning tree forward carrier packets not explicitly addressed to themselves, while forwarding those that arrived via the secured spanning tree to the next hop also via the secured spanning tree and forwarding all others via the unsecured spanning tree. When an FHS Gateway receives a carrier packet over the secured spanning tree addressed to its ULA or the FHS Subnet Router Anycast ULA, it instead reassembles/decapsulates to obtain the NS then verifies the checksum. The FHS Gateway next creates an MFV (i.e., the same as the FHS Proxy/Server had done) while assigning the current B entry in the MFVI List as the MFV "B2" index. The FHS Gateway also caches the NS Multilink Forwarding Parameters FHS information in the MFV, and also caches the first B entry in the MFVI List as "FHS-Client" when FHS FMT-Forward/Mode are both set to enable future direct forwarding to this FHS Client. The FHS Gateway then generates a "B1" MFVI for the MFV and also writes it as the next B entry in the NS's MFVI List.

The FHS Gateway then examines the SRT prefixes corresponding to both FHS and LHS. If the FHS Gateway has a local interface connection to both the FHS and LHS (whether they are the same or different segments), the FHS/LHS Gateway caches the NS LHS information, writes its ULA suffix and LHS INADDR into the NS OMNI Multilink Forwarding Parameters LHS fields, then sets its own ULA as the source and the ULA of the LHS Proxy/Server as the destination while selecting an appropriate identification. If the FHS and LHS prefixes are different, the FHS Gateway instead sets the LHS Subnet Router Anycast ULA as the destination. The FHS Gateway then recalculates the NS checksum, selects an appropriate Identification and L2 headers as above then forwards the carrier packets into the secured spanning tree.

When the FHS and LHS Gateways are different, the LHS Gateway will receive carrier packets over the secured spanning tree from the FHS Gateway. The LHS Gateway reassembles/decapsulates to obtain the NS then verifies the checksum and creates an MFV (i.e., the same as the FHS Gateway had done) while assigning the current B entry in the MFVI List as the MFV "B2" index. The LHS Gateway also caches the ULA of the FHS Gateway found in the Multilink Forwarding Parameters as the spanning tree address for "B2", caches the NS Multilink Forwarding Parameters LHS information then generates a "B1" MFVI for the MFV while also writing it as the next B entry in the MFVI List. The LHS Gateway also writes its own ULA suffix and LHS INADDR into the OMNI Multilink Forwarding Parameters. The LHS Gateway then sets the its own ULA as the source and the ULA of the LHS Proxy/Server as the OAL destination, recalculates the checksum, selects an appropriate Identification, then fragments while including appropriate L2 headers and forwards the carrier packets into the secured spanning tree.

When the LHS Proxy/Server receives the carrier packets from the secured spanning tree, it reassembles/decapsulates to obtain the NS, verifies the checksum then verifies that the LHS information supplied by the FHS source is consistent with its own cached information. If the information is consistent, the LHS Proxy/Server then creates an MFV and assigns the current B entry in the MFVI List as the "B2" MFVI the same as for the prior hop. If the NS destination is the XLA of the target Client, the LHS Proxy/Server also generates a "B1" MFVI and assigns it both to the MFVI and as the next B entry in the MFVI List. The LHS Proxy/Server then examines FHS FMT; if FMT-Forward is clear and FMT-Mode is set, the LHS Proxy/Server creates a NCE for the ULA of the FHS Proxy/Server (if necessary) and sets the state to STALE, then caches any Tunnel Window Synchronization parameters.

If the NS destination is its own ULA, the LHS Proxy/Server next prepares to return a solicited NA with Job code '01'. If the NS source was the XLA of the FHS Client, the LHS Proxy/Server first

creates or updates an NCE for the XLA with state set to STALE. The LHS Proxy/Server next caches the NS OMNI Neighbor Coordination header window synchronization parameters and Multilink Forwarding Parameters information (including the MFVI List) in the NCE corresponding to the ULA source. When the LHS Proxy/Server forwards future carrier packets based on the NCE, it can populate reverse-path forwarding information in a CRH-32 routing header to enable forwarding based on the cached MFVI List B entries instead of ULA addresses.

The LHS Proxy/Server then creates an NA with Job code '01' while copying the NS OMNI Multilink Forwarding Parameters FHS/LHS information into the corresponding fields in the NA. The LHS Proxy/Server then generates an "A1" MFVI and both assigns it to the MFV and includes it as the first A entry in NA's MFVI List (see: [I-D.templin-6man-omni] for details on MFVI List A/B processing). The LHS Proxy/Server then includes end-to-end window synchronization parameters in the OMNI Neighbor Coordination header (if necessary) and also tunnel window synchronization parameters in the Multilink Forwarding Parameters (if necessary). The LHS Proxy/Server then encapsulates the NA, calculates the checksum, sets the source to its own ULA, sets the destination to the ULA of the LHS Gateway, selects an appropriate Identification value and L2 headers then forwards the carrier packets into the secured spanning tree.

If the NS destination was the XLA of the LHS Client, the LHS Proxy/Server instead includes an authentication signature in the NS if necessary (otherwise recalculates the checksum), then changes the OAL source to its own ULA and changes the destination to the ULA-MNP of the LHS Client. The LHS Proxy/Server then selects an appropriate Identification value, fragments if necessary, includes appropriate L2 headers and forwards the carrier packets to the LHS Client. When the LHS Client receives the carrier packets, it verifies the Identification and reassembles/decapsulates to obtain the NS then verifies the authentication signature or checksum. The LHS Client then creates a NCE for the NS ULA source address in the STALE state. If LHS FMT-Forward is set, FHS FMT-Forward is clear and the NS source was an XLA, the Client also creates a NCE for the ULA of the FHS Proxy/Server in the STALE state and caches any Tunnel Window Synchronization parameters. The Client then caches the NS OMNI Neighbor Coordination header window synchronization parameters and Multilink Forwarding Parameters in the NCE corresponding to the NS ULA source, then creates an MFV and assigns both the current MFVI List B entry as "B2" and a locally generated "A1" MFVI the same as for previous hops (the LHS Client also includes the "A1" value in the solicited NA - see above and below). The LHS Client also caches the previous MFVI List B entry as "LHS-Gateway" since it can include this value when it sends future carrier packets directly to the Gateway (following appropriate neighbor coordination).

The LHS Client then prepares an NA using exactly the same procedures as for the LHS Proxy/Server above, except that it uses its XLA as the source and the ULA or XLA of the FHS correspondent as the destination. The LHS Client also includes an authentication signature if necessary (otherwise calculates the checksum), then encapsulates the NA with OAL source set to its own ULA-MNP and destination set to the ULA of the LHS Proxy/Server, includes an appropriate Identification and L2 headers and forwards the carrier packets to the LHS Proxy/Server. When the LHS Proxy/Server receives the carrier packets, it verifies the Identifications, reassembles/decapsulates to obtain the NA, verifies the authentication signature or checksum, then uses the current MFVI List B entry to locate the MFV. The LHS Proxy/Server then writes the current MFVI List A entry as the "A2" value for the MVE, generates an "A1" MFVI and both assigns it to the MFV and writes it as the next MFVI List A entry. The LHS Proxy/Server then examines the FHS/LHS FMT codes to determine if it needs to include Tunnel Window Synchronization parameters. The LHS Proxy/Server then recalculates the checksum, re-fragments the NA while setting the OAL source to its own ULA and destination to the ULA of the LHS Gateway, includes an appropriate Identification and L2 headers and forwards the carrier packets into the secured spanning tree.

When the LHS Gateway receives the carrier packets, it reassembles/decapsulates to obtain the NA while verifying the checksum then uses the current MFVI List B entry to locate the MFV. The LHS Gateway then writes the current MFVI List A entry as the MFV "A2" index and generates a new "A1" value which it both assigns the MFV and writes as the next MFVI List A entry. (The LHS Gateway also caches the first A entry in the MFVI List as "LHS-Client" when LHS FMT-Forward/Mode are both set to enable future direct forwarding to this LHS Client.) If the LHS Gateway is connected directly to both the FHS and LHS segments (whether the segments are the same or different), the FHS/LHS Gateway will have already cached the FHS/LHS information based on the original NS. The FHS/LHS Gateway recalculates the checksum then re-fragments the NA while setting the OAL source to its own ULA and destination to the ULA of the FHS Proxy/Server. If the FHS and LHS prefixes are different, the FHS Gateway instead re-fragments while setting the destination to the ULA of the FHS Gateway. The LHS Gateway selects an appropriate Identification and L2 headers then forwards the carrier packets into the secured spanning tree.

When the FHS and LHS Gateways are different, the FHS Gateway will receive the carrier packets from the LHS Gateway over the secured spanning tree. The FHS Gateway reassembles/decapsulates to obtain the NA while verifying the checksum, then locates the MFV based on the current MFVI List B entry. The FHS Gateway then assigns the

current MFVI List A entry as the MFV "A2" index and caches the ULA of the LHS Gateway as the spanning tree address for "A2". The FHS Gateway then generates an "A1" MFVI and both assigns it to the MFV and writes it as the next MFVI List A entry while also writing its ULA and INADDR in the NA FHS Gateway fields. The FHS Gateway then recalculates the checksum, re-encapsulates/re-fragments with its own ULA as the source, with the ULA of the FHS Proxy/Server as the destination, then selects an appropriate Identification value and L2 headers and forwards the carrier packets into the secured spanning tree.

When the FHS Proxy/Server receives the carrier packets from the secured spanning tree, it reassembles/decapsulates to obtain the NA while verifying the checksum then locates the MFV based on the current MFVI List B entry. The FHS Proxy/Server then assigns the current MFVI List A entry as the "A2" MFVI the same as for the prior hop. If the NA destination is its own ULA, the FHS Proxy/Server then caches the NA Multilink Forwarding Parameters with the MFV and examines LHS FMT. If FMT-Forward is clear, the FHS Proxy/Server locates the NCE for the ULA of the LHS Proxy/Server and sets the state to REACHABLE then caches any Tunnel Window Synchronization parameters. If the NA source is the XLA of the LHS Client, the FHS Proxy/Server then locates the LHS Client NCE and sets the state to REACHABLE then caches the OMNI Neighbor Coordination header window synchronization parameters and prepares to return an NA acknowledgement, if necessary.

If the NA destination is the XLA of the FHS Client, the FHS Proxy/Server also searches for and updates the NCE for the ULA of the LHS Proxy/Server if necessary the same as above. The FHS Proxy/Server then generates an "A1" MFVI and assigns it both to the MFVI and as the next MFVI List A entry, then includes an authentication signature or checksum in the NA message. The FHS Proxy/Server then sets the OAL source to its own ULA and sets the destination to the ULA-MNP of the FHS Client, then selects an appropriate Identification value and L2 headers and forwards the carrier packets to the FHS Client.

When the FHS Client receives the carrier packets, it verifies the Identification, reassembles/decapsulates to obtain the NA, verifies the authentication signature or checksum, then locates the MFV based on the current MFVI List B entry. The FHS Client then assigns the current MFVI List A entry as the "A2" MFVI the same as for the prior hop. The FHS Client then caches the NA Multilink Forwarding Parameters (including the MFVI List) with the MFV and examines LHS FMT. If FMT-Forward is clear, the FHS Client locates the NCE for the ULA of the LHS Proxy/Server and sets the state to REACHABLE then caches any Tunnel Window Synchronization parameters. If the NA source is the XLA of the LHS Client, the FHS Proxy/Server then

locates the LHS Client NCE and sets the state to REACHABLE then caches the OMNI Neighbor Coordination header window synchronization parameters and prepares to return an NA acknowledgement, if necessary. The FHS Client also caches the previous MFVI List A entry as "FHS-Gateway" since it can include this value when it sends future carrier packets directly to the Gateway (following appropriate neighbor coordination).

If either the FHS Client or FHS Proxy/Server needs to return an acknowledgement to complete window synchronization, it prepares a uNA message with an OMNI Multilink Forwarding Parameters sub-option with Job code set to '10' (Follow A; Record B) (note that this step is unnecessary when Rapid Commit route optimization is used per Section 3.13.3). The FHS node sets the source to its own ULA or XLA, sets the destination to the ULA or XLA of the LHS node then includes Tunnel Window Synchronization parameters if necessary. The FHS node next sets the MFVI List to the cached list of A entries received in the Job code '01' NA, but need not set any other FHS/LHS information. The FHS node then encapsulates the uNA message in an OAL header with its own ULA as the source. If the FHS node is the Client, it next sets the ULA of the FHS Proxy/Server as the OAL destination, includes an authentication signature or checksum, selects an appropriate Identification value and L2 headers and forwards the carrier packets to the FHS Proxy/Server. The FHS Proxy/Server then verifies the Identification, reassembles/decapsulates, verifies the authentication signature or checksum, then uses the current MFVI List A entry to locate the MFV. The FHS Proxy/Server then writes its "B1" MFVI as the next MFVI List B entry and determines whether it needs to include Tunnel Window Synchronization parameters the same as it had done when it forwarded the original NS.

The FHS Proxy/Server recalculates the uNA checksum then re-fragments while setting its own ULA as the source and the ULA of the FHS Gateway as the destination, then selects an appropriate Identification and L2 headers and forwards the carrier packets into the secured spanning tree. When the FHS Gateway receives the carrier packets, it reassembles/decapsulates to obtain the uNA while verifying the checksum then uses the current MFVI List A entry to locate the MFV. The FHS Gateway then writes its "B1" MFVI as the next MFVI List B entry, then re-fragments while setting the OAL source and destination. If the FHS Gateway is also the LHS Gateway, it sets the ULA of the LHS Proxy/Server as the destination; otherwise it sets the ULA of the LHS Gateway. The FHS Gateway recalculates the checksum then selects an appropriate Identification and L2 headers, re-fragments/forwards the carrier packets into the secured spanning tree. If an LHS Gateway receives the carrier packets, it processes them exactly the same as the FHS Gateway had done while setting the carrier packet destination to the ULA of the LHS Proxy/Server.

When the LHS Proxy/Server receives the carrier packets, it reassembles/decapsulates to obtain the uNA message while verifying the checksum. The LHS Proxy/Server then locates the MFV based on the current MFVI List A entry then determines whether it is a tunnel ingress the same as for the original NS. If it is a tunnel ingress, the LHS Proxy/Server updates the NCE for the tunnel far-end based on the Tunnel Window Synchronization parameters. If the uNA destination is its own ULA, the LHS Proxy/Server next updates the NCE for the source ULA based on the OMNI Neighbor Coordination header window synchronization parameters and MAY compare the MFVI List to the version it had cached in the MFV based on the original NS.

If the uNA destination is the XLA of the LHS Client, the LHS Proxy/Server instead writes its "B1" MFV as the next MFVI List B entry, includes an authentication signature or checksum, writes its own ULA as the OAL source and the ULA-MNP of the Client as the OAL destination then selects an appropriate Identification and L2 headers and forwards the resulting carrier packets to the LHS Client. When the LHS Client receives the carrier packets, it verifies the Identification, reassembles/decapsulates to obtain the uNA, verifies the authentication signature or checksum then processes the message exactly the same as for the LHS Proxy/Server case above.

Following the NS/NA exchange with Multilink Forwarding Parameters, OAL end systems and tunnel endpoints can begin exchanging ordinary carrier packets with Identification values within their respective send/receive windows without requiring security signatures and/or secured spanning tree traversal. Either peer can refresh window synchronization parameters and/or send other carrier packets requiring security at any time using the same secured procedures described above. OAL end systems and intermediate nodes can also use their own A1/B1 MFVIs when they receive carrier packets to unambiguously locate the correct MFV and determine directionality and can use any discovered A2/B2 MFVIs to forward carrier packets to other OAL nodes that configure the corresponding A1/B1 MFVIs. When an OAL node uses an MFVI included in a carrier packet to locate an MFV, it need not also examine the carrier packet addresses.

OAL sources can also begin including CRH-32s in carrier packets with a list of A/B MFVIs that OAL intermediate nodes can use for shortest-path carrier packet forwarding based on MFVIs instead of spanning tree addresses. OAL sources and intermediate nodes can also begin forwarding carrier packets with OAL compressed headers termed "OCH" (see: [I-D.templin-6man-omni]) that include only a single A/B MFVI meaningful to the next hop, since all nodes in the path up to (and sometimes including) the OAL destination have already established MFV forwarding information. Note that when an FHS OAL source receives a solicited NA with Job code '01', the message will contain an MFVI

List with A entries populated in the reverse order needed for populating a CRH-32 routing header. The FHS OAL source must therefore write the MFVI List A entries last-to-first when it populates a CRH-32, or must select the correct A entry to include in an OCH header based on the intended OAL intermediate node or destination.

When a Gateway receives unsecured carrier packets destined to a local segment Client that has asserted direct reachability, the Gateway performs direct carrier packet forwarding while bypassing the local Proxy/Server based on the Client's advertised MFVIs and discovered NATed INADDR information (see: Section 3.13.4). If the Client cannot be reached directly (or if NAT traversal has not yet converged), the Gateway instead forwards carrier packets directly to the local Proxy/Server.

When a Proxy/Server receives carrier packets destined to a local Client or forwards carrier packets received from a local Client, it first locates the correct MFV. If the carrier packets include a secured IPv6 ND message, the Proxy/Server uses the Client's NCE established through RS/RA exchanges to re-encapsulate/re-fragment while forwarding outbound secured carrier packets via the secured spanning tree and forwarding inbound secured carrier packets while including an authentication signature or checksum. For ordinary carrier packets, the Proxy/Server uses the same MFV if directed by MFVI and/or OAL addressing. Otherwise it locates an MFV established through an NS/NA exchange between the Client and the remote peer, and forwards the carrier packets without first reassembling/decapsulating.

When a Proxy/Server or Client configured as a tunnel ingress receives a carrier packet with a full OAL header with a ULA-MNP source and CRH-32 routing header, or an OCH header with an MFVI that matches an MFV, the ingress encapsulates the carrier packet in a new full OAL header or an OCH header containing the next hop MFVI and an Identification value appropriate for the end-to-end window and the outer header containing an Identification value appropriate for the tunnel endpoints. When a Proxy/Server or Client configured as a tunnel egress receives an encapsulated carrier packet, it verifies the Identification in the outer header, then discards the outer header and forwards the inner carrier packet to the final destination.

When a Proxy/Server with FMT-Forward/Mode set to 0/1 for a source Client receives carrier packets from the source Client, it first reassembles to obtain the original OAL packet then re-fragments if necessary to cause the Client's packets to match the MPS on the path from the Proxy/Server as a tunnel ingress to the tunnel egress. The

Proxy/Server then performs OAL-in-OAL encapsulation and forwards the resulting carrier packets to the tunnel egress. When a Proxy/Server with FMT-Forward/Mode set to 0/1 for a target Client receives carrier packets from a tunnel ingress, it first decapsulates to obtain the original fragments then reassembles to obtain the original OAL packet. The Proxy/Server then re-fragments if necessary to cause the fragments to match the target Client's underlay interface (Path) MTU and forwards the resulting carrier packets to the target Client.

When a source Client forwards carrier packets it can employ header compression according to the MFVIs established through an NS/NA exchange with a remote or local peer. When the source Client forwards to a remote peer, it can forward carrier packets to a local SRT Gateway (following the establishment of INADDR information) while bypassing the Proxy/Server (see: Section 3.13.4). When a target Client receives carrier packets that match a local MFV, the Client first verifies the Identification then decompresses the headers if necessary, reassembles if necessary to obtain the OAL packet then decapsulates and delivers the IP packet to upper layers.

When synchronized peer Clients in the same SRT segment with FMT-Forward and FMT-Mode set discover each other's NATed INADDR addresses, they can exchange carrier packets directly with header compression using MFVIs discovered as above (see: Section 3.13.5). The FHS Client will have cached the A MFVI for the LHS Client, which will have cached the B MFVI for the FHS Client.

After window synchronization state has been established, the ROS and ROR can begin forwarding carrier packets while performing additional NS/NA exchanges as above to update window state, register new interface pairs for optimized multilink forwarding and/or confirm reachability. The ROS sends carrier packets to the FHS Gateway discovered through the NS/NA exchange. The FHS Gateway then forwards the carrier packets over the unsecured spanning tree to the LHS Gateway, which forwards them via LHS encapsulation to the LHS Proxy/Server or directly to the target Client itself. The target Client in turn sends packets to the ROS in the reverse direction while forwarding through the Gateways to minimize Proxy/Server load whenever possible.

While the ROS continues to actively forward packets to the target Client, it is responsible for updating window synchronization state and per-interface reachability before expiration. Window synchronization state is shared by all underlay interfaces in the ROS' NCE that use the same destination ULA so that a single NS/NA exchange applies for all interfaces regardless of the specific interface used to conduct the exchange. However, the window synchronization exchange only confirms target Client reachability

over the specific underlay interface pair. Reachability for other underlay interfaces that share the same window synchronization state must be determined individually using additional NS/NA messages.

3.13.3. Rapid Commit Route Optimization

When the ROR receives an NS(AR) with a set of Interface Attributes for the source Client, it can perform "rapid commit" by immediately invoking multilink route optimization as above instead of returning an NA(AR). In order to perform rapid commit, the ROR prepares a unicast NS message with an OMNI option with window synchronization information responsive to the NS(AR), with a Multilink Forwarding Parameters sub-option selected for a specific underlay interface pair and with Interface Attributes for all of the ROR's other underlay interfaces. The ROR can also include ordinary IP packets as OAL super-packet extensions to the NS message if it has immediate data to send to the ROS. The ROR then returns the NS to the ROS the same as for the NA(AR) case.

When the NS message traverses the return path to the ROR, all intermediate nodes in the path establish state exactly the same as for an ordinary NS/NA multilink route optimization exchange. When the NS message arrives at the ROS, the window synchronization parameters confirm that the NS is taking the place of the NA(AR), thereby eliminating an extraneous message transmission and associated delay. The ROS then completes the route optimization by returning a responsive NA.

Note: The ROS must accept unicast NS messages with an ACK matching the SYN included in the NS(AR) as an equivalent message replacement for the NA(AR). Address resolution and multilink forwarding coordination can therefore be coordinated in a single three-way handshake connection with minimal messaging and delay (i.e., as opposed to a four-message exchange).

3.13.4. Client/Gateway Route Optimization

Following multilink route optimization for specific underlay interface pairs, ROS/ROR Clients located on open INETs can invoke Client/Gateway route optimization to improve performance and reduce load and congestion on their respective FHS/LHS Proxy/Servers. To initiate Client/Gateway route optimization, the Client prepares an NS message with its own XLA address as the source and the ULA of its Gateway as the destination while creating a NCE for the Gateway if necessary. The NS message must be no larger than the minimum MPS and encapsulated as an atomic fragment.

The Client then includes an Interface Attributes sub-option for its underlay interface as well as an authentication signature but does not include window synchronization parameters. The Client then performs OAL encapsulation with its own ULA-MNP as the source and the ULA of the Gateway as the destination while including a randomly-chosen Identification value, then performs L2 encapsulation on the atomic fragment and sends the resulting carrier packet directly to the Gateway.

When the Gateway receives the carrier packet, it verifies the authentication signature then creates a NCE for the Client. The Gateway then caches the L2 encapsulation addresses (which may have been altered by one or more NATs on the path) as well as the Interface Attributes for this Client omIndex, and marks this Client underlay interface as "trusted". The Gateway then prepares an NA reply with its own ULA as the source and the XLA of the Client as the destination where the NA again must be no larger than the minimum MPS.

The Gateway then echoes the Client's Interface Attributes, includes an Origin Indication with the Client's observed L2 addresses and includes an authentication signature. The Gateway then performs OAL encapsulation with its own ULA as the source and the ULA-MNP of the Client as the destination while using the same Identification value that appeared in the NS, then performs L2 encapsulation on the atomic fragment and sends the resulting carrier packet directly to the Client.

When the Client receives the NA reply, it caches the carrier packet L2 source address information as the Gateway target address via this underlay interface while marking the interface as "trusted". The Client also caches the Origin Indication L2 address information as its own (external) source address for this underlay interface.

After the Client and Gateway have established NCEs as well as "trusted" status for a particular underlay interface pair, each node can begin forwarding ordinary carrier packets intended for this multilink route optimization directly to one another while omitting the Proxy/Server from the forwarding path while the status is "trusted". The NS/NA messaging will have established the correct state in any NATs in the path so that NAT traversal is naturally supported. The Client and Gateway must maintain a timer that watches for activity on the path; if no carrier packets and/or NS/NA messages are sent or received over the path before NAT state is likely to have expired, the underlay interface pair status becomes "untrusted".

Thereafter, when the Client forwards a carrier packet with an MFVI toward the Gateway as the next hop, the Client uses the MFVI for the Gateway (discovered during multilink route optimization) instead of the MFVI for its Proxy/Server; the Gateway will accept the packet from the Client if and only if the underlay interface status is trusted and if the MFVI is correct for the next hop toward the final destination. (The same is true in the reverse direction when the Gateway sends carrier packets directly to the Client.)

Note that the Client and Gateway each maintain a single NCE, but that the NCE may aggregate multiple underlay interface pairs. Each underlay interface pair may use differing source and target L2 addresses according to NAT mappings, and the "trusted/untrusted" status of each pair must be tested independently. When no "trusted" pairs remain, the NCE is deleted.

Note that the above method requires Gateways to participate in NS/NA message authentication signature application and verification. In an alternate approach, the Client could instead exchange NS/NA messages with authentication signatures via its Proxy/Server but addressed to the ULA of the Gateway, and the Proxy/Server and Gateway could relay the messages over the secured spanning tree. However, this would still require the Client to send additional messages toward the L2 address of the Gateway to populate NAT state; hence the savings in complexity for Gateways would result in increased message overhead for Clients.

3.13.5. Client/Client Route Optimization

When the ROS/ROR Clients are both located on the same SRT segment, Client-to-Client route optimization is possible following the establishment of any necessary state in NATs in the path. Both Clients will have already established state via their respective shared segment Proxy/Servers (and possibly also the shared segment Gateway) and can begin forwarding packets directly via NAT traversal while avoiding any Proxy/Server and/or Gateway hops.

When the ROR/ROS Clients on the same SRT segment perform the initial NS/NA exchange to establish Multilink Forwarding state, they also include an Origin Indication (i.e., in addition to Multilink Forwarding Parameters) with the mapped addresses discovered during the RS/RA exchanges with their respective Proxy/Servers. After the MFV paths have been established, both Clients can begin sending packets via strict MFV paths while establishing a direct path for Client-to-Client route optimization.

To establish the direct path, either Client (acting as the source) transmits a bubble to the mapped L2 address for the target Client which primes its local chain of NATs for reception of future packets from that L2 address (see: [RFC4380] and [I-D.templin-6man-omni]). The source Client then prepares an NS message with its own XLA as the source, with the XLA of the target as the destination and with an OMNI option with an Interface Attributes sub-option. The source Client then encapsulates the NS in an OAL header with its own ULA-MNP as the source, with the ULA-MNP of the target Client as the destination and with an in-window Identification for the target. The source Client then fragments and encapsulates in L2 headers addressed to its FHS Proxy/Server then forwards the resulting carrier packets to the Proxy/Server.

When the FHS Proxy/Server receives the carrier packets, it re-encapsulates and forwards them as unsecured carrier packets according to MFV state where they will eventually arrive at the target Client which can verify that the identifications are within the acceptable window and reassemble if necessary. Following reassembly, the target Client prepares an NA message with its own XLA as the source, with the XLA of the source Client as the destination and with an OMNI option with an Interface Attributes sub-option. The target Client then encapsulates the NA in an OAL header with its own ULA-MNP as the source, with the ULA-MNP of the source Client as the destination and with an in-window Identification for the source Client. The target Client then fragments and encapsulates in L2 headers addressed to the source Client's Origin addresses then forwards the resulting carrier packets directly to the source Client.

Following the initial NS/NA exchange, both Clients mark their respective (source, target) underlay interface pairs as "trusted" for no more than ReachableTime seconds. While the Clients continue to exchange carrier packets via the direct path avoiding all Proxy/Servers and Gateways, they should perform additional NS/NA exchanges via their local Proxy/Servers to refresh NCE state as well as send additional bubbles to the peer's Origin address information if necessary to refresh NAT state.

Note that these procedures are suitable for a widely-deployed but basic class of NATs. Procedures for advanced NAT classes are outlined in [RFC6081], which provides mechanisms that can be employed equally for AERO using the corresponding sub-options specified by OMNI.

Note also that each communicating pair of Clients may need to maintain NAT state for peer to peer communications via multiple underlay interface pairs. It is therefore important that Origin Indications are maintained with the correct peer interface and that the NCE may cache information for multiple peer interfaces.

Note that the source and target Client exchange Origin information during the secured NS/NA multilink route optimization exchange. This allows for subsequent NS/NA exchanges to proceed using only the Identification value as a data origin confirmation. However, Client-to-Client peerings that require stronger security may also include authentication signatures for mutual authentication.

3.13.6. Client-to-Client OMNI Link Extension

Clients may be recursively nested within the ENETs of other Clients. When a Client is the downstream-attached ENET neighbor of an upstream Client, it still supports the route optimization functions discussed above by maintaining an MFIB and assigning MFVI values. When the Client processes an IPv6 ND NS/NA message that includes a Multilink Forwarding Parameters sub-option, it writes its MFVI information as the first/last MFVI list entry the same as for the single Client case discussed above.

The Client then forwards the NS/NA message to the next Client in the extended OMNI link toward the FHS/LHS Proxy/Server, which records the MFVI value then overwrites the MFVI list entry with its own MFVI value. This process iteratively continues until the Client that will forward the NS/NA message to the FHS/LHS Proxy/Server is reached, at which point the NS/NA MFVI list entries are populated by the intermediate nodes on the path to the LHS/FHS the same as discussed above.

In this way, each Client in the extended OMNI link discovers the A/B MFVIs of the next/previous Client without intruding into the Multilink Forwarding Parameters MFVI list. Therefore the list can remain fixed at 5 entries even though the Client-to-Client OMNI link extension can be arbitrarily long. Therefore, route optimization is not possible between consecutive Client members of the extended OMNI link but becomes possible at the Internetworking border that separates the FHS and LHS elements.

3.13.7. Intra-ANET/ENET Route Optimization for AERO Peers

When a Client forwards a packet from a Host or another Client connected to one of its downstream ENETs to a peer within the same downstream ENET, the Client returns an IPv6 ND Redirect message to inform the source that that target can be reached directly. The contents of the Redirect message are the same as specified in [RFC4861].

In the same fashion, when a Proxy/Server forwards a packet from a Host or Client connected to one of its downstream ANETs to a peer within the same downstream ANET, the Proxy/Server returns an IPv6 ND Redirect message.

All other route optimization functions are conducted per the NS/NA messaging discussed in the previous sections.

3.14. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) per [RFC4861] either reactively in response to persistent link-layer errors (see Section 3.11) or proactively to confirm reachability. The NUD algorithm is based on periodic control message exchanges and may further be seeded by IPv6 ND hints of forward progress, but care must be taken to avoid inferring reachability based on spoofed information. For example, IPv6 ND message exchanges that include authentication codes and/or in-window Identifications may be considered as acceptable hints of forward progress, while spurious random carrier packets should be ignored.

AERO nodes can perform NS/NA(NUD) exchanges over the OMNI link secured spanning tree (i.e. the same as described above) to test reachability without risk of DoS attacks from nodes pretending to be a neighbor. These NS/NA(NUD) messages use the unicast XLAs/ULAs of the parties involved in the NUD test. When only reachability information is required without updating any other NCE state, AERO nodes can instead perform NS/NA(NUD) exchanges directly between neighbors without employing the secured spanning tree as long as they include in-window Identifications and either an authentication signature or checksum.

After an ROR directs an ROS to a target neighbor with one or more link-layer addresses, either node may invoke multilink forwarding state initialization to establish authentic intermediate node state between specific underlay interface pairs which also tests their reachability. Thereafter, either node acting as the source may perform additional reachability probing through NS(NUD) messages over the SRT secured or unsecured spanning tree, or through NS(NUD)

messages sent directly to an underlay interface of the target itself. While testing a target underlay interface, the source can optionally continue to forward carrier packets via alternate interfaces, maintain a small queue of carrier packets until target reachability is confirmed or include them as trailing data with the NS(NUD) in an OAL super-packet [I-D.templin-6man-omni].

NS(NUD) messages are encapsulated, fragmented and transmitted as carrier packets the same as for ordinary original IP data packets, however the encapsulated destinations are either the ULA or XLA of the source and either the ULA of the LHS Proxy/Server or the XLA of the target itself. The source encapsulates the NS(NUD) message the same as described in Section 3.13.2 and includes an Interface Attributes sub-option with omIndex set to identify its underlay interface used for forwarding. The source then includes an in-window Identification, fragments the OAL packet and forwards the resulting carrier packets into the unsecured spanning tree, directly to the target if it is in the local segment or directly to a Gateway in the local segment.

When the target receives the NS(NUD) carrier packets, it verifies that it has a NCE for this source and that the Identification is in-window, then submits the carrier packets for reassembly. The target then verifies the authentication signature or checksum, then searches for Interface Attributes in its NCE for the source that match the NS(NUD) for the NA(NUD) reply. The target then prepares the NA(NUD) with the source and destination addresses reversed, encapsulates and sets the OAL source and destination, includes an Interface Attributes sub-option in the NA(NUD) to identify the omIndex of the underlay interface the NS(NUD) arrived on and sets the Target Address to the same value included in the NS(NUD). The target next sets the R flag to 1, the S flag to 1 and the O flag to 1, then selects an in-window Identification for the source and performs fragmentation. The node then forwards the carrier packets into the unsecured spanning tree, directly to the source if it is in the local segment or directly to a Gateway in the local segment.

When the source receives the NA(NUD), it marks the target underlay interface tested as "trusted". Note that underlay interface states are maintained independently of the overall NCE REACHABLE state, and that a single NCE may have multiple target underlay interfaces in various "trusted/untrusted" states while the NCE state as a whole remains REACHABLE.

3.15. Mobility Management and Quality of Service (QoS)

AERO is a fully Distributed Mobility Management (DMM) service in which each Proxy/Server is responsible for only a small subset of the Clients on the OMNI link. This is in contrast to a Centralized Mobility Management (CMM) service where there are only one or a few network mobility collective entities for large Client populations. Clients coordinate with their associated FHS and Hub Proxy/Servers via RS/RA exchanges to maintain the DMM profile, and the AERO routing system tracks all current Client/Proxy/Server peering relationships.

Hub Proxy/Servers provide a designated router service for their dependent Clients, while FHS Proxy/Servers provide a proxy conduit between the Client and both the Hub and OMNI link in general. Clients are responsible for maintaining neighbor relationships with their Proxy/Servers through periodic RS/RA exchanges, which also serves to confirm neighbor reachability. When a Client's underlay interface attributes change, the Client is responsible for updating the Hub Proxy/Server through new RS/RA exchanges using the FHS Proxy/Server as a first-hop conduit. The FHS Proxy/Server can also act as a proxy to perform some IPv6 ND exchanges on the Client's behalf without consuming bandwidth on the Client underlay interface.

Mobility management considerations are specified in the following sections.

3.15.1. Mobility Update Messaging

RORs and ROSSs accommodate Client mobility and/or multilink change events by sending secured uNA messages to each active neighbor. When an ROR/ROS sends a uNA message, it sets the IPv6 source address to the its own ULA or XLA, sets the destination address to the neighbor's ULA or XLA and sets the Target Address to the Client's XLA. The ROR/ROS also includes an OMNI option with OMNI Neighbor Coordination header Preflen set to the prefix length associated with the Client's XLA, includes Interface Attributes and Traffic Selectors for the Client's underlay interfaces and includes an authentication signature if necessary. The ROR then sets the uNA R flag to 1, S flag to 0 and O flag to 1, then encapsulates the message in an OAL header with source set to its own ULA and destination set to its FHS Proxy/Server's ULA. When the FHS Proxy/Server receives the uNA, it reassembles, verifies the authentication signature, then changes the destination to the ULA corresponding to the uNA destination and forwards the uNA into the secured spanning tree.

As discussed in Section 7.2.6 of [RFC4861], the transmission and reception of uNA messages is unreliable but provides a useful optimization. In well-connected Internetworks with robust data links

uNA messages will be delivered with high probability, but in any case the ROR/ROS can optionally send up to MAX_NEIGHBOR_ADVERTISEMENT uNAs to each neighbor to increase the likelihood that at least one will be received. Alternatively, the ROR/ROS can set the PNG flag in the uNA OMNI option header to request a uNA acknowledgement as specified in [I-D.templin-6man-omni].

When the ROR/ROS Proxy/Server receives a uNA message prepared as above, if the uNA destination was its own ULA the Proxy/Server uses the included OMNI option information to update its NCE for the target but does not reset ReachableTime since the receipt of a uNA message does not provide confirmation that any forward paths to the target Client are working. If the destination was the XLA of the ROR/ROS Client, the Proxy/Server instead changes the OAL source to its own ULA, includes an authentication signature if necessary, and includes an in-window Identification for this Client. Finally, if the uNA message PNG flag was set, the node that processes the uNA returns a uNA acknowledgement as specified in [I-D.templin-6man-omni].

3.15.2. Announcing Link-Layer Information Changes

When a Client needs to change its underlay Interface Attributes and/or Traffic Selectors (e.g., due to a mobility event), the Client sends an RS message to its Hub Proxy/Server via a first-hop FHS Proxy/Server, if necessary. The RS includes an OMNI option with an Interface Attributes sub-option with the omIndex and with new link quality and any other information.

Note that the first FHS Proxy/Server may change due to the underlay interface change. If the Client supplies the address of the former FHS Proxy/Server, the new FHS Proxy/Server can send a departure indication (see below); otherwise, any stale state in the former FHS Proxy/Server will simply expire after ReachableTime expires with no effect on the Hub Proxy/Server.

Up to MAX_RTR_SOLICITATIONS RS messages MAY be sent in parallel with sending carrier packets containing user data in case one or more RAs are lost. If all RAs are lost, the Client SHOULD re-associate with a new Proxy/Server.

After performing the RS/RA exchange, the Client sends uNA messages to all neighbors the same as described in the previous section.

3.15.3. Bringing New Links Into Service

When a Client needs to bring new underlay interfaces into service (e.g., when it activates a new data link), it sends an RS message to the Hub Proxy/Server via a FHS Proxy/Server for the underlay interface (if necessary) with an OMNI option that includes an Interface Attributes sub-option with appropriate link quality values and with link-layer address information for the new link. The Client then again sends uNA messages to all neighbors the same as described above.

3.15.4. Deactivating Existing Links

When a Client needs to deactivate an existing underlay interface, it sends a uNA message toward the Hub Proxy/Server via an FHS Proxy/Server with an OMNI option with appropriate Interface Attributes values for the deactivated link - in particular, the link quality value 0 assures that neighbors will cease to use the link.

If the Client needs to send uNA messages over an underlay interface other than the one being deactivated, it MUST include Interface Attributes with appropriate link quality values for any underlay interfaces being deactivated. The Client then again sends uNA messages to all neighbors the same as described above.

Note that when a Client deactivates an underlay interface, neighbors that receive the ensuing uNA messages need not purge all references for the underlay interface from their neighbor cache entries. The Client may reactivate or reuse the underlay interface and/or its omIndex at a later point in time, when it will send new RS messages to an FHS Proxy/Server with fresh interface parameters to update any neighbors.

3.15.5. Moving Between Proxy/Servers

The Client performs the procedures specified in Section 3.12.2 when it first associates with a new Hub Proxy/Server or renews its association with an existing Hub Proxy/Server.

When a Client associates with a new Hub Proxy/Server, it sends RS messages to register its underlay interfaces with the new Hub while including the old Hub's ULA in the "Old Hub Proxy/Server ULA" field of a Proxy/Server Departure OMNI sub-option. When the new Hub Proxy/Server returns the RA message via the FHS Proxy/Server (acting as a Proxy), the FHS Proxy/Server sends a uNA to the old Hub Proxy/Server (i.e., if the ULA is non-zero and different from its own). The uNA has the XLA of the Client as the source and the ULA of the old hub as the destination and with OMNI Neighbor Coordination header Preflen

set to 0. The FHS Proxy/Server encapsulates the uNA in an OAL header with the ULA of the new Hub as the source and the ULA of the old Hub as the destination, the fragments and sends the carrier packets via the secured spanning tree.

When the old Hub Proxy/Server receives the uNA, it changes the Client's NCE state to DEPARTED, resets DepartTime and caches the new Hub Proxy/Server ULA. After a short delay (e.g., 2 seconds) the old Hub Proxy/Server withdraws the Client's MNP from the routing system. While in the DEPARTED state, the old Hub Proxy/Server forwards any carrier packets received via the secured spanning tree destined to the Client's ULA-MNP to the new Hub Proxy/Server's ULA. After DepartTime expires, the old Hub Proxy/Server deletes the Client's NCE.

Mobility events may also cause a Client to change to a new FHS Proxy/Server over a specific underlay interface at any time such that a Client RS/RA exchange over the underlay interface will engage the new FHS Proxy/Server instead of the old. The Client can arrange to inform the old FHS Proxy/Server of the departure by including a Proxy/Server Departure sub-option with a ULA for the "Old FHS Proxy/Server ULA", and the new FHS Proxy/Server will issue a uNA using the same procedures as outlined for the Hub above while using its own ULA as the source address. This can often result in successful delivery of packets that would otherwise be lost due to the mobility event.

Clients SHOULD NOT move rapidly between Hub Proxy/Servers in order to avoid causing excessive oscillations in the AERO routing system. Examples of when a Client might wish to change to a different Hub Proxy/Server include a Hub Proxy/Server that has gone unreachable, topological movements of significant distance, movement to a new geographic region, movement to a new OMNI link segment, etc.

3.16. Multicast

Clients provide an IGMP (IPv4) [RFC2236] or MLD (IPv6) [RFC3810] proxy service for its ENETs and/or hosted applications [RFC4605] and act as a Protocol Independent Multicast - Sparse-Mode (PIM-SM, or simply "PIM") Designated Router (DR) [RFC7761] on the OMNI link. Proxy/Servers act as OMNI link PIM routers for Clients on ANET, VPNed or Direct interfaces, and Relays also act as OMNI link PIM routers on behalf of nodes on other links/networks.

Clients on VPNed, Direct or ANET underlay interfaces for which the ANET has deployed native multicast services forward IGMP/MLD messages into the ANET. The IGMP/MLD messages may be further forwarded by a first-hop ANET access router acting as an IGMP/MLD-snooping switch [RFC4541], then ultimately delivered to an ANET Proxy/Server. The

FHS Proxy/Server then acts as an ROS to send NS(AR) messages to an ROR for the multicast source. Clients on INET and ANET underlay interfaces without native multicast services instead send NS(AR) messages as an ROS to cause their FHS Proxy/Server forward the message to an ROR. When the ROR receives an NA(AR) response, it initiates PIM protocol messaging according to the Source-Specific Multicast (SSM) and Any-Source Multicast (ASM) operational modes as discussed in the following sections.

3.16.1. Source-Specific Multicast (SSM)

When an ROS "X" (i.e., either a Client or Proxy/Server) acting as PIM router receives a Join/Prune message from a node on its downstream interfaces containing one or more ((S)ource, (G)roup) pairs, it updates its Multicast Routing Information Base (MRIB) accordingly. For each S belonging to a prefix reachable via X's non-OMNI interfaces, X then forwards the (S, G) Join/Prune to any PIM routers on those interfaces per [RFC7761].

For each S belonging to a prefix reachable via X's OMNI interface, X sends an NS(AR) message (see: Section 3.13) using its own ULA or XLA as the source address, the solicited node multicast address corresponding to S as the destination and the XLA of S as the target address. X then encapsulates the NS(AR) in an OAL header with source address set to its own ULA and destination address set to the ULA for S, then forwards the message into the secured spanning tree which delivers it to ROR "Y" that services S. The resulting NA(AR) will return an OMNI option with Interface Attributes for any underlay interfaces that are currently servicing S.

When X processes the NA(AR) it selects one or more underlay interfaces for S and performs an NS/NA multilink route optimization exchange over the secured spanning tree while including a PIM Join/Prune message for each multicast group of interest in the OMNI option. If S is located behind any Proxys "Z*", each Z* then updates its MRIB accordingly and maintains the XLA of X as the next hop in the reverse path. Since Gateways forward messages not addressed to themselves without examining them, this means that the (reverse) multicast tree path is simply from each Z* (and/or S) to X with no other multicast-aware routers in the path.

Following the initial combined Join/Prune and NS/NA messaging, X maintains a NCE for each S the same as if X was sending unicast data traffic to S. In particular, X performs additional NS/NA exchanges to keep the NCE alive for up to t_{periodic} seconds [RFC7761]. If no new Joins are received within t_{periodic} seconds, X allows the NCE to expire. Finally, if X receives any additional Join/Prune messages for (S,G) it forwards the messages over the secured spanning tree.

Client C that holds an MNP for source S may later depart from a first Proxy/Server Z1 and/or connect via a new Proxy/Server Z2. In that case, Y sends a uNA message to X the same as specified for unicast mobility in Section 3.15. When X receives the uNA message, it updates its NCE for the XLA for source S and sends new Join messages in NS/NA exchanges addressed to the new target Client underlay interface connection for S. There is no requirement to send any Prune messages to old Proxy/Server Z1 since source S will no longer source any multicast data traffic via Z1. Instead, the multicast state for (S,G) in Proxy/Server Z1 will soon expire since no new Joins will arrive.

3.16.2. Any-Source Multicast (ASM)

When an ROS X acting as a PIM router receives Join/Prune messages from a node on its downstream interfaces containing one or more (*,G) pairs, it updates its Multicast Routing Information Base (MRIB) accordingly. X first performs an NS/NA(AR) exchange to receive route optimization information for Rendezvous Point (RP) R for each G. X then includes a copy of each Join/Prune message in the OMNI option of an NS message with its own ULA or XLA as the source address and the ULA or XLA for R as the destination address, then encapsulates the NS message in an OAL header with its own ULA as the source and the ULA of R's Proxy/Server as the destination then sends the message into the secured spanning tree.

For each source S that sends multicast traffic to group G via R, Client S* that aggregates S (or its Proxy/Server) encapsulates the original IP packets in PIM Register messages, includes the PIM Register messages in the OMNI options of uNA messages, performs OAL encapsulation and fragmentation then forwards the resulting carrier packets with Identification values within the receive window for Client R* that aggregates R. Client R* may then elect to send a PIM Join to S* in the OMNI option of a uNA over the secured spanning tree. This will result in an (S,G) tree rooted at S* with R as the next hop so that R will begin to receive two copies of the original IP packet; one native copy from the (S, G) tree and a second copy from the pre-existing (*, G) tree that still uses uNA PIM Register encapsulation. R can then issue a uNA PIM Register-stop message over the secured spanning tree to suppress the Register-encapsulated stream. At some later time, if Client S* moves to a new Proxy/Server, it resumes sending original IP packets via uNA PIM Register encapsulation via the new Proxy/Server.

At the same time, as multicast listeners discover individual S's for a given G, they can initiate an (S,G) Join for each S under the same procedures discussed in Section 3.16.1. Once the (S,G) tree is established, the listeners can send (S, G) Prune messages to R so

that multicast original IP packets for group G sourced by S will only be delivered via the (S, G) tree and not from the (*, G) tree rooted at R. All mobility considerations discussed for SSM apply.

3.16.3. Bi-Directional PIM (BIDIR-PIM)

Bi-Directional PIM (BIDIR-PIM) [RFC5015] provides an alternate approach to ASM that treats the Rendezvous Point (RP) as a Designated Forwarder (DF). Further considerations for BIDIR-PIM are out of scope.

3.17. Operation over Multiple OMNI Links

An AERO Client can connect to multiple OMNI links the same as for any data link service. In that case, the Client maintains a distinct OMNI interface for each link, e.g., 'omni0' for the first link, 'omni1' for the second, 'omni2' for the third, etc. Each OMNI link would include its own distinct set of Gateways and Proxy/Servers, thereby providing redundancy in case of failures.

Each OMNI link could utilize the same or different ANET connections. The links can be distinguished at the link-layer via the SRT prefix in a similar fashion as for Virtual Local Area Network (VLAN) tagging (e.g., IEEE 802.1Q) and/or through assignment of distinct sets of MSPs on each link. This gives rise to the opportunity for supporting multiple redundant networked paths (see: Section 3.2.4).

The Client's IP layer can select the outgoing OMNI interface appropriate for a given traffic profile while (in the reverse direction) correspondent nodes must have some way of steering their original IP packets destined to a target via the correct OMNI link.

In a first alternative, if each OMNI link services different MSPs the Client can receive a distinct MNP from each of the links. IP routing will therefore assure that the correct OMNI link is used for both outbound and inbound traffic. This can be accomplished using existing technologies and approaches, and without requiring any special supporting code in correspondent nodes or Gateways.

In a second alternative, if each OMNI link services the same MSP(s) then each link could assign a distinct "OMNI link Anycast" address that is configured by all Gateways on the link. Correspondent nodes can then perform Segment Routing to select the correct SRT, which will then direct the original IP packet over multiple hops to the target.

3.18. DNS Considerations

AERO Client MNs and INET correspondent nodes consult the Domain Name System (DNS) the same as for any Internetworking node. When correspondent nodes and Client MNs use different IP protocol versions (e.g., IPv4 correspondents and IPv6 MNs), the INET DNS must maintain A records for IPv4 address mappings to MNs which must then be populated in Relay NAT64 mapping caches. In that way, an IPv4 correspondent node can send original IPv4 packets to the IPv4 address mapping of the target MN, and the Relay will translate the IPv4 header and destination address into an IPv6 header and IPv6 destination address of the MN.

When an AERO Client registers with an AERO Proxy/Server, the Proxy/Server can return the address(es) of DNS servers in RDNS options [RFC6106]. The DNS server provides the IP addresses of other MNs and correspondent nodes in AAAA records for IPv6 or A records for IPv4.

3.19. Transition/Coexistence Considerations

OAL encapsulation ensures that dissimilar INET partitions can be joined into a single unified OMNI link, even though the partitions themselves may have differing protocol versions and/or incompatible addressing plans. However, a commonality can be achieved by incrementally distributing globally routable (i.e., native) IP prefixes to eventually reach all nodes (both mobile and fixed) in all OMNI link segments. This can be accomplished by incrementally deploying AERO Gateways on each INET partition, with each Gateway distributing its MNPs and/or discovering non-MNP IP GUA prefixes on its INET links.

This gives rise to the opportunity to eventually distribute native IP addresses to all nodes, and to present a unified OMNI link view even if the INET partitions remain in their current protocol and addressing plans. In that way, the OMNI link can serve the dual purpose of providing a mobility/multilink service and a transition/coexistence service. Or, if an INET partition is transitioned to a native IP protocol version and addressing scheme that is compatible with the OMNI link MNP-based addressing scheme, the partition and OMNI link can be joined by Gateways.

Relays that connect INETs/ENETs with dissimilar IP protocol versions may need to employ a network address and protocol translation function such as NAT64 [RFC6146].

3.20. Proxy/Server-Gateway Bidirectional Forwarding Detection

In environments where rapid failure recovery is required, Proxy/Servers and Gateways SHOULD use Bidirectional Forwarding Detection (BFD) [RFC5880]. Nodes that use BFD can quickly detect and react to failures so that cached information is re-established through alternate nodes. BFD control messaging is carried only over well-connected ground domain networks (i.e., and not low-end radio links) and can therefore be tuned for rapid response.

Proxy/Servers and Gateways maintain BFD sessions in parallel with their BGP peerings. If a Proxy/Server or Gateway fails, BGP peers will quickly re-establish routes through alternate paths the same as for common BGP deployments. Similarly, Proxys maintain BFD sessions with their associated Gateways even though they do not establish BGP peerings with them.

3.21. Time-Varying MNPs

In some use cases, it is desirable, beneficial and efficient for the Client to receive a constant MNP that travels with the Client wherever it moves. For example, this would allow air traffic controllers to easily track aircraft, etc. In other cases, however (e.g., intelligent transportation systems), the MN may be willing to sacrifice a modicum of efficiency in order to have time-varying MNPs that can be changed every so often to defeat adversarial tracking.

The DHCPv6 service offers a way for Clients that desire time-varying MNPs to obtain short-lived prefixes (e.g., on the order of a small number of minutes). In that case, the identity of the Client would not be bound to the MNP but rather to a Node Identification value (see: [I-D.templin-6man-omni]) to be used as the Client ID seed for MNP prefix delegation. The Client would then be obligated to renumber its internal networks whenever its MNP (and therefore also its XLA) changes. This should not present a challenge for Clients with automated network renumbering services, however presents limits for the durations of ongoing sessions that would prefer to use a constant address.

4. Implementation Status

An early AERO implementation based on OpenVPN (<https://openvpn.net/>) was announced on the v6ops mailing list on January 10, 2018 and an initial public release of the AERO proof-of-concept source code was announced on the intarea mailing list on August 21, 2015.

Many AERO/OMNI functions are implemented and undergoing final integration. OAL fragmentation/reassembly buffer management code has been cleared for public release.

5. IANA Considerations

The IANA has assigned the UDP port number "8060" for an earlier experimental first version of AERO [RFC6706]. This document together with [I-D.templin-6man-omni] reclaims UDP port number "8060" as the service port for UDP/IP encapsulation. This document makes no request of IANA, since [I-D.templin-6man-omni] already provides instructions. (Note: although [RFC6706] was not widely implemented or deployed, it need not be obsoleted since its messages use the invalid ICMPv6 message type number '0' which implementations of this specification can easily distinguish and ignore.)

No further IANA actions are required.

6. Security Considerations

AERO Gateways configure secured tunnels with AERO Proxy/Servers and Relays within their local OMNI link segments. Applicable secured tunnel alternatives include IPsec [RFC4301], TLS/SSL [RFC8446], DTLS [RFC6347], WireGuard [WG], etc. The AERO Gateways of all OMNI link segments in turn configure secured tunnels for their neighboring AERO Gateways in a secured spanning tree topology. Therefore, control messages exchanged between any pair of OMNI link neighbors over the secured spanning tree are already protected.

To prevent spoofing vectors, Proxy/Servers MUST discard without responding to any unsecured NS/NA(AR) messages. Also, Proxy/Servers MUST discard without forwarding any original IP packets received from one of their own Clients (whether directly or following OAL reassembly) with a source address that does not match the Client's MNP and/or a destination address that does match the Client's MNP. Finally, Proxy/Servers MUST discard without forwarding any carrier packets with an OAL source and destination that both match the same MNP.

For INET partitions that require strong security in the data plane, two options for securing communications include 1) disable route optimization so that all traffic is conveyed over secured tunnels, or 2) enable on-demand secure tunnel creation between Client neighbors. Option 1) would result in longer routes than necessary and impose traffic concentration on critical infrastructure elements. Option 2) could be coordinated between Clients using NS/NA messages with OMNI Host Identity Protocol (HIP) "Initiator/Responder" message sub-options [RFC7401][I-D.templin-6man-omni] to create a secured tunnel on-demand, or to use the QUIC-TLS protocol to establish a secured connection [RFC9000][RFC9001][RFC9002].

AERO Clients that connect to secured ANETs need not apply security to their IPv6 ND messages, since the messages will be authenticated and forwarded by a perimeter Proxy/Server that applies security on its INET-facing interface as part of the secured spanning tree (see above). AERO Clients connected to the open INET can use network and/or transport layer security services such as VPNs or can by some other means establish a direct link to a Proxy/Server. When a VPN or direct link may be impractical, however, INET Clients and Proxy/Servers SHOULD include and verify authentication signatures for their IPv6 ND messages as specified in [I-D.templin-6man-omni].

Application endpoints SHOULD use transport-layer (or higher-layer) security services such as QUIC-TLS, TLS/SSL, DTLS or SSH [RFC4251] to assure the same level of protection as for critical secured Internet services. AERO Clients that require host-based VPN services SHOULD use network and/or transport layer security services such as IPsec, TLS/SSL, DTLS, etc. AERO Proxys and Proxy/Servers can also provide a network-based VPN service on behalf of the Client, e.g., if the Client is located within a secured enclave and cannot establish a VPN on its own behalf.

AERO Proxy/Servers and Gateways present targets for traffic amplification Denial of Service (DoS) attacks. This concern is no different than for widely-deployed VPN security gateways in the Internet, where attackers could send spoofed packets to the gateways at high data rates. This can be mitigated through the AERO/OMNI data origin authentication procedures, as well as connecting Proxy/Servers and Gateways over dedicated links with no connections to the Internet and/or when connections to the Internet are only permitted through well-managed firewalls. Traffic amplification DoS attacks can also target an AERO Client's low data rate links. This is a concern not only for Clients located on the open Internet but also for Clients in secured enclaves. AERO Proxy/Servers and Proxys can institute rate limits that protect Clients from receiving packet floods that could DoS low data rate links.

AERO Relays must implement ingress filtering to avoid a spoofing attack in which spurious messages with ULA addresses are injected into an OMNI link from an outside attacker. AERO Clients MUST ensure that their connectivity is not used by unauthorized nodes on their ENETs to gain access to a protected network, i.e., AERO Clients that act as routers MUST NOT provide routing services for unauthorized nodes. (This concern is no different than for ordinary hosts that receive an IP address delegation but then "share" the address with other nodes via some form of Internet connection sharing such as tethering.)

The PRL MUST be well-managed and secured from unauthorized tampering, even though the list contains only public information. The PRL can be conveyed to the Client in a similar fashion as in [RFC5214] (e.g., through layer 2 data link login messaging, secure upload of a static file, DNS lookups, etc.).

The AERO service for open INET Clients depends on a public key distribution service in which Client public keys and identities are maintained in a shared database accessible to all open INET Proxy/Servers. Similarly, each Client must be able to determine the public key of each Proxy/Server, e.g. by consulting an online database. When AERO nodes register their public keys indexed by a unique Host Identity Tag (HIT) [RFC7401] in a distributed database such as the DNS, and use the HIT as an identity for applying IPv6 ND message authentication signatures, a means for determining public key attestation is available.

Security considerations for IPv6 fragmentation and reassembly are discussed in [I-D.templin-6man-omni]. In environments where spoofing is considered a threat, OMNI nodes SHOULD employ Identification window synchronization and OAL destinations SHOULD configure an (end-system-based) firewall.

SRH authentication facilities are specified in [RFC8754]. Security considerations for accepting link-layer ICMP messages and reflected packets are discussed throughout the document.

7. Acknowledgements

Discussions in the IETF, aviation standards communities and private exchanges helped shape some of the concepts in this work. Individuals who contributed insights include Mikael Abrahamsson, Mark Andrews, Fred Baker, Bob Braden, Stewart Bryant, Scott Burleigh, Brian Carpenter, Wojciech Dec, Pavel Drasil, Ralph Droms, Adrian Farrel, Nick Green, Sri Gundavelli, Brian Haberman, Bernhard Haindl, Joel Halpern, Tom Herbert, Bob Hinden, Sascha Hlusiak, Lee Howard, Christian Huitema, Zdenek Jaron, Andre Kostur, Hubert Kuenig, Eliot

Lear, Ted Lemon, Andy Malis, Satoru Matsushima, Tomek Mrugalski, Thomas Narten, Madhu Niraula, Alexandru Petrescu, Behcet Saikaya, Michal Skorepa, Dave Thaler, Joe Touch, Bernie Volz, Ryuji Wakikawa, Tony Whyman, Lloyd Wood and James Woodyatt. Members of the IESG also provided valuable input during their review process that greatly improved the document. Special thanks go to Stewart Bryant, Joel Halpern and Brian Haberman for their shepherding guidance during the publication of the AERO first edition.

This work has further been encouraged and supported by Boeing colleagues including Akash Agarwal, Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam Brodie, John Bush, Balaguruna Chidambaram, Irene Chin, Bruce Cornish, Claudiu Danilov, Don Dillenburg, Joe Dudkowski, Wen Fang, Samad Farooqui, Anthony Gregory, Jeff Holland, Seth Jahne, Brian Jaury, Greg Kimberly, Ed King, Madhuri Madhava Badgandi, Laurel Matthew, Gene MacLean III, Kyle Mikos, Rob Muszkiewicz, Sean O'Sullivan, Satish Raghavendran, Vijay Rajagopalan, Greg Saccone, Bhargava Raman Sai Prakash, Rod Santiago, Madhanmohan Savadamuthu, Kent Shuey, Brian Skeen, Mike Slane, Carrie Spiker, Katie Tran, Brendan Williams, Amelia Wilson, Julie Wulff, Yueli Yang, Eric Yeh and other members of the Boeing mobility, networking and autonomy teams. Akash Agarwal, Kyle Bae, Wayne Benson, Madhuri Madhava Badgandi, Vijayasaratthy Rajagopalan, Bhargava Raman Sai Prakash, Katie Tran and Eric Yeh are especially acknowledged for their work on the AERO implementation. Chuck Klabunde is honored and remembered for his early leadership, and we mourn his untimely loss.

This work was inspired by the support and encouragement of countless outstanding colleagues, managers and program directors over the span of many decades. Beginning in the late 1980s, the Digital Equipment Corporation (DEC) Ultrix Engineering and DECnet Architects groups identified early issues with fragmentation and bridging links with diverse MTUs. In the early 1990s, engagements at DEC Project Sequoia at UC Berkeley and the DEC Western Research Lab in Palo Alto included investigations into large-scale networked filesystems, ATM vs Internet and network security proxys. In the mid-1990s to early 2000s employment at the NASA Ames Research Center (Sterling Software) and SRI International supported early investigations of IPv6, ONR UAV Communications and the IETF. An employment at Nokia where important IETF documents were published gave way to a present-day engagement with The Boeing Company. The work matured at Boeing through major programs including Future Combat Systems, Advanced Airplane Program, DTN for the International Space Station, Mobility Vision Lab, CAST, Caravan, Airplane Internet of Things, the NASA UAS/CNS program, the FAA/ICAO ATN/IPS program and many others. An attempt to name all who gave support and encouragement would double the current document size and result in many unintentional omissions - but to all a humble thanks.

Earlier works on NBMA tunneling approaches are found in [RFC2529][RFC5214][RFC5569].

Many of the constructs presented in this second edition of AERO are based on the author's earlier works, including:

- * The Internet Routing Overlay Network (IRON)
[RFC6179][I-D.templin-ironbis]
- * Virtual Enterprise Traversal (VET)
[RFC5558][I-D.templin-intarea-vet]
- * The Subnetwork Encapsulation and Adaptation Layer (SEAL)
[RFC5320][I-D.templin-intarea-seal]
- * AERO, First Edition [RFC6706]

Note that these works cite numerous earlier efforts that are not also cited here due to space limitations. The authors of those earlier works are acknowledged for their insights.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the Boeing Commercial Airplanes (BCA) Internet of Things (IoT) and autonomy programs.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

8. References

8.1. Normative References

- [I-D.templin-6man-omni]
Templin, F. L., "Transmission of IP Packets over Overlay Multilink Network (OMNI) Interfaces", Work in Progress, Internet-Draft, draft-templin-6man-omni-60, 22 April 2022, <<https://www.ietf.org/archive/id/draft-templin-6man-omni-60.txt>>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <<https://www.rfc-editor.org/info/rfc3972>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, DOI 10.17487/RFC6081, January 2011, <<https://www.rfc-editor.org/info/rfc6081>>.

- [RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <<https://www.rfc-editor.org/info/rfc7401>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<https://www.rfc-editor.org/info/rfc7739>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.

8.2. Informative References

- [BGP] Huston, G., "BGP in 2015, <http://potaroo.net>", January 2016.
- [I-D.bonica-6man-comp-rtg-hdr] Bonica, R., Kamite, Y., Alston, A., Henriques, D., and L. Jalil, "The IPv6 Compact Routing Header (CRH)", Work in Progress, Internet-Draft, draft-bonica-6man-comp-rtg-hdr-27, 15 November 2021, <<https://www.ietf.org/archive/id/draft-bonica-6man-comp-rtg-hdr-27.txt>>.
- [I-D.bonica-6man-crh-helper-opt] Li, X., Bao, C., Ruan, E., and R. Bonica, "Compressed Routing Header (CRH) Helper Option", Work in Progress, Internet-Draft, draft-bonica-6man-crh-helper-opt-04, 11 October 2021, <<https://www.ietf.org/archive/id/draft-bonica-6man-crh-helper-opt-04.txt>>.

[I-D.ietf-intarea-frag-fragile]

Bonica, R., Baker, F., Huston, G., Hinden, R. M., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", Work in Progress, Internet-Draft, draft-ietf-intarea-frag-fragile-17, 30 September 2019, <<https://www.ietf.org/archive/id/draft-ietf-intarea-frag-fragile-17.txt>>.

[I-D.ietf-intarea-tunnels]

Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-10, 12 September 2019, <<https://www.ietf.org/archive/id/draft-ietf-intarea-tunnels-10.txt>>.

[I-D.ietf-ipwave-vehicular-networking]

Jeong, J. (., "IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-ietf-ipwave-vehicular-networking-28, 30 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-ipwave-vehicular-networking-28.txt>>.

[I-D.ietf-rtgwg-atn-bgp]

Templin, F. L., Saccone, G., Dawra, G., Lindem, A., and V. Moreno, "A Simple BGP-based Mobile Routing System for the Aeronautical Telecommunications Network", Work in Progress, Internet-Draft, draft-ietf-rtgwg-atn-bgp-17, 19 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-rtgwg-atn-bgp-17.txt>>.

[I-D.templin-6man-dhcpv6-ndopt]

Templin, F. L., "A Unified Stateful/Stateless Configuration Service for IPv6", Work in Progress, Internet-Draft, draft-templin-6man-dhcpv6-ndopt-11, 1 January 2021, <<https://www.ietf.org/archive/id/draft-templin-6man-dhcpv6-ndopt-11.txt>>.

[I-D.templin-intarea-seal]

Templin, F. L., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)", Work in Progress, Internet-Draft, draft-templin-intarea-seal-68, 3 January 2014, <<https://www.ietf.org/archive/id/draft-templin-intarea-seal-68.txt>>.

- [I-D.templin-intarea-vet]
Templin, F. L., "Virtual Enterprise Traversal (VET)", Work in Progress, Internet-Draft, draft-templin-intarea-vet-40, 3 May 2013, <<https://www.ietf.org/archive/id/draft-templin-intarea-vet-40.txt>>.
- [I-D.templin-ipwave-uam-its]
Templin, F. L., "Urban Air Mobility Implications for Intelligent Transportation Systems", Work in Progress, Internet-Draft, draft-templin-ipwave-uam-its-04, 4 January 2021, <<https://www.ietf.org/archive/id/draft-templin-ipwave-uam-its-04.txt>>.
- [I-D.templin-ironbis]
Templin, F. L., "The Interior Routing Overlay Network (IRON)", Work in Progress, Internet-Draft, draft-templin-ironbis-16, 28 March 2014, <<https://www.ietf.org/archive/id/draft-templin-ironbis-16.txt>>.
- [I-D.templin-v6ops-pdhost]
Templin, F. L., "IPv6 Prefix Delegation and Multi-Addressing Models", Work in Progress, Internet-Draft, draft-templin-v6ops-pdhost-27, 1 January 2021, <<https://www.ietf.org/archive/id/draft-templin-v6ops-pdhost-27.txt>>.
- [IEN48] Cerf, V., "The Catenet Model For Internetworking, <https://www.rfc-editor.org/ien/ien48.txt>", July 1978.
- [IEN48-2] Cerf, V., "The Catenet Model For Internetworking (with figures), <http://www.postel.org/ien/pdf/ien048.pdf>", July 1978.
- [OVPN] OpenVPN, O., "<http://openvpn.net>", October 2016.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <<https://www.rfc-editor.org/info/rfc1812>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.

- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.
- [RFC2004] Perkins, C., "Minimal Encapsulation within IP", RFC 2004, DOI 10.17487/RFC2004, October 1996, <<https://www.rfc-editor.org/info/rfc2004>>.
- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, DOI 10.17487/RFC2236, November 1997, <<https://www.rfc-editor.org/info/rfc2236>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<https://www.rfc-editor.org/info/rfc2464>>.
- [RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, DOI 10.17487/RFC2529, March 1999, <<https://www.rfc-editor.org/info/rfc2529>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <<https://www.rfc-editor.org/info/rfc3330>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <<https://www.rfc-editor.org/info/rfc4251>>.

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006, <<https://www.rfc-editor.org/info/rfc4511>>.
- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <<https://www.rfc-editor.org/info/rfc4605>>.
- [RFC4982] Bagnulo, M. and J. Arkko, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)", RFC 4982, DOI 10.17487/RFC4982, July 2007, <<https://www.rfc-editor.org/info/rfc4982>>.

- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <<https://www.rfc-editor.org/info/rfc5015>>.
- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <<https://www.rfc-editor.org/info/rfc5214>>.
- [RFC5320] Templin, F., Ed., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)", RFC 5320, DOI 10.17487/RFC5320, February 2010, <<https://www.rfc-editor.org/info/rfc5320>>.
- [RFC5522] Eddy, W., Ivancic, W., and T. Davis, "Network Mobility Route Optimization Requirements for Operational Use in Aeronautics and Space Exploration Mobile Networks", RFC 5522, DOI 10.17487/RFC5522, October 2009, <<https://www.rfc-editor.org/info/rfc5522>>.
- [RFC5558] Templin, F., Ed., "Virtual Enterprise Traversal (VET)", RFC 5558, DOI 10.17487/RFC5558, February 2010, <<https://www.rfc-editor.org/info/rfc5558>>.
- [RFC5569] Despres, R., "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)", RFC 5569, DOI 10.17487/RFC5569, January 2010, <<https://www.rfc-editor.org/info/rfc5569>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <<https://www.rfc-editor.org/info/rfc6106>>.
- [RFC6139] Russert, S., Ed., Fleischman, E., Ed., and F. Templin, Ed., "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER) Scenarios", RFC 6139, DOI 10.17487/RFC6139, February 2011, <<https://www.rfc-editor.org/info/rfc6139>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.

- [RFC6179] Templin, F., Ed., "The Internet Routing Overlay Network (IRON)", RFC 6179, DOI 10.17487/RFC6179, March 2011, <<https://www.rfc-editor.org/info/rfc6179>>.
- [RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <<https://www.rfc-editor.org/info/rfc6221>>.
- [RFC6273] Kuvec, A., Krishnan, S., and S. Jiang, "The Secure Neighbor Discovery (SEND) Hash Threat Analysis", RFC 6273, DOI 10.17487/RFC6273, June 2011, <<https://www.rfc-editor.org/info/rfc6273>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <<https://www.rfc-editor.org/info/rfc6355>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <<https://www.rfc-editor.org/info/rfc6706>>.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <<https://www.rfc-editor.org/info/rfc6935>>.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <<https://www.rfc-editor.org/info/rfc7333>>.

- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC8402] Filtsch, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8754] Filtsch, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <<https://www.rfc-editor.org/info/rfc9001>>.
- [RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <<https://www.rfc-editor.org/info/rfc9002>>.
- [WG] Wireguard, "WireGuard, <https://www.wireguard.com>", August 2020.

Appendix A. Non-Normative Considerations

AERO can be applied to a multitude of Internetworking scenarios, with each having its own adaptations. The following considerations are provided as non-normative guidance:

A.1. Implementation Strategies for Route Optimization

Route optimization as discussed in Section 3.13 results in the creation of NCEs. The NCE state is set to REACHABLE for at most ReachableTime seconds. In order to refresh the NCE lifetime before the ReachableTime timer expires, the specification requires implementations to issue a new NS/NA(AR) exchange to reset ReachableTime while data packets are still flowing. However, the decision of when to initiate a new NS/NA(AR) exchange and to perpetuate the process is left as an implementation detail.

One possible strategy may be to monitor the NCE watching for data packets for (ReachableTime - 5) seconds. If any data packets have been sent to the neighbor within this timeframe, then send an NS(AR) to receive a new NA(AR). If no data packets have been sent, wait for 5 additional seconds and send an immediate NS(AR) if any data packets are sent within this "expiration pending" 5 second window. If no additional data packets are sent within the 5 second window, reset the NCE state to STALE.

The monitoring of the neighbor data packet traffic therefore becomes an ongoing process during the NCE lifetime. If the NCE expires, future data packets will trigger a new NS/NA(AR) exchange while the packets themselves are delivered over a longer path until route optimization state is re-established.

A.2. Implicit Mobility Management

OMNI interface neighbors MAY provide a configuration option that allows them to perform implicit mobility management in which no IPv6 ND messaging is used. In that case, the Client only transmits packets over a single interface at a time, and the neighbor always observes packets arriving from the Client from the same link-layer source address.

If the Client's underlay interface address changes (either due to a readdressing of the original interface or switching to a new interface) the neighbor immediately updates the NCE for the Client and begins accepting and sending packets according to the Client's new address. This implicit mobility method applies to use cases such as cellphones with both WiFi and Cellular interfaces where only one of the interfaces is active at a given time, and the Client automatically switches over to the backup interface if the primary interface fails.

A.3. Direct Underlying Interfaces

When a Client's OMNI interface is configured over a Direct interface, the neighbor at the other end of the Direct link can receive packets without any encapsulation. In that case, the Client sends packets over the Direct link according to traffic selectors. If the Direct interface is selected, then the Client's IP packets are transmitted directly to the peer without going through an ANET/INET. If other interfaces are selected, then the Client's IP packets are transmitted via a different interface, which may result in the inclusion of Proxy/Servers and Gateways in the communications path. Direct interfaces must be tested periodically for reachability, e.g., via NUD.

A.4. AERO Critical Infrastructure Considerations

AERO Gateways can be either Commercial off-the Shelf (COTS) standard IP routers or virtual machines in the cloud. Gateways must be provisioned, supported and managed by the INET administrative authority, and connected to the Gateways of other INETs via inter-domain peerings. Cost for purchasing, configuring and managing Gateways is nominal even for very large OMNI links.

AERO INET Proxy/Servers can be standard dedicated server platforms, but most often will be deployed as virtual machines in the cloud. The only requirements for INET Proxy/Servers are that they can run the AERO/OMNI code and have at least one network interface connection to the INET. INET Proxy/Servers must be provisioned, supported and managed by the INET administrative authority. Cost for purchasing, configuring and managing cloud Proxy/Servers is nominal especially for virtual machines.

AERO ANET Proxy/Servers are most often standard dedicated server platforms with one underlay interface connected to the ANET and a second interface connected to an INET. As with INET Proxy/Servers, the only requirements are that they can run the AERO/OMNI code and have at least one interface connection to the INET. ANET Proxy/Servers must be provisioned, supported and managed by the ANET administrative authority. Cost for purchasing, configuring and managing Proxys is nominal, and borne by the ANET administrative authority.

AERO Relays are simply Proxy/Servers connected to INETs and/or ENETs that provide forwarding services for non-MNP destinations. The Relay connects to the OMNI link and engages in eBGP peering with one or more Gateways as a stub AS. The Relay then injects its MNPs and/or non-MNP prefixes into the BGP routing system, and provisions the prefixes to its downstream-attached networks. The Relay can perform ROS/ROR services the same as for any Proxy/Server, and can route between the MNP and non-MNP address spaces.

A.5. AERO Server Failure Implications

AERO Proxy/Servers may appear as a single point of failure in the architecture, but such is not the case since all Proxy/Servers on the link provide identical services and loss of a Proxy/Server does not imply immediate and/or comprehensive communication failures. Proxy/Server failure is quickly detected and conveyed by Bidirectional Forward Detection (BFD) and/or proactive NUD allowing Clients to migrate to new Proxy/Servers.

If a Proxy/Server fails, ongoing packet forwarding to Clients will continue by virtue of the neighbor cache entries that have already been established in route optimization sources (ROs). If a Client also experiences mobility events at roughly the same time the Proxy/Server fails, uNA messages may be lost but neighbor cache entries in the DEPARTED state will ensure that packet forwarding to the Client's new locations will continue for up to DepartTime seconds.

If a Client is left without a Proxy/Server for a considerable length of time (e.g., greater than ReachableTime seconds) then existing neighbor cache entries will eventually expire and both ongoing and new communications will fail. The original source will continue to retransmit until the Client has established a new Proxy/Server relationship, after which time continuous communications will resume.

Therefore, providing many Proxy/Servers on the link with high availability profiles provides resilience against loss of individual Proxy/Servers and assurance that Clients can establish new Proxy/Server relationships quickly in event of a Proxy/Server failure.

A.6. AERO Client / Server Architecture

The AERO architectural model is client / server in the control plane, with route optimization in the data plane. The same as for common Internet services, the AERO Client discovers the addresses of AERO Proxy/Servers and connects to one or more of them. The AERO service is analogous to common Internet services such as google.com, yahoo.com, cnn.com, etc. However, there is only one AERO service for the link and all Proxy/Servers provide identical services.

Common Internet services provide differing strategies for advertising server addresses to clients. The strategy is conveyed through the DNS resource records returned in response to name resolution queries. As of January 2020 Internet-based 'nslookup' services were used to determine the following:

- * When a client resolves the domainname "google.com", the DNS always returns one A record (i.e., an IPv4 address) and one AAAA record (i.e., an IPv6 address). The client receives the same addresses each time it resolves the domainname via the same DNS resolver, but may receive different addresses when it resolves the domainname via different DNS resolvers. But, in each case, exactly one A and one AAAA record are returned.
- * When a client resolves the domainname "ietf.org", the DNS always returns one A record and one AAAA record with the same addresses regardless of which DNS resolver is used.
- * When a client resolves the domainname "yahoo.com", the DNS always returns a list of 4 A records and 4 AAAA records. Each time the client resolves the domainname via the same DNS resolver, the same list of addresses are returned but in randomized order (i.e., consistent with a DNS round-robin strategy). But, interestingly, the same addresses are returned (albeit in randomized order) when the domainname is resolved via different DNS resolvers.
- * When a client resolves the domainname "amazon.com", the DNS always returns a list of 3 A records and no AAAA records. As with "yahoo.com", the same three A records are returned from any worldwide Internet connection point in randomized order.

The above example strategies show differing approaches to Internet resilience and service distribution offered by major Internet services. The Google approach exposes only a single IPv4 and a single IPv6 address to clients. Clients can then select whichever IP protocol version offers the best response, but will always use the same IP address according to the current Internet connection point. This means that the IP address offered by the network must lead to a highly-available server and/or service distribution point. In other words, resilience is predicated on high availability within the network and with no client-initiated failovers expected (i.e., it is all-or-nothing from the client's perspective). However, Google does provide for worldwide distributed service distribution by virtue of the fact that each Internet connection point responds with a different IPv6 and IPv4 address. The IETF approach is like google (all-or-nothing from the client's perspective), but provides only a single IPv4 or IPv6 address on a worldwide basis. This means that the addresses must be made highly-available at the network level with

no client failover possibility, and if there is any worldwide service distribution it would need to be conducted by a network element that is reached via the IP address acting as a service distribution point.

In contrast to the Google and IETF philosophies, Yahoo and Amazon both provide clients with a (short) list of IP addresses with Yahoo providing both IP protocol versions and Amazon as IPv4-only. The order of the list is randomized with each name service query response, with the effect of round-robin load balancing for service distribution. With a short list of addresses, there is still expectation that the network will implement high availability for each address but in case any single address fails the client can switch over to using a different address. The balance then becomes one of function in the network vs function in the end system.

The same implications observed for common highly-available services in the Internet apply also to the AERO client/server architecture. When an AERO Client connects to one or more ANETs, it discovers one or more AERO Proxy/Server addresses through the mechanisms discussed in earlier sections. Each Proxy/Server address presumably leads to a fault-tolerant clustering arrangement such as supported by Linux-HA, Extended Virtual Synchrony or Paxos. Such an arrangement has precedence in common Internet service deployments in lightweight virtual machines without requiring expensive hardware deployment. Similarly, common Internet service deployments set service IP addresses on service distribution points that may relay requests to many different servers.

For AERO, the expectation is that a combination of the Google/IETF and Yahoo/Amazon philosophies would be employed. The AERO Client connects to different ANET access points and can receive 1-2 Proxy/Server ULAs at each point. It then selects one AERO Proxy/Server address, and engages in RS/RA exchanges with the same Proxy/Server from all ANET connections. The Client remains with this Proxy/Server unless or until the Proxy/Server fails, in which case it can switch over to an alternate Proxy/Server. The Client can likewise switch over to a different Proxy/Server at any time if there is some reason for it to do so. So, the AERO expectation is for a balance of function in the network and end system, with fault tolerance and resilience at both levels.

Appendix B. Change Log

<< RFC Editor - remove prior to publication >>

Changes from earlier versions:

- * Submit for RFC publication.

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
United States of America
Email: fltemplin@acm.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 27 October 2022

F. L. Templin, Ed.
The Boeing Company
25 April 2022

Transmission of IP Packets over Overlay Multilink Network (OMNI)
Interfaces
draft-templin-6man-omni-61

Abstract

Mobile nodes (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, space systems, enterprise wireless devices, pedestrians with cell phones, etc.) communicate with networked correspondents over multiple access network data links and configure mobile routers to connect end user networks. A multilink virtual interface specification is presented that enables mobile nodes to coordinate with a network-based mobility service and/or with other mobile node peers. The virtual interface provides an adaptation layer service that also applies for more static deployments such as enterprise and home networks. This document specifies the transmission of IP packets over Overlay Multilink Network (OMNI) Interfaces.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
2. Terminology	7
3. Requirements	15
4. Overlay Multilink Network (OMNI) Interface Model	15
5. OMNI Interface Maximum Transmission Unit (MTU)	22
5.1. Jumbograms	23
5.2. IPv6 Parcels	24
6. The OMNI Adaptation Layer (OAL)	24
6.1. OAL Source Encapsulation and Fragmentation	25
6.2. OAL L2 Encapsulation and Re-Encapsulation	30
6.3. OAL L2 Decapsulation and Reassembly	33
6.4. OAL Header Compression	34
6.5. OAL-in-OAL Encapsulation	38
6.6. OAL Identification Window Maintenance	40
6.7. OAL Fragment Retransmission	45
6.8. OAL MTU Feedback Messaging	46
6.9. OAL Super-Packets	48
6.10. OAL Bubbles	49
6.11. OAL Requirements	50
6.12. OAL Fragmentation Security Implications	51
6.13. OMNI Hosts	52
6.14. IP Parcels	55
7. Frame Format	58
8. Link-Local Addresses (LLAs)	59
9. Unique-Local Addresses (ULAs)	60
10. Global Unicast Addresses (GUAs)	63
11. Node Identification	64
12. Address Mapping - Unicast	65
12.1. The OMNI Option	66
12.2. OMNI Sub-Options	66
12.2.1. Pad1	69
12.2.2. PadN	69
12.2.3. Neighbor Coordination	70
12.2.4. Interface Attributes	72
12.2.5. Multilink Forwarding Parameters	75
12.2.6. Traffic Selector	80
12.2.7. Geo Coordinates	81

12.2.8.	Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Message	82
12.2.9.	Host Identity Protocol (HIP) Message	83
12.2.10.	PIM-SM Message	85
12.2.11.	Fragmentation Report (FRAGREP)	86
12.2.12.	Node Identification	87
12.2.13.	ICMPv6 Error	89
12.2.14.	QUIC-TLS Message	90
12.2.15.	Proxy/Server Departure	90
12.2.16.	Sub-Type Extension	91
13.	Address Mapping - Multicast	94
14.	Multilink Conceptual Sending Algorithm	95
14.1.	Multiple OMNI Interfaces	95
14.2.	Client-Proxy/Server Loop Prevention	96
15.	Router Discovery and Prefix Registration	96
15.1.	Window Synchronization	105
15.2.	Router Discovery in IP Multihop and IPv4-Only Networks	106
15.3.	DHCPv6-based Prefix Registration	108
15.4.	OMNI Link Extension	110
16.	Secure Redirection	110
17.	Proxy/Server Resilience	111
18.	Detecting and Responding to Proxy/Server Failures	111
19.	Transition Considerations	112
20.	OMNI Interfaces on Open Internetworks	113
21.	Time-Varying MNPs	115
22.	(H)HITs and Temporary ULA (TLA)s	116
23.	Address Selection	117
24.	Error Messages	118
25.	IANA Considerations	118
25.1.	"Protocol Numbers" Registry	118
25.2.	"IEEE 802 Numbers" Registry	118
25.3.	"IPv4 Special-Purpose Address" Registry	118
25.4.	"IPv6 Neighbor Discovery Option Formats" Registry	119
25.5.	"Ethernet Numbers" Registry	119
25.6.	"ICMPv6 Code Fields: Type 2 - Packet Too Big" Registry	119
25.7.	"OMNI Option Sub-Type Values" (New Registry)	119
25.8.	"OMNI Geo Coordinates Type Values" (New Registry)	120
25.9.	"OMNI Node Identification ID-Type Values" (New Registry)	120
25.10.	"OMNI Option Sub-Type Extension Values" (New Registry)	121
25.11.	"OMNI RFC4380 UDP/IP Header Option" (New Registry)	121
25.12.	"OMNI RFC6081 UDP/IP Trailer Option" (New Registry)	122
25.13.	Additional Considerations	122
26.	Security Considerations	123
27.	Implementation Status	124

28. Document Updates	124
29. Acknowledgements	124
30. References	126
30.1. Normative References	126
30.2. Informative References	128
Appendix A. OAL Checksum Algorithm	137
Appendix B. IPv6 ND Message Authentication and Integrity	137
Appendix C. VDL Mode 2 Considerations	138
Appendix D. Client-Proxy/Server Isolation Through Link-Layer Address Mapping	139
Appendix E. Change Log	140
Author's Address	140

1. Introduction

Mobile nodes (e.g., aircraft of various configurations, terrestrial vehicles, seagoing vessels, space systems, enterprise wireless devices, pedestrians with cellphones, etc.) configure mobile routers with multiple interface connections to wireless and/or wired-line data links. These data links may have diverse performance, cost and availability properties that can change dynamically according to mobility patterns, flight phases, proximity to infrastructure, etc. The mobile router acts as a Client of a network-based Mobility Service (MS) by configuring a virtual interface over its underlay interface data link connections to support the "6M's of modern Internetworking" (see below).

Each Client configures a virtual interface (termed the "Overlay Multilink Network Interface (OMNI)") as a thin layer over its underlay network interfaces (which may themselves connect to virtual or physical links). The OMNI interface is therefore the only interface abstraction exposed to the IP layer and behaves according to the Non-Broadcast, Multiple Access (NBMA) interface principle, while underlay interfaces appear as link layer communication channels in the architecture. The OMNI interface internally employs the "OMNI Adaptation Layer (OAL)" to ensure that original IP packets are adapted to diverse underlay interfaces with heterogeneous properties. The OMNI interface connects to a virtual overlay known as the "OMNI link". The OMNI link multinet service spans one or more Internetworks that may include private-use infrastructures (e.g., enterprise networks) and/or the global public Internet itself.

Client OMNI interfaces interact with the MS and/or other OMNI nodes through IPv6 Neighbor Discovery (ND) control message exchanges [RFC4861]. The MS consists of a distributed set of service nodes (including Proxy/Servers and other infrastructure elements) that also configure OMNI interfaces. Automatic Extended Route Optimization (AERO) in particular provides a companion MS compatible with the OMNI

architecture [I-D.templin-6man-aero]. AERO discusses details of ND message based route optimization, mobility management, and multinet traversal while the fundamental aspects of OMNI link operation are discussed in this document.

Each OMNI interface provides a multilink nexus for exchanging inbound and outbound traffic via selected underlay interface(s). The IP layer sees the OMNI interface as a point of connection to the OMNI link. Each OMNI link has one or more associated Mobility Service Prefixes (MSPs), which are typically IP Global Unicast Address (GUA) prefixes assigned to the link and from which Mobile Network Prefixes (MNPs) are derived. If there are multiple OMNI links, the IP layer will see multiple OMNI interfaces.

Each Client receives an MNP through IPv6 ND control message exchanges with Proxy/Servers over Access Networks (ANETs) and/or open Internetworks (INETs). The Client sub-delegates the MNP to downstream-attached End-user Networks (ENETs) independently of the underlay interfaces selected for data transport. The Client acts as a fixed or mobile router on behalf of peers on its ENETs, and uses OMNI interface control messaging to coordinate with Hosts, Proxy/Servers and/or other Clients. The Client iterates its control messaging over each of the OMNI interface's ANET/INET underlay interfaces in order to register each interface with the MS (see Section 15). The Client can also provide Proxy/Server-like services for a recursively nested chain of other Clients located in downstream-attached ENETs.

Clients may connect to multiple distinct OMNI links within the same OMNI domain by configuring multiple OMNI interfaces, e.g., omni0, omni1, omni2, etc. Each OMNI interface is configured over a set of underlay interfaces and provides a nexus for Safety-Based Multilink (SBM) operation. The IP layer applies SBM routing to select a specific OMNI interface, then the selected OMNI interface applies Performance-Based Multilink (PBM) internally to select appropriate underlay interfaces. Applications select SBM topologies based on IP layer Segment Routing [RFC8402], while each OMNI interface orchestrates PBM internally based on OMNI layer Segment Routing.

OMNI provides a link model suitable for a wide range of use cases. For example, the International Civil Aviation Organization (ICAO) Working Group-I Mobility Subgroup is developing a future Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS) and has issued a liaison statement requesting IETF adoption [ATN] in support of ICAO Document 9896 [ATN-IPS]. The IETF IP Wireless Access in Vehicular Environments (ipwave) working group has further included problem statement and use case analysis for OMNI in a document now in AD evaluation for RFC publication

[I-D.ietf-ipwave-vehicular-networking]. Still other communities of interest include AEEC, RTCA Special Committee 228 (SC-228) and NASA programs that examine commercial aviation, Urban Air Mobility (UAM) and Unmanned Air Systems (UAS). Pedestrians with handheld devices represent another large class of potential OMNI users.

OMNI supports the "6M's of modern Internetworking" including:

1. Multilink - a Client's ability to coordinate multiple diverse underlay interfaces as a single logical unit (i.e., the OMNI interface) to achieve the required communications performance and reliability objectives.
2. Multinet - the ability to span the OMNI link over a segment routing topology with multiple diverse administrative domain network segments while maintaining seamless end-to-end communications between mobile Clients and correspondents such as air traffic controllers, fleet administrators, etc.
3. Mobility - a Client's ability to change network points of attachment (e.g., moving between wireless base stations) which may result in an underlay interface address change, but without disruptions to ongoing communication sessions with peers over the OMNI link.
4. Multicast - the ability to send a single network transmission that reaches multiple Clients belonging to the same interest group, but without disturbing other Clients not subscribed to the interest group.
5. Multihop - a mobile Client vehicle-to-vehicle relaying capability useful when multiple forwarding hops between vehicles may be necessary to "reach back" to an infrastructure access point connection to the OMNI link.
6. MTU assurance - the ability to deliver packets of various robust sizes between peers without loss due to a link size restriction, and to dynamically adjust packets sizes to achieve the optimal performance for each independent traffic flow.

This document specifies the transmission of IP packets and control messages over OMNI interfaces. The operation of both IP protocol versions (i.e., IPv4 [RFC0791] and IPv6 [RFC8200]) is specified as the network layer data plane, while OMNI interfaces use IPv6 ND messaging in the control plane independently of the data plane protocol(s). OMNI interfaces also provide an OAL based on encapsulation and fragmentation over heterogeneous underlay interfaces as an adaptation sublayer between L3 and L2. Both OMNI and the OAL are specified in detail throughout the remainder of this document.

2. Terminology

The terminology in the normative references applies; especially, the terms "link" and "interface" are the same as defined in the IPv6 [RFC8200] and IPv6 Neighbor Discovery (ND) [RFC4861] specifications. Additionally, this document assumes the following IPv6 ND message types: Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA) and Redirect. Hosts, Clients and Proxy/Servers that implement IPv6 ND maintain per-neighbor state in Neighbor Cache Entries (NCEs). Each NCE is indexed by the neighbor's network layer address(es) while the neighbor's OAL encapsulation address provides context for Identification verification.

The Protocol Constants defined in Section 10 of [RFC4861] are used in their same format and meaning in this document. The terms "All-Routers multicast", "All-Nodes multicast" and "Subnet-Router anycast" are the same as defined in [RFC4291] (with Link-Local scope assumed).

The term "IP" is used to refer collectively to either Internet Protocol version (i.e., IPv4 [RFC0791] or IPv6 [RFC8200]) when a specification at the layer in question applies equally to either version.

The following terms are defined within the scope of this document:

L2

The Data Link layer in the OSI network model. Also known as "layer-2", "link-layer", "sub-IP layer", etc.

L3

The Network layer in the OSI network model. Also known as "layer-3", "IP layer", etc.

Adaptation layer

A mid-layer that adapts L3 to a diverse collection of L2 underlay interfaces and their encapsulations. (No layer number is

assigned, since numbering was an artifact of the legacy reference model that need not carry forward in the modern architecture.) The adaptation layer sees the upper layer as "L3" and sees all lower layer encapsulations as "L2 encapsulations", which may include UDP, IP and true link-layer (e.g., Ethernet, etc.) headers.

Access Network (ANET)

a connected network region (e.g., an aviation radio access network, satellite service provider network, cellular operator network, WiFi network, etc.) that connects Clients to the Mobility Service. Physical and/or data link level security is assumed, and sometimes referred to as "protected spectrum". Private enterprise networks and ground domain aviation service networks may provide multiple secured IP hops between the Client's point of connection and the nearest Proxy/Server.

Internetwork (INET)

a connected network region with a coherent IP addressing plan that provides transit forwarding services between ANETs and/or OMNI nodes that coordinate with the Mobility Service over unprotected media. Since physical and/or data link level security cannot always be assumed, security must be applied by upper layers if necessary. The global public Internet itself is an example.

End-user Network (ENET)

a simple or complex "downstream" network that travels with the Client as a single logical unit. The ENET could be as simple as a single link connecting a single Host, or as complex as a large network with many links, routers, bridges and Hosts. The ENET could also provide an "upstream" link in a recursively-descending chain of additional Clients and ENETs. In this way, an ENET of an upstream Client is seen as the ANET of a downstream Client.

{A,I,E}NET interface

a Client's attachment to a link in an {A,I,E}NET.

*NET

a "wildcard" term used when a given specification applies equally to both ANET/INET cases. From the Client's perspective, *NET interfaces are "upstream" interfaces that connect the Client to the Mobility Service, while ENET interfaces are "downstream" interfaces that the Client uses to connect downstream ENETs, Hosts and/or other Clients.

underlay interface

an ANET/INET/ENET interface over which an OMNI interface is configured. The OMNI interface is seen as a L3 interface by the

IP layer, and each underlay interface is seen as a L2 interface by the OMNI interface. The underlay interface either connects directly to the physical communications media or coordinates with another node where the physical media is hosted.

OMNI link

a Non-Broadcast, Multiple Access (NBMA) virtual overlay configured over one or more INETs and their connected ANETs/ENETs. An OMNI link may comprise multiple distinct "segments" joined by L2 forwarding devices the same as for any link; the addressing plans in each segment may be mutually exclusive and managed by different administrative entities. Proxy/Servers and other infrastructure elements extend the link to support communications between Clients as single-hop neighbors.

OMNI interface

a node's attachment to an OMNI link, and configured over one or more underlay interfaces. If there are multiple OMNI links in an OMNI domain, a separate OMNI interface is configured for each link. The OMNI interface configures a Maximum Transmission Unit (MTU) and a Maximum Reassembly Unit (MRU) the same as any interface.

OMNI Adaptation Layer (OAL)

an OMNI interface sublayer service that encapsulates original IP packets admitted into the interface in an IPv6 header and/or subjects them to fragmentation and reassembly. The OAL is also responsible for generating MTU-related control messages as necessary, and for providing addressing context for OMNI link SRT traversal. The OAL presents a new layer in the Internet architecture known simply as the "adaptation layer".

Host

an end user device that extends the OMNI link over an ENET interface serviced by a Client. (As an implementation matter, the Host either assigns the same IP address from the ENET (underlay) interface to an (overlay) OMNI interface, or configures an OMNI-like function as a virtual sublayer of the ENET interface itself.) The IP addresses assigned to each Host ENET interface remain stable even if the Client's upstream *NET interface connections change.

Client

a network platform/device mobile router that configures one or more OMNI interfaces over distinct sets of underlay interfaces grouped as logical OMNI link units. The Client coordinates with the Mobility Service via upstream networks over *NET interfaces, and provides Proxy/Server services for Hosts and other Clients on

ENET interface downstream networks. The Client's *NET interface addresses and performance characteristics may change over time (e.g., due to node mobility, link quality, etc.) while downstream-attached Hosts and other Clients see the ENET as a stable ANET.

Proxy/Server

a segment routing topology edge node that configures an OMNI interface and connects Clients to the Mobility Service. As a server, the Proxy/Server responds directly to some Client IPv6 ND messages. As a proxy, the Proxy/Server forwards other Client IPv6 ND messages to other Proxy/Servers and Clients. As a router, the Proxy/Server provides a forwarding service for ordinary data packets that may be essential in some environments and a last resort in others. Proxy/Servers at ANET boundaries configure both an ANET downstream interface and *NET upstream interface, while INET-based Proxy/Servers configure only an INET interface.

First-Hop Segment (FHS) Proxy/Server

a Proxy/Server connected to the source Client's *NET that forwards packets sent by the source into the segment routing topology. FHS Proxy/Servers also act as intermediate forwarding nodes to facilitate RS/RA exchanges between Clients and Hub Proxy/Servers.

Last-Hop Segment (LHS) Proxy/Server

a Proxy/Server connected to the target Client's *NET that forwards packets received from the segment routing topology to the target.

Hub Proxy/Server

a single Proxy/Server selected by the Client that provides a designated router service for all of the Client's *NET underlay networks. Since all Proxy/Servers provide equivalent services, Clients normally select the first FHS Proxy/Server they coordinate with to serve as the Hub. However, the Hub can also be any available Proxy/Server for the OMNI link, i.e., and not necessarily one of the Client's FHS Proxy/Servers.

Segment Routing Topology (SRT)

a multinet forwarding region configured over one or more INETs between the FHS Proxy/Server and LHS Proxy/Server. The SRT spans the OMNI link on behalf of source/target Client pairs using segment routing in a manner outside the scope of this document (see: [I-D.templin-6man-aero]).

Mobility Service (MS)

a mobile routing service that tracks Client movements and ensures that Clients remain continuously reachable even across mobility events. The MS consists of the set of all Proxy/Servers and any other OMNI link supporting infrastructure nodes. Specific MS details are out of scope for this document, with an example found in [I-D.templin-6man-aero].

Mobility Service Prefix (MSP)

an aggregated IP Global Unicast Address (GUA) prefix (e.g., 2001:db8::/32, 192.0.2.0/24, etc.) assigned to the OMNI link and from which more-specific Mobile Network Prefixes (MNPs) are delegated. OMNI link administrators typically obtain MSPs from an Internet address registry, however private-use prefixes can also be used subject to certain limitations (see: Section 10). OMNI links that connect to the global Internet advertise their MSPs to their interdomain routing peers.

Mobile Network Prefix (MNP)

a longer IP prefix delegated from an MSP (e.g., 2001:db8:1000:2000::/56, 192.0.2.8/30, etc.) and assigned to a Client. Clients receive MNPs from Proxy/Servers and sub-delegate them to routers, Hosts and other Clients located in ENETs.

original IP packet

a whole IP packet or fragment admitted into the OMNI interface by the network layer prior to OAL encapsulation and fragmentation, or an IP packet delivered to the network layer by the OMNI interface following OAL decapsulation and reassembly.

OAL packet

an original IP packet encapsulated in an IPv6 header (i.e., the OAL header) then submitted for OAL fragmentation and reassembly.

OAL fragment

a portion of an OAL packet following fragmentation but prior to encapsulation, or following encapsulation but prior to OAL reassembly.

(OAL) atomic fragment

an OAL packet that does not require fragmentation is always encapsulated as an "atomic fragment" with a Fragment Header with Fragment Offset and More Fragments both set to 0, but with a valid Identification value.

(OAL) carrier packet

an encapsulated OAL fragment following L2 encapsulation or prior to L2 decapsulation. OAL sources and destinations exchange

carrier packets over underlay interfaces, and may be separated by one or more OAL intermediate nodes. OAL intermediate nodes may perform re-encapsulation on carrier packets by removing the L2 headers of the first hop network and replacing them with new L2 headers for the next hop network. (The term "carrier" honors agents of the service postulated by [RFC1149] and [RFC6214].)

OAL source

an OMNI interface acts as an OAL source when it encapsulates original IP packets to form OAL packets, then performs OAL fragmentation and encapsulation to create carrier packets.

OAL destination

an OMNI interface acts as an OAL destination when it decapsulates carrier packets, then performs OAL reassembly and decapsulation to derive the original IP packet.

OAL intermediate node

an OMNI interface acts as an OAL intermediate node when it removes the L2 encapsulation headers of carrier packets received on a first segment, then re-encapsulates the carrier packets in new L2 encapsulation headers and forwards them into the next segment.

OMNI Option

an IPv6 Neighbor Discovery option providing multilink parameters for the OMNI interface as specified in Section 12.

Interface Identifier (IID)

the least significant 64 bits of an IPv6 address, as specified in the IPv6 addressing architecture [RFC4291].

Link Local Address (LLA)

an IPv6 address beginning with fe80::/64 per the IPv6 addressing architecture [RFC4291] and with either a 64-bit MNP (LLA-MNP) or a 56-bit random value (LLA-RND) encoded in the IID as specified in Section 8.

Unique Local Address (ULA)

an IPv6 address beginning with fd00::/8 followed by a 40-bit Global ID followed by a 16-bit Subnet ID per [RFC4193] and with either a 64-bit MNP (ULA-MNP) or a 56-bit random value (ULA-RND) encoded in the IID as specified in Section 9. (Note that [RFC4193] specifies a second form of ULAs based on the prefix fc00::/8, which are referred to as "ULA-C" throughout this document to distinguish them from the ULAs defined here.)

Temporary Local Address (TLA)

a ULA beginning with fd00::/16 followed by a 48-bit randomly-initialized value followed by an MNP-based (TLA-MNP) or random (TLA-RND) IID as specified in Section 9. Clients use TLAs to bootstrap autoconfiguration in the presence of OMNI link infrastructure or for sustained communications in the absence of infrastructure. (Note that in some environments Clients can instead use a (Hierarchical) Host Identity Tag ((H)HIT) instead of a TLA - see: Section 22.)

eXtended Local Address (XLA)

a TLA beginning with fd00::/64 followed by an MNP-based (XLA-MNP) or random (XLA-RND) IID as specified in Section 9. An XLA is simply a TLA with an all-0 48-bit value following fd00::/16, and can be used to supply a "wildcard match" for IPv6 ND cache entries, a routing table entry for the OMNI link routing system, etc. (Note that XLAs can also be statelessly formed from LLAs (and vice-versa) simply by inverting prefix bits 7 and 8.)

Multilink

a Client OMNI interface's manner of managing multiple diverse *NET underlay interfaces as a single logical unit. The OMNI interface provides a single unified interface to upper layers, while underlay interface selections are performed on a per-packet basis considering traffic selectors such as DSCP, flow label, application policy, signal quality, cost, etc. Multilink selections are coordinated in both the outbound and inbound directions based on source/target underlay interface pairs.

Multinet

an intermediate node's manner of spanning multiple diverse IP Internetwork and/or private enterprise network "segments" at the OAL layer below IP. Through intermediate node concatenation of SRT network segments, multiple diverse Internetworks (such as the global public IPv4 and IPv6 Internets) can serve as transit segments in an end-to-end L2 forwarding path. This OAL concatenation capability provides benefits such as supporting IPv4/IPv6 transition and coexistence, joining multiple diverse operator networks into a cooperative single service network, etc. See: [I-D.templin-6man-aero] for further information.

Multihop

an iterative relaying of IP packets between Client's over an OMNI underlay interface technology (such as omnidirectional wireless) without support of fixed infrastructure. Multihop services entail Client-to-Client relaying within a Mobile/Vehicular Ad-hoc Network (MANET/VANET) for Vehicle-to-Vehicle (V2V) communications and/or for Vehicle-to-Infrastructure (V2I) "range extension" where Clients within range of communications infrastructure elements provide forwarding services for other Clients.

Mobility

any action that results in a change to a Client underlay interface address. The change could be due to, e.g., a handover to a new wireless base station, loss of link due to signal fading, an actual physical node movement, etc.

Safety-Based Multilink (SBM)

A means for ensuring fault tolerance through redundancy by connecting multiple OMNI interfaces within the same domain to independent routing topologies (i.e., multiple independent OMNI links).

Performance Based Multilink (PBM)

A means for selecting one or more underlay interface(s) for packet transmission and reception within a single OMNI interface.

OMNI Domain

The set of all SBM/PBM OMNI links that collectively provides services for a common set of MSPs. All OMNI links within the same domain configure, advertise and respond to the same OMNI IPv6 Anycast address(es).

Multilink Forwarding Information Base (MFIB)

A forwarding table on each OMNI source, destination and intermediate node that includes Multilink Forwarding Vectors (MFV) with both next hop forwarding instructions and context for reconstructing compressed headers for specific underlay interface pairs used to communicate with peers. See: [I-D.templin-6man-aero] for further discussion.

Multilink Forwarding Vector (MFV)

An MFIB entry that includes soft state for each underlay interface pairwise communication session between peers. MFVs are identified by both a next-hop and previous-hop MFV Index (MFVI), with the next-hop established based on an IPv6 ND solicitation and the previous hop established based on the solicited IPv6 ND advertisement response. See: [I-D.templin-6man-aero] for further discussion.

Multilink Forwarding Vector Index (MVFI)

A 4 octet value selected by an OMNI node when it creates an MFV, then advertised to either a next-hop or previous-hop. OMNI intermediate nodes assign two distinct MFVIs for each MFV and advertise one to the next-hop and the other to the previous-hop. OMNI end systems assign and advertise a single MFVI. See: [I-D.templin-6man-aero] for further discussion.

IP Jumbogram

an IPv4 or IPv6 packet with a Jumbo Payload option that includes a 32-bit length field to be used instead of the 16-bit {Total, Payload} Length field (see: Section 5.1). For IPv4, the Total Length field must be set to the length of the IPv4 header only. For IPv6, the Payload Length must be set to 0.

IP Parcel

a special form of an IP Jumbogram with a segment length value included in the {Total, Payload} Length field and also with a Jumbo Payload option (see: Section 5.2).

INADDR

the IP address (and also the UDP port number when UDP is used) that appears in (L2) encapsulation headers in the data plane and in IPv6 ND OMNI option sub-options in the control plane.

3. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

An implementation is not required to internally use the architectural constructs described here so long as its external behavior is consistent with that described in this document.

4. Overlay Multilink Network (OMNI) Interface Model

An OMNI interface is a virtual interface configured over one or more underlay interfaces, which may be physical (e.g., an aeronautical radio link, etc.) or virtual (e.g., an Internet or higher-layer "tunnel"). The OMNI interface architectural layering model is the same as in [RFC5558][RFC7847], and augmented as shown in Figure 1. The IP layer therefore sees the OMNI interface as a single L3 interface nexus for multiple underlay interfaces that appear as L2 communication channels in the architecture.

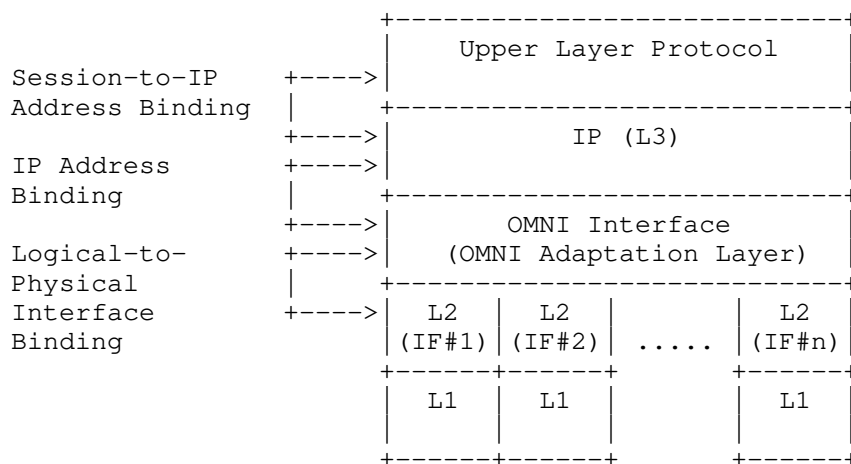


Figure 1: OMNI Interface Architectural Layering Model

Each underlay interface provides an L2/L1 abstraction according to one of the following models:

- * ANET interfaces connect to a protected and secured ANET that is separated from the open INET by Proxy/Servers. The ANET interface may be either on the same L2 link segment as a Proxy/Server, or separated from a Proxy/Server by multiple IP hops. (Note that NATs may appear internally within an ANET or on the Proxy/Server itself and may require NAT traversal the same as for the INET case.)
- * INET interfaces connect to an INET either natively or through one or several IPv4 Network Address Translators (NATs). Native INET interfaces have global IP addresses that are reachable from any INET correspondent. NATed INET interfaces typically configure private IP addresses and connect to a private network behind one or more NATs with the outermost NAT providing INET access.
- * ENET interfaces connect a Client's downstream-attached networks, where the Client provides forwarding services for ENET Host and Client communications to remote peers. An ENET may be as simple as a small stub network that travels with a mobile Client (e.g., an Internet-of-Things) to as complex as a large private enterprise network that the Client connects to a larger ANET or INET. Downstream-attached Hosts and Clients see the ENET as an ANET and see the (upstream) Client as a Proxy/Server.

- * VPNed interfaces use security encapsulation over an underlay network to a Client or Proxy/Server acting as a Virtual Private Network (VPN) gateway. Other than the link-layer encapsulation format, VPNed interfaces behave the same as for Direct interfaces.
- * Direct (aka "point-to-point") interfaces connect directly to a Client or Proxy/Server without crossing any networked paths. An example is a line-of-sight link between a remote pilot and an unmanned aircraft.

The OMNI interface forwards original IP packets from the network layer (L3) using the OMNI Adaptation Layer (OAL) (see: Section 5) as an encapsulation and fragmentation sublayer service. This "OAL source" then further encapsulates the resulting OAL packets/fragments in underlay network headers (e.g., UDP/IP, IP-only, Ethernet-only, etc.) to create L2-encapsulated "carrier packets" for transmission over underlay interfaces. The target OMNI interface receives the carrier packets from underlay interfaces and discards the L2 encapsulation headers. If the resulting OAL packets/fragments are addressed to itself, the OMNI interface acts as an "OAL destination" and performs reassembly if necessary, discards the OAL encapsulation, and delivers the original IP packet to the network layer. If the OAL fragments are addressed to another node, the OMNI interface instead acts as an "OAL intermediate node" by re-encapsulating the carrier packets in new underlay network L2 headers and forwarding them over an underlay interface without reassembling or discarding the OAL encapsulation. The OAL source and OAL destination are seen as "neighbors" on the OMNI link, while OAL intermediate nodes provide a virtual bridging service that joins the segments of a (multinet) Segment Routing Topology (SRT).

The OMNI interface can forward original IP packets over underlay interfaces while including/omitting various lower layer encapsulations including OAL, UDP, IP and Ethernet (ETH) or other link-layer header. The network layer can also access the underlay interfaces directly while bypassing the OMNI interface entirely when necessary. This architectural flexibility may be beneficial for underlay interfaces (e.g., some aviation data links) for which encapsulation overhead may be a primary consideration. OMNI interfaces that send original IP packets directly over underlay interfaces without invoking the OAL can only reach peers located on the same OMNI link segment. Source Clients can instead use the OAL to coordinate with target Clients in the same or different OMNI link segments by sending initial carrier packets to a First-Hop Segment (FHS) Proxy/Server. The FHS Proxy/Server then forwards the packets into the SRT spanning tree, which transports them to a Last-Hop Segment (LHS) Proxy/Server for the target Client.

Original IP packets sent directly over underlay interfaces are subject to the same path MTU related issues as for any Internetworking path, and do not include per-packet identifications that can be used for data origin verification and/or link-layer retransmissions. Original IP packets presented directly to an underlay interface that exceed the underlay network path MTU are dropped with an ordinary ICMPv6 Packet Too Big (PTB) message returned. These PTB messages are subject to loss [RFC2923] the same as for any non-OMNI IP interface.

The OMNI interface encapsulation/decapsulation layering possibilities are shown in Figure 2 below. Imaginary vertical lines drawn between the Network Layer and Underlay interfaces in the figure denote the encapsulation/decapsulation layering combinations possible. Common combinations include IP-only (i.e., direct access to underlay interfaces with or without using the OMNI interface), IP/IP, IP/UDP/IP, IP/UDP/IP/ETH(ERNET), IP/OAL/UDP/IP, IP/OAL/UDP/ETH, etc.

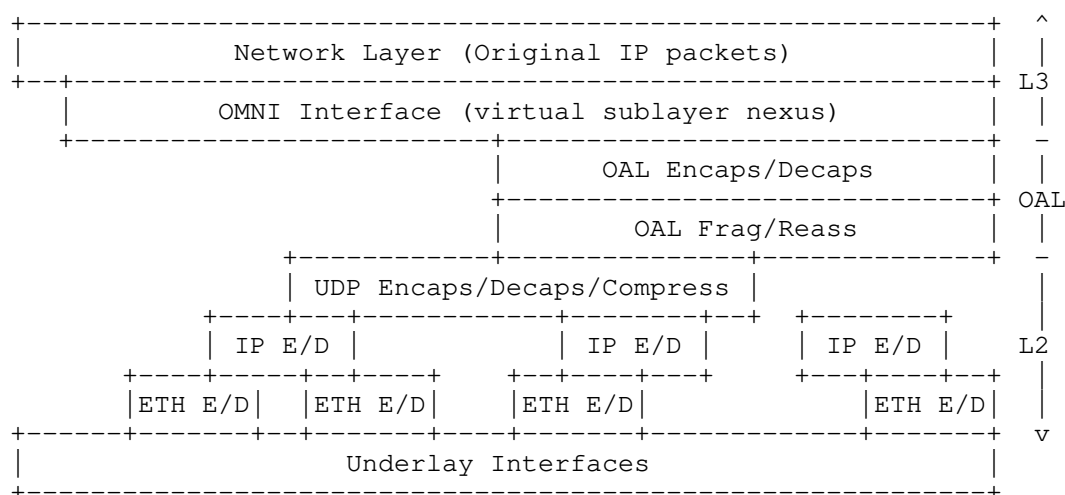


Figure 2: OMNI Interface Layering

The OMNI/OAL model gives rise to a number of opportunities:

- * Clients receive MNPs from the MS, and coordinate with the MS through IPv6 ND message exchanges with Proxy/Servers. Clients use the MNP to construct a unique Link-Local Address (LLA-MNP) through the algorithmic derivation specified in Section 8 and assign the LLA to the OMNI interface. Since LLA-MNPs are uniquely derived from an MNP, no Duplicate Address Detection (DAD) or Multicast Listener Discovery (MLD) messaging is necessary.

- * since Temporary ULAs with random IIDs (TLA-RNDs) are statistically unique, they can be used without DAD until an MNP is obtained.
- * underlay interfaces on the same L2 link segment as a Proxy/Server do not require any L3 addresses (i.e., not even link-local) in environments where communications are coordinated entirely over the OMNI interface.
- * as underlay interface properties change (e.g., link quality, cost, availability, etc.), any active interface can be used to update the profiles of multiple additional interfaces in a single message. This allows for timely adaptation and service continuity under dynamically changing conditions.
- * coordinating underlay interfaces in this way allows them to be represented in a unified MS profile with provisions for mobility and multilink operations.
- * exposing a single virtual interface abstraction to the IPv6 layer allows for multilink operation (including QoS based link selection, packet replication, load balancing, etc.) at L2 while still permitting L3 traffic shaping based on, e.g., DSCP, flow label, etc.
- * the OMNI interface allows multinet traversal over the SRT when communications across different administrative domain network segments are necessary. This mode of operation would not be possible via direct communications over the underlay interfaces themselves.
- * the OAL supports lossless and adaptive path MTU mitigations not available for communications directly over the underlay interfaces themselves. The OAL supports "packing" of multiple IP payload packets within a single OAL "super-packet" and also supports transmission of IP packets and parcels of all sizes up to and including Jumbograms.
- * the OAL applies per-packet identification values that allow for link-layer reliability and data origin authentication.
- * L3 sees the OMNI interface as a point of connection to the OMNI link; if there are multiple OMNI links, L3 will see multiple OMNI interfaces.
- * Multiple independent OMNI interfaces can be used for increased fault tolerance through Safety-Based Multilink (SBM), with Performance-Based Multilink (PBM) applied within each interface.

- * Multiple independent OMNI links can be joined together into a single link without requiring renumbering of infrastructure elements, since the ULAs assigned to the different links will be mutually exclusive.
- * the OMNI/OAL model supports transmission of a new form of IP packets known as "IP Parcels" that improve performance and efficiency for both upper layer protocols and networked paths.

Note that even when the OMNI virtual interface is present, applications can still access underlay interfaces either through the network protocol stack using an Internet socket or directly using a raw socket. This allows for intra-network (or point-to-point) communications without invoking the OMNI interface and/or OAL. For example, when an OMNI interface is configured over an underlay IP interface, applications can still invoke intra-network IP communications directly over the underlay interface as long as the communicating endpoints are not subject to mobility dynamics.

Figure 3 depicts the architectural model for a source Client with an attached ENET connecting to the OMNI link via multiple independent ANETs/INETs (i.e., *NETs). The Client's OMNI interface sends IPv6 ND solicitation messages over available *NET underlay interfaces using any necessary L2 encapsulations. The IPv6 ND messages traverse the *NETs until they reach an FHS Proxy/Server (FHS#1, FHS#2, ..., FHS#n), which returns an IPv6 ND advertisement message and/or forwards a proxied version of the message over the SRT to an LHS Proxy/Server near the target Client (LHS#1, LHS#2, ..., LHS#m). The Hop Limit in IPv6 ND messages is not decremented due to encapsulation; hence, the source and target Client OMNI interfaces appear to be attached to a common link.

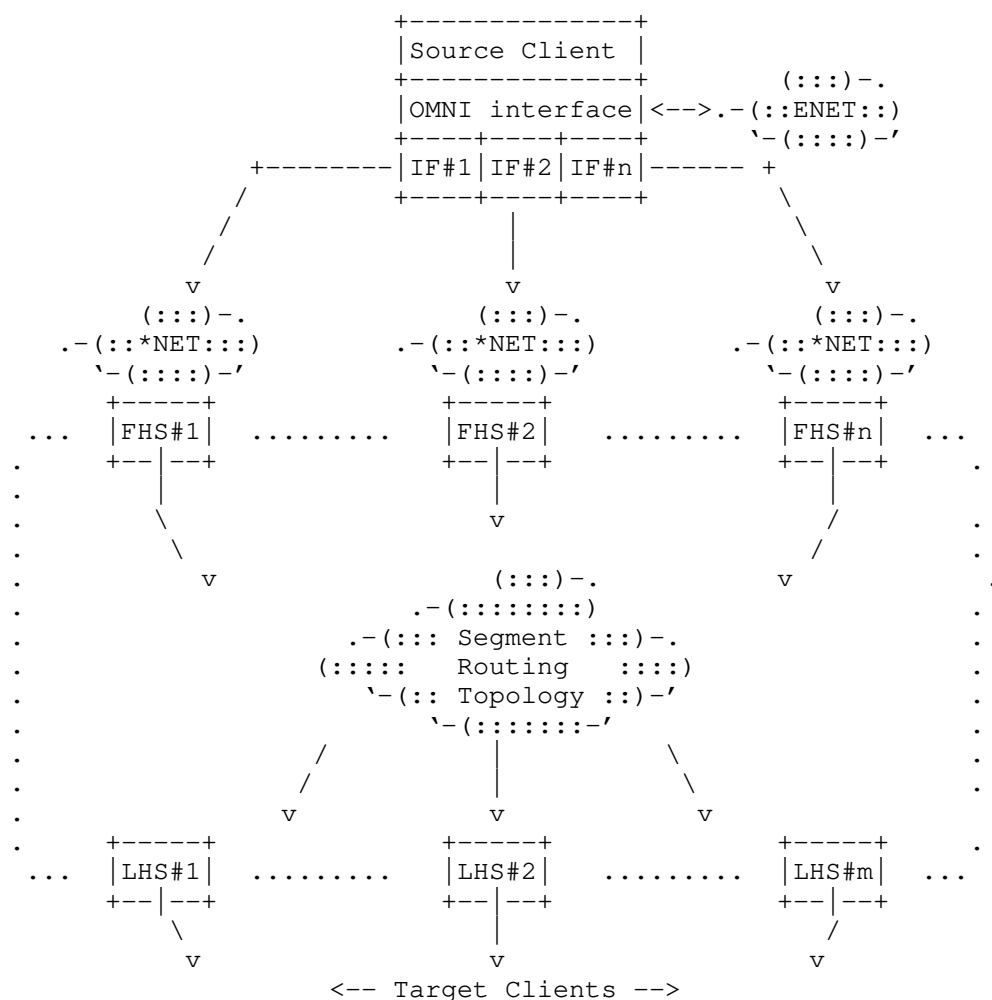


Figure 3: Source/Target Client Coordination over the OMNI Link

After the initial IPv6 ND message exchange, the source Client (as well as any nodes on its attached ENETs) can send packets to the target Client over the OMNI interface. OMNI interface multilink services will forward the packets via FHS Proxy/Servers for the correct underlay *NETs. The FHS Proxy/Server then forwards the packets over the SRT which delivers them to an LHS Proxy/Server, and the LHS Proxy/Server in turn forwards them to the target Client. (Note that when the source and target Client are on the same SRT segment, the FHS and LHS Proxy/Servers may be one and the same.)

Clients select a Hub Proxy/Server (not shown in the figure), which will often be one of their FHS Proxy/Servers but could also be any Proxy/Server on the OMNI link. Clients then register all of their *NET underlay interfaces with the Hub Proxy/Server via the FHS Proxy/Server in a pure proxy role. The Hub Proxy/Server then provides a designated router service for the Client, and the Client can quickly migrate to a new Hub Proxy/Server if the first becomes unresponsive.

Clients therefore use Proxy/Servers as gateways into the SRT to reach OMNI link correspondents via a spanning tree established in a manner outside the scope of this document. Proxy/Servers forward critical MS control messages via the secured spanning tree and forward other messages via the unsecured spanning tree (see Security Considerations). When route optimization is applied as discussed in [I-D.templin-6man-aero], Clients can instead forward directly to SRT intermediate nodes (or directly to correspondents in the same SRT segment) to reduce Proxy/Server load.

Note: while not shown in the figure, a Client's ENET may connect many additional Hosts and even other Clients in a recursive extension of the OMNI link. This OMNI virtual link extension will be discussed more fully throughout the document.

5. OMNI Interface Maximum Transmission Unit (MTU)

The OMNI interface observes the link nature of tunnels, including the Maximum Transmission Unit (MTU), Maximum Reassembly Unit (MRU) and the role of fragmentation and reassembly [I-D.ietf-intarea-tunnels]. The OMNI interface is configured over one or more underlay interfaces as discussed in Section 4, where the interfaces (and their associated underlay network paths) may have diverse MTUs. OMNI interface considerations for accommodating original IP packets of various sizes are discussed in the following sections.

IPv6 underlay interfaces are REQUIRED to configure a minimum MTU of 1280 octets and a minimum MRU of 1500 octets [RFC8200]. Therefore, the minimum IPv6 path MTU is 1280 octets since routers on the path are not permitted to perform network fragmentation even though the destination is required to reassemble more. The network therefore MUST forward original IP packets of at least 1280 octets without generating an IPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) message [RFC8201]. (While the source can apply "source fragmentation" for locally-generated IPv6 packets up to 1500 octets and larger still if it knows the destination configures a larger MRU, this does not affect the minimum IPv6 path MTU.)

IPv4 underlay interfaces are REQUIRED to configure a minimum MTU of 68 octets [RFC0791] and a minimum MRU of 576 octets [RFC0791][RFC1122]. Therefore, when the Don't Fragment (DF) bit in the IPv4 header is set to 0 the minimum IPv4 path MTU is 576 octets since routers on the path support network fragmentation and the destination is required to reassemble at least that much. The OMNI interface therefore MUST set DF to 0 in the IPv4 encapsulation headers of carrier packets that are no larger than 576 octets, and SHOULD set DF to 1 in larger carrier packets unless it has a way to determine the encapsulation destination MRU and has carefully considered the issues discussed in Section 6.12.

When the network layer admits an original IP packet into the OMNI interface the OAL prepends an IPv6 encapsulation header (see: Section 6) where the 16-bit Payload Length field limits the maximum-sized original IP packet to $(2^{16} - 1) = 65535$ octets; this is also the maximum size that the OAL can accommodate with IPv6 fragmentation. The OMNI interface therefore sets an MTU and MRU of 65535 octets to support assured delivery of original packets no larger than this size even if IPv6 fragmentation is required. (The OMNI interface MAY set a larger MTU to support best-effort delivery for larger packets; see below.) The OMNI interface then employs the OAL as an encapsulation sublayer service to transform original IP packets into OAL packets/fragments, and the OAL in turn uses underlay network encapsulation to forward carrier packets over underlay interfaces (see: Section 6).

5.1. Jumbograms

While the maximum-sized original IP packet that the OAL can accommodate using IPv6 fragmentation is 65535 octets, OMNI interfaces can forward still larger IPv6 packets as OAL "atomic fragments" through the application of IPv6 Jumbograms [RFC2675]. For such larger packets, the OMNI interface performs OAL encapsulation by appending an IPv6 header followed by an 8-octet Hop-By-Hop header with Jumbo Payload option followed by a Routing Header of no more than 40-octets (if necessary) and finally followed by an 8-octet Fragment Header.

Since the Jumbo Payload option includes a 32-bit length field, OMNI interfaces can therefore configure a larger IP MTU up to a maximum of $((2^{32} - 1) - 8 - 40 - 8) = 4294967239$ octets. In that case, the OAL will still provide original IP packets no larger than 65535 with an IPv6 fragmentation-based assured delivery service while larger IP packets will receive a best-effort delivery service as atomic fragments (note that the OAL destination is permitted to accept atomic fragments that exceed the OMNI interface MRU).

The OAL source forwards jumbo atomic fragments under the assumption that upper and lower layers will employ sufficient integrity assurance, noting that commonly-used 32-bit CRCs may be inadequate for these larger sizes [CRC]. If the packet is dropped along the path to the OAL destination, the OAL source must arrange to return a PTB "hard error" to the original source Section 6.8.

This document notes that a Jumbogram service for IPv4 is also specified in [I-D.templin-intarea-parcels], where all OMNI link aspects of the service are conducted in a similar fashion as for IPv6 above.

5.2. IPv6 Parcels

As specified in [I-D.templin-intarea-parcels], an IP Parcel is a variation of the IP Jumbogram construction beginning with an IP header with the length of the first upper layer protocol segment in the {Total, Payload} Length field, but with a Jumbo Payload option with a length that may be the same as or larger than the length in the IP header. The differences in these lengths determines the size and number of upper layer protocol segments within the parcel.

The IP Parcel format and transmission/reception procedures for OMNI interfaces are specified in Section 6.14. End systems that implement either the full OMNI interface (i.e., Clients) or enough of the OAL to process parcels (i.e., Hosts) are permitted to exchange parcels with consenting peers.

6. The OMNI Adaptation Layer (OAL)

When an OMNI interface forwards an original IP packet from the network layer for transmission over one or more underlay interfaces, the OMNI Adaptation Layer (OAL) acting as the OAL source applies encapsulation to form OAL packets subject to fragmentation producing OAL fragments suitable for L2 encapsulation and transmission as carrier packets over underlay interfaces as described in Section 6.1.

These carrier packets travel over one or more underlay networks spanned by OAL intermediate nodes in the SRT, which re-encapsulate by removing the L2 headers of the first underlay network and appending L2 headers appropriate for the next underlay network in succession. (This process supports the multinet concatenation capability needed for joining multiple diverse networks.) After re-encapsulation by zero or more OAL intermediate nodes, the carrier packets arrive at the OAL destination.

When the OAL destination receives the carrier packets, it discards the L2 headers and reassembles the resulting OAL fragments (if necessary) into an OAL packet as described in Section 6.3. The OAL destination next decapsulates the OAL packet to obtain the original IP packet then delivers the original IP packet to the network layer. The OAL source may be either the source Client or its FHS Proxy/Server, while the OAL destination may be either the LHS Proxy/Server or the target Client. Proxy/Servers (and SRT Gateways as discussed in [I-D.templin-6man-aero]) may also serve as OAL intermediate nodes.

The OAL presents an OMNI sublayer abstraction similar to ATM Adaptation Layer 5 (AAL5). Unlike AAL5 which performs segmentation and reassembly with fixed-length 53 octet cells over ATM networks, however, the OAL uses IPv6 encapsulation, fragmentation and reassembly with larger variable-length cells over heterogeneous underlay networks. Detailed operations of the OAL are specified in the following sections.

6.1. OAL Source Encapsulation and Fragmentation

When the network layer forwards an original IP packet into the OMNI interface, the OAL source creates an "OAL packet" by prepending an IPv6 OAL encapsulation header per [RFC2473] but does not decrement the Hop Limit/TTL of the original IP packet since encapsulation occurs at a layer below IP forwarding. The OAL source copies the "Type of Service/Traffic Class" [RFC2983] and "Explicit Congestion Notification (ECN)" [RFC3168] values in the original packet's IP header into the corresponding fields in the OAL header, then sets the OAL header "Flow Label" as specified in [RFC6438]. The OAL source finally sets the OAL header IPv6 Payload Length to the length of the original IP packet and sets Hop Limit to a value that MUST NOT be larger than 63 yet is still sufficiently large to enable loop-free forwarding over multiple concatenated OMNI link intermediate hops.

The OAL next selects OAL packet source and destination addresses. Client OMNI interfaces set the OAL source address to a Unique Local Address (ULA) based on the Mobile Network Prefix (ULA-MNP). When a Client OMNI interface does not (yet) have a ULA prefix and/or an MNP suffix, it can instead use a Temporary ULA (TLA) (or a (Hierarchical) Host Identity Tag ((H)HIT - see: Section 22) as an OAL address. Finally, when the Client needs to express its MNP outside the context of a specific ULA prefix, it can use an eXtended ULA (XLA). Proxy/Server OMNI interfaces instead set the source address to a Random ULA (ULA-RND) (see: Section 9), but also process packets with anycast and/or multicast OAL addresses that they are configured to recognize.)

The OAL source next selects a 32-bit OAL packet Identification value as specified in Section 6.6. The OAL then calculates a 2-octet OAL checksum using the algorithm specified in Appendix A. The OAL source calculates the checksum over the OAL packet beginning with a pseudo-header of the OAL header similar to that found in Section 8.1 of [RFC8200], then extending over the entire length of the original IP packet. The OAL pseudo-header is formed as shown in Figure 4:

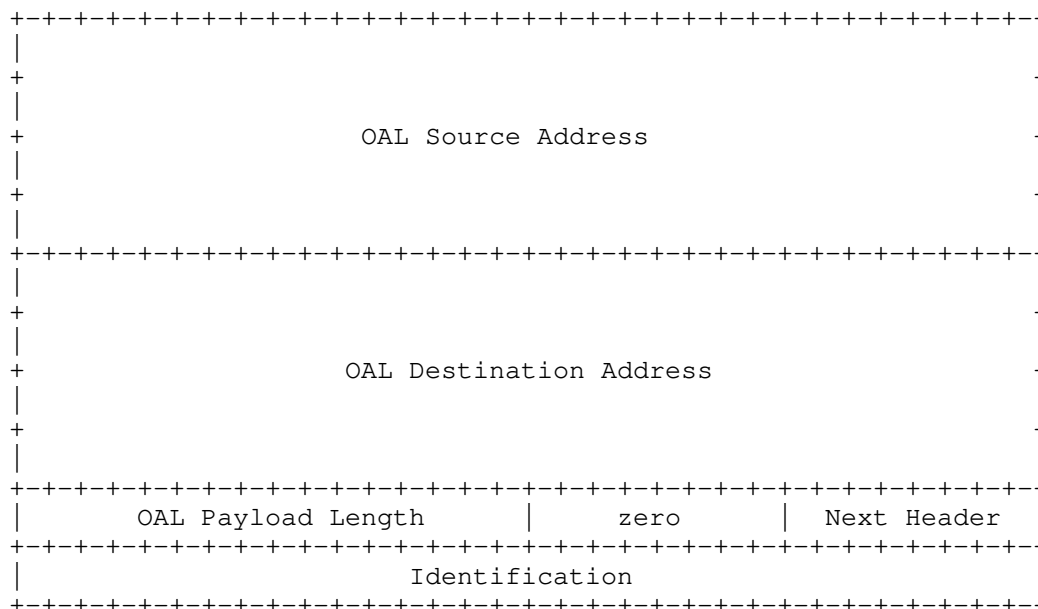


Figure 4: OAL Pseudo-Header

After calculating the checksum, the OAL source next fragments the OAL packet if necessary while assuming the IPv4 minimum path MTU (i.e., 576 octets) as the worst case for OAL fragmentation regardless of the underlay interface IP protocol version since IPv6/IPv4 protocol translation and/or IPv6-in-IPv4 encapsulation may occur in any underlay network path. By initially assuming the IPv4 minimum even for IPv6 underlay interfaces, the OAL source may produce smaller fragments with additional encapsulation overhead but avoids loss due to presenting an underlay interface with a carrier packet that exceeds its MRU. Additionally, the OAL path could traverse multiple SRT segments with intermediate OAL forwarding nodes performing re-encapsulation where the L2 encapsulation of the previous segment is replaced by the L2 encapsulation of the next segment which may be based on a different IP protocol version and/or encapsulation sizes.

The OAL source therefore assumes a default minimum path MTU of 576 octets at each SRT segment for the purpose of generating OAL fragments for L2 encapsulation and transmission as carrier packets. Each successive SRT intermediate node may include either a 20 octet IPv4 or 40 octet IPv6 header, an 8 octet UDP header and in some cases an IP security encapsulation (40 octets maximum assumed) during re-encapsulation. Intermediate nodes at any SRT segment may also insert or modify the Routing Header (40 octets maximum) following the 40 octet OAL IPv6 header and preceding the 8 octet Fragment Header. Therefore, assuming a worst case of $(40 + 40 + 8) = 88$ octets for L2 encapsulations plus $(40 + 40 + 8) = 88$ octets for OAL encapsulation leaves no less than $(576 - 88 - 88) = 400$ octets remaining to accommodate a portion of the original IP packet/fragment. The OAL source therefore sets a minimum Maximum Payload Size (MPS) of 400 octets as the basis for the minimum-sized OAL fragment that can be assured of traversing all SRT segments without loss due to an MTU/MRU restriction. The Maximum Fragment Size (MFS) for OAL fragmentation is therefore determined by the MPS plus the size of the OAL encapsulation headers.

The OAL source SHOULD maintain "path MPS" values for individual OAL destinations initialized to the minimum MPS and increased to larger values if better information is known or discovered. For example, when peers share a common underlay network link or a fixed path with a known larger MTU, the OAL source can set path MPS to a larger size (i.e., greater than 400 octets) as long as the peer reassembles before re-encapsulating and forwarding (while re-fragmenting if necessary). Also, if the OAL source has a way of knowing the maximum L2 encapsulation size for all SRT segments along the path it may be able to increase path MPS to reserve additional room for payload data. Even when OAL header compression is used, the OAL source must include the uncompressed OAL header size in its path MPS calculation since it may need to include a full header at any time.

The OAL source can also optimistically set a larger path MPS and/or actively probe individual OAL destinations to discover larger sizes using packetization layer probes in a similar fashion as [RFC4821][RFC8899], but care must be taken to avoid setting static values for dynamically changing paths leading to black holes. The probe involves sending an OAL packet larger than the current path MPS and receiving a small acknowledgement response (with the possible receipt of link-layer error message when a probe is lost). For this purpose, the OAL source can send an NS message with one or more OMNI options with large PadN sub-options (see: Section 12) and/or with a trailing large NULL packet in a super-packet (see: Section 6.9) in order to receive a small NA response from the OAL destination. While observing the minimum MPS will always result in robust and secure behavior, the OAL source should optimize path MPS values when more

efficient utilization may result in better performance (e.g. for wireless aviation data links). The OAL source should maintain separate path MPS values for each (source, target) underlay interface pair for the same OAL destination, since different underlay interface pairs may support differing path MPS values.

When the OAL source performs fragmentation, it SHOULD produce the minimum number of non-overlapping fragments under current MPS constraints, where each non-final fragment MUST be at least as large as the minimum MPS, while the final fragment MAY be smaller. The OAL source also converts all original IP packets no larger than the current MPS (or larger than 65535 octets) into atomic fragments by including a Fragment Header with Fragment Offset and More Fragments both set to 0. The OAL source then inserts a Routing Header (if necessary) following the IPv6 encapsulation header and before the Fragment Header. If the original IP packet is larger than 65535, the OAL source also inserts a Hop-By-Hop header with Jumbo Payload option immediately following the IPv6 encapsulation header and before the Routing Header (if necessary), then includes an (atomic) Fragment Header. The header extension order for each fragment therefore appears as the OAL IPv6 header followed by Hop-By-Hop header followed by Routing Header followed by Fragment Header.

The OAL source next appends the OAL checksum as the final two octets of the final fragment while increasing its (Jumbo) Payload Length by 2. If appending the checksum would cause the final fragment to exceed the current MPS, the OAL source instead reduces this "former" final fragment's Payload Length (PL) by $(N*8 + (PL \bmod 8))$ octets, where N is an integer that would result in a non-zero reduction but without causing the former final fragment to become smaller than the minimum MPS. The OAL source then creates a "new" final fragment by copying the OAL IPv6 header and extension headers from the former final fragment, then copying the $(N*8 + (PL \bmod 8))$ octets from the end of the former final fragment immediately following the new final fragment extension headers. The OAL source then sets the former final fragment's More Fragments flag to 1, increments the new final fragment's fragment offset by the former final fragment's new $(PL / 8)$ and finally appends the checksum the same as discussed above.

Next, the OAL source replaces the IPv6 Fragment Header 1-octet "Reserved" field (and for first fragments also the 2-bit "Reserved Flags" field) with OMNI-specific encodings as shown in:

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Next Header | Parcel ID |A| Fragment Offset |P|S|M|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Identification
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

a) First fragment

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Next Header | Ordinal |A| Fragment Offset |Res|M|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Identification
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

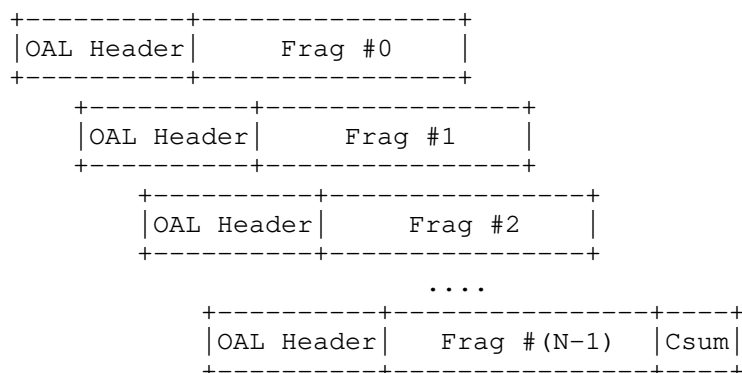
```

a) Non-first fragment

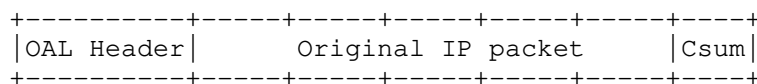
Figure 5: IPv6 Fragment Header Reserved Fields Redefined

For the first fragment, the OAL source sets the "(A)RQ" flag then sets "Parcel ID", "(P)arcel" and "(S)ub-Parcels" as specified in Section 6.14. For each non-first fragment, the OAL source instead sets the "(A)RQ" flag and writes a monotonically-increasing "Ordinal" value between 1 and 127. Specifically, the OAL source writes the ordinal number '1' for the first non-first fragment, '2' for the second, '3' for the third, etc. up to the final fragment or the ordinal value '127', whichever comes first. (For any additional non-first fragments beyond ordinal '127', the OAL source instead writes the value '0' in the Ordinal field and clears the "(A)RQ" flag. The first fragment is implicitly always considered ordinal number '0' even though the header does not include an explicit Ordinal field.)

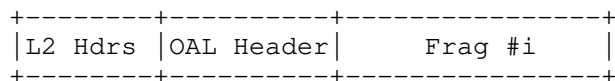
The OAL source finally encapsulates the fragments in L2 headers to form carrier packets and forwards them over an underlay interface, while retaining the fragments and their ordinal numbers (i.e., #0, #1, #2, etc. up to #127) for a brief period to support link-layer retransmissions (see: Section 6.7). OAL fragment and carrier packet formats are shown in Figure 6.



a) OAL fragmentation (Csum in final fragment)



b) An OAL atomic fragment



c) OAL carrier packet after L2 encapsulation

Figure 6: OAL Fragments and Carrier Packets

Note: the minimum MPS assumes that any middleboxes (e.g. IPv4 NATs) that connect private networks with path MTUs smaller than 576 octets must reassemble any fragmented (outbound) IPv4 carrier packets sent by OAL sources before forwarding them to external Internetworks since middleboxes that connect OAL destinations often unconditionally drop (inbound) IPv4 fragments. However, when the path MTU in the destination private network is small, the OAL destination itself will be able to reassemble any IPv4 fragmentation that occurs in the inbound path.

6.2. OAL L2 Encapsulation and Re-Encapsulation

The OAL source or intermediate node next encapsulates each OAL fragment (with either full or compressed headers) in L2 encapsulation headers to create a carrier packet. The OAL source or intermediate node (i.e., the L2 source) includes a UDP header as the innermost sublayer if NAT traversal and/or packet filtering middlebox traversal are required; otherwise, the L2 source includes either a full or compressed IP header and/or an actual link-layer header (e.g., such

as for Ethernet-compatible links). The L2 source then appends any additional encapsulation sublayer headers necessary and presents the resulting carrier packet to an underlay interface, where the underlay network conveys it to a next-hop OAL intermediate node or destination (i.e., the L2 destination).

The L2 source encapsulates the OAL information immediately following the innermost L2 sublayer header. If the first four bits of the encapsulated OAL information following the innermost sublayer header encode the value '6', the information must include an uncompressed IPv6 header (plus extensions) followed by upper layer protocol headers and data. If the first four bits encode the value '4', an uncompressed IPv4 header (plus extensions) followed by upper layer protocol headers and data follows. Otherwise, the first four bits include a "Type" value, and the OAL information appears in an alternate format as specified in Section 6.4 (Types '0' and '1' are currently specified while all other values are reserved for future use). Carrier packets that contain an unrecognized Type value are unconditionally dropped.

The OAL node prepares the innermost L2 encapsulation header for OAL packets as follows:

- * For UDP encapsulation, the L2 source sets the UDP source port to 8060 (i.e., the port number reserved for AERO/OMNI). When the L2 destination is a Proxy/Server or Gateway, the L2 source sets the UDP destination port to 8060; otherwise, the L2 source sets the UDP destination port to its cached port number value for the peer. The L2 source finally sets the UDP Length the same as specified in [RFC0768]. (If the OAL packet includes an IP Jumbogram, the L2 source instead sets the UDP length to 0 and includes a Jumbo Payload option in the L2 IP header.)
- * For IP encapsulation, the L2 source sets the IP {Protocol, Next-Header} to TBD1 (see: IANA Considerations) and sets the {Total, Payload} Length the same as specified in [RFC0791] or [RFC8200]. (If the OAL packet includes a true Jumbogram, the L2 source includes a Jumbo Payload option and sets {Total, Payload} Length plus the Jumbo Payload length according to the OAL length information.)

- * For direct encapsulations over Ethernet-compatible links, the EtherType is set to TBD2 (see: IANA Considerations). Since the Ethernet header does not include a length field, for the OMNI EtherType the Ethernet header is followed by a four-octet Payload Length field followed immediately by the encapsulated OAL information. The Payload Length field encodes the length in octets (in network byte order) of the OAL information exclusive of the lengths of the Ethernet header and trailer.

When an L2 source includes a UDP header, it SHOULD calculate and include a UDP checksum in carrier packets with full OAL headers to prevent mis-delivery, and MAY disable UDP checksums in carrier packets with compressed OAL headers (see: Section 6.4). If the L2 source discovers that a path is dropping carrier packets with UDP checksums disabled, it should enable UDP checksums in future carrier packets sent to the same L2 destination. If the L2 source discovers that a path is dropping carrier packets that do not include a UDP header, it should include a UDP header in future carrier packets.

When an L2 source sends carrier packets with compressed OAL headers and with UDP checksums disabled, mis-delivery due to corruption of the 4-octet Multilink Forwarding Vector Index (MFVI) is possible but unlikely since the corrupted index would somehow have to match valid state in the (sparsely-populated) Multilink Forwarding Information Based (MFIB). In the unlikely event that a match occurs, an OAL destination may receive a mis-delivered carrier packet but can immediately reject packets with an incorrect Identification. If the Identification value is somehow accepted, the OAL destination may submit the mis-delivered carrier packet to the reassembly cache where it will most likely be rejected due to incorrect reassembly parameters. If a reassembly that includes the mis-delivered carrier packets somehow succeeds (or, for atomic fragments) the OAL destination will verify the OAL checksum to detect corruption. Finally, any spurious data that somehow eludes all prior checks will be detected and rejected by end-to-end upper layer integrity checks. See: [RFC6935][RFC6936] for further discussion.

For L2 encapsulations over IP, when the L2 source is also the OAL source it next copies the "Type of Service/Traffic Class" [RFC2983] and "Explicit Congestion Notification (ECN)" [RFC3168] values in the OAL header into the corresponding fields in the L2 IP header, then (for IPv6) set the L2 IPv6 header "Flow Label" as specified in [RFC6438]. The L2 source then sets the L2 IP TTL/Hop Limit the same as for any host (i.e., it does not copy the Hop Limit value from the OAL header) and finally sets the source and destination IP addresses to direct the carrier packet to the next hop. For carrier packets undergoing re-encapsulation, the OAL intermediate node L2 source decrements the OAL header Hop Limit and discards the carrier packet

if the value reaches 0. The L2 source then copies the "Type of Service/Traffic Class" and "Explicit Congestion Notification (ECN)" values from the previous hop L2 encapsulation header into the OAL header (if present), then finally sets the source and destination IP addresses the same as above.

Following L2 encapsulation/re-encapsulation, the L2 source forwards the resulting carrier packets over one or more underlay interfaces. The underlay interfaces often connect directly to physical media on the local platform (e.g., a laptop computer with WiFi, etc.), but in some configurations the physical media may be hosted on a separate Local Area Network (LAN) node. In that case, the OMNI interface can establish a Layer-2 VLAN or a point-to-point tunnel (at a layer below the underlay interface) to the node hosting the physical media. The OMNI interface may also apply encapsulation at the underlay interface layer (e.g., as for a tunnel virtual interface) such that carrier packets would appear "double-encapsulated" on the LAN; the node hosting the physical media in turn removes the LAN encapsulation prior to transmission or inserts it following reception. Finally, the underlay interface must monitor the node hosting the physical media (e.g., through periodic keepalives) so that it can convey up/down/status information to the OMNI interface.

6.3. OAL L2 Decapsulation and Reassembly

When an OMNI interface receives a carrier packet from an underlay interface, it copies the ECN value from the L2 encapsulation headers into the OAL header if the carrier packet contains a first-fragment. The OMNI interface next discards the L2 encapsulation headers and examines the OAL header of the enclosed OAL fragment. If the OAL fragment is addressed to a different node, the OMNI interface (acting as an OAL intermediate node) re-encapsulates and forwards while decrementing the OAL Hop Limit as discussed in Section 6.2. If the OAL fragment is addressed to itself, the OMNI interface (acting as an OAL destination) accepts or drops the fragment based on the (Source, Destination, Identification)-tuple and/or integrity checks.

The OAL destination next drops all non-final OAL fragments smaller than the minimum MPS and all fragments that would overlap or leave "holes" smaller than the minimum MPS with respect to other fragments already received. The OAL destination updates a checklist of accepted fragments of the same OAL packet that include an Ordinal number (i.e., Ordinals 0 through 127), but admits all accepted fragments into the reassembly cache after first removing any extension headers except for the fragment header itself. When the OAL destination receives the final fragment (i.e., the one with More Fragments set to 0), it caches the trailing checksum and reduces the Payload Length by 2. When reassembly is complete, the OAL

destination verifies the OAL packet checksum and discards the packet if the checksum is incorrect. If the OAL packet was accepted, the OAL destination finally removes the OAL headers and delivers the original IP packet to the network layer.

Carrier packets often travel over paths where all links in the path include CRC-32 integrity checks for effective hop-by-hop error detection for payload sizes up to 9180 octets [CRC], but other paths may traverse links (such as tunnels over IPv4) that do not include adequate integrity protection. The OAL checksum therefore allows OAL destinations to detect reassembly misassociation splicing errors and/or carrier packet corruption caused by unprotected links [CKSUM].

The OAL checksum also provides algorithmic diversity with respect to both lower layer CRCs and upper layer Internet checksums as part of a complimentary multi-layer integrity assurance architecture. Any corruption not detected by lower layer integrity checks is therefore very likely to be detected by upper layer integrity checks that use diverse algorithms.

6.4. OAL Header Compression

OAL sources that send carrier packets with full OAL headers include a CRH-32 extension for segment-by-segment forwarding based on a Multilink Forwarding Information Base (MFIB) in each OAL intermediate node. OAL source, intermediate and destination nodes can instead establish header compression state through IPv6 ND NS/NA message exchanges. After an initial NS/NA exchange, OAL nodes can apply OAL Header Compression to significantly reduce encapsulation overhead.

Each OAL node establishes MFIB soft state entries known as Multilink Forwarding Vectors (MVF's) which support both carrier packet forwarding and OAL header compression/decompression. For OAL sources, each MFV is referenced by a single Multilink Forwarding Vector Index (MFVI) that provides compression/decompression and forwarding context for the next hop. For OAL destinations, the MFV is referenced by a single MFVI that provides context for the previous hop. For OAL intermediate nodes, the MFV is referenced by two MFVIs - one for the previous hop and one for the next hop.

When an OAL node forwards carrier packets to a next hop, it can include a full OAL header with a CRH-32 extension containing one or more MVFIs. Whenever possible, however, the OAL node should instead omit significant portions of the OAL header (including the CRH-32) while applying OAL header compression. The full or compressed OAL header follows immediately after the innermost L2 encapsulation (i.e., UDP, IP or L2) as discussed in Section 6.2. Two OAL compressed header types (Types '0' and '1') are currently specified below (note that the (A)RQ flag is always considered set and therefore omitted from the compressed headers themselves).

For OAL first-fragments (including atomic fragments), the OAL node uses OMNI Compressed Header - Type 0 (OCH-0) format as shown in Figure 7:

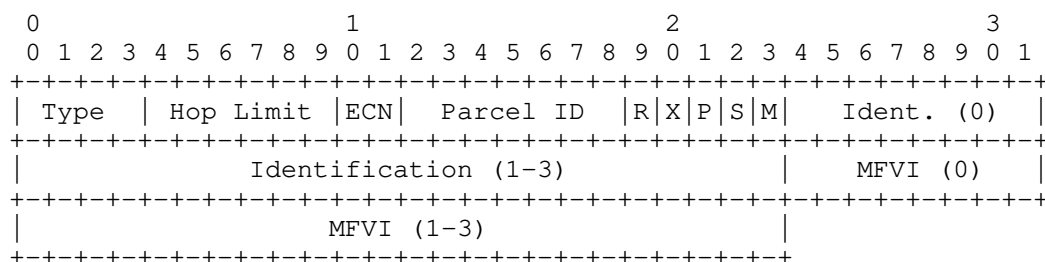


Figure 7: OMNI Compressed Header - Type 0 (OCH-0)

The format begins with a 4-bit Type, a 6-bit Hop Limit, a 2-bit Explicit Congestion Notification (ECN) field, a 7-bit Parcel ID and 5 flag bits. The format concludes with a 4-octet Identification field followed (optionally) by a 4-octet MFVI field. The OAL node sets Type to the value 0, sets Hop Limit to the minimum of the uncompressed OAL header Hop Limit and 63, sets ECN the same as for an uncompressed OAL header, and sets (P)arcel, (S)ub-parcels, (M)ore Fragments, Identification) the same as for an uncompressed fragment header. The OAL node finally sets Inde(X) and includes an MFVI if necessary; otherwise, it clears Inde(X) and omits the MFVI. (The (R)eserved flag is set to 0 on transmission and ignored on reception.)

The OAL first fragment (beginning with the original IP header) is then included immediately following the OCH-0 header, and the L2 header length field is reduced by the difference in length between the compressed headers and full-length OAL IPv6 and Fragment headers. The OAL destination can therefore determine the Payload Length by examining the L2 header length field and/or the length field(s) in the original IP header. The OCH-0 format applies for first fragments only, which are always regarded as ordinal fragment 0 even though no explicit Ordinal field is included. The (A)RQ flag is always implicitly set, and therefore omitted from the OCH-0 header.

For OAL non-first fragments (i.e., those with non-zero Fragment Offsets), the OAL uses OMNI Compressed Header - Type 1 (OCH-1) format as shown in Figure 8:

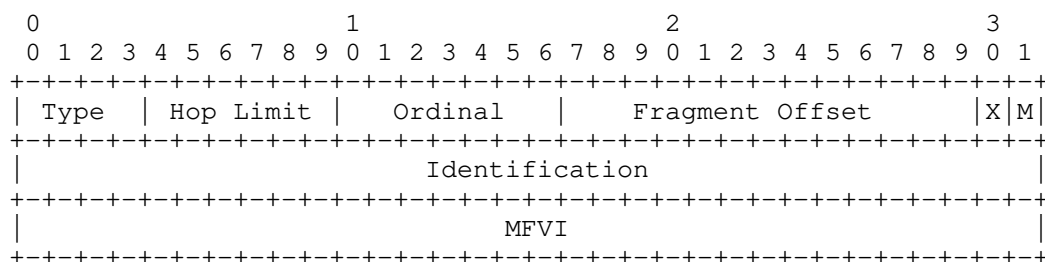


Figure 8: OMNI Compressed Header - Type 1 (OCH-1)

The format begins with a 4-bit Type, a 6-bit Hop Limit, a 7-bit Ordinal, a 13-bit Fragment Offset and 2 flag bits. The format concludes with a 4-octet Identification field followed (optionally) by a 4-octet MFVI field. The OAL node sets Type to the value 1, sets Hop Limit to the minimum of the uncompressed OAL header Hop Limit and 63, and sets (Ordinal, Fragment Offset, (M)ore Fragments, Identification) the same as for an uncompressed fragment header. If an MFVI is needed, the OAL node finally sets Inde(X) and includes an MFVI; otherwise, the node clears Inde(X) and omits the MFVI.

The OAL non-first fragment body is then included immediately following the OCH-1 header, and the L2 header length field is reduced by the difference in length between the compressed headers and full-length OAL IPv6 and Fragment headers. The OAL destination will then be able to determine the Payload Length by examining the L2 header length field. The OCH-1 format applies for non-first fragments only; therefore, the OAL source sets Ordinal to a monotonically increasing value beginning with 1 for the first non-first fragment, 2 for the second non-first fragment, etc., up to and including the final fragment. If more than 127 non-first fragments are included, these additional fragments instead set Ordinal to 0. The (A)RQ flag is always implicitly set, and therefore omitted from the OCH-1 header.

When an OAL destination or intermediate node receives a carrier packet, it determines the length of the encapsulated OAL information by examining the length field of the innermost L2 header, verifies that the innermost next header field indicates OMNI (see: Section 6.2), then examines the first four bits immediately following the innermost header. If the bits contain the value 4 or 6, the OAL node processes the remainder as an uncompressed OAL/IP header. If the bits contain a value 0 or 1, the OAL node instead processes the remainder of the header as an OCH-0/1 as specified above.

For carrier packets with OCH or full OAL headers addressed to itself and with CRH-32 extensions, the OAL node then uses the MFVI to locate the cached MFV which determines the next hop. During forwarding, the OAL node changes the MFVI to the cached value for the MVE next hop. If the OAL node is the destination, it instead reconstructs the full OAL headers then adds the resulting OAL fragment to the reassembly cache if the Identification is acceptable. (Note that for carrier packets that include an OCH-0 with both the X and M flags set to 0, the OAL node can instead locate forwarding state by examining the original IP packet header information that appears immediately after the OCH-0 header.)

Note: OAL header compression does not interfere with checksum calculation and verification, which must be applied according to the full OAL pseudo-header per Section 6.1 even when compression is used.

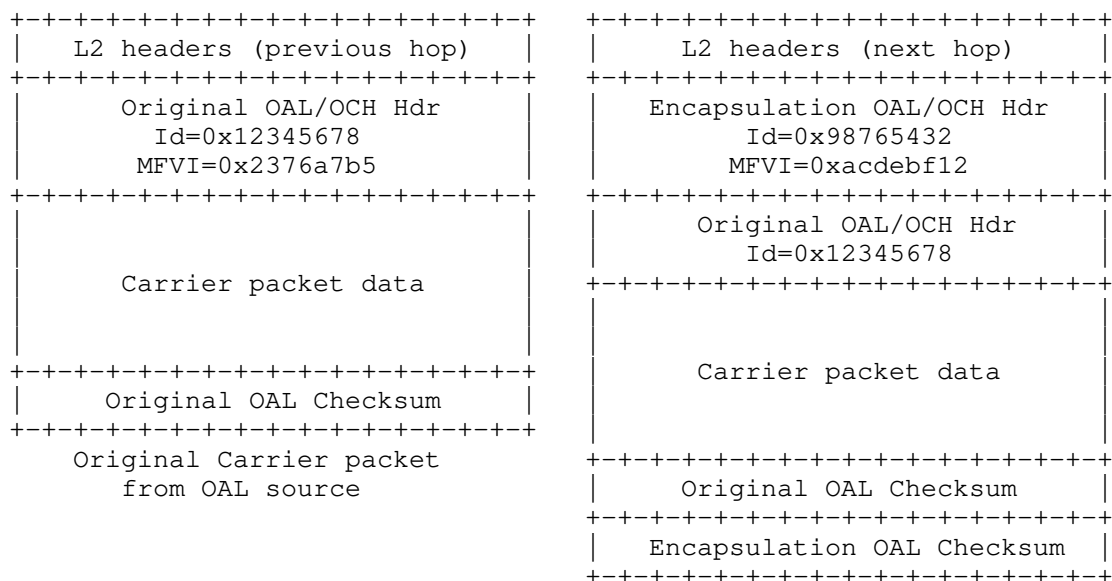
Note: The OCH-0/1 formats do not include the Traffic Class and Flow Label information that appears in uncompressed OAL IPv6 headers. Therefore, when OAL header compression state is initialized the Traffic Class and Flow Label are considered fixed for as long as the flow uses OCH-0/1 headers. If the flow requires frequent changes to Traffic Class and/or Flow Label information, it can include uncompressed OAL headers either continuously or periodically to update header compression state.

6.5. OAL-in-OAL Encapsulation

When an OAL source is unable to forward carrier packets directly to an OAL destination without "tunneling" through a pair of OAL intermediate nodes, the OAL source must regard the intermediate nodes as ingress and egress tunnel endpoints. This will result in nested OAL-in-OAL encapsulation in which the OAL source performs fragmentation on the inner OAL packet then forwards the fragments to the ingress tunnel endpoint which encapsulates each resulting OAL fragment in an additional OAL header before performing fragmentation following encapsulation.

For example, if the OAL source has an NCE for the OAL destination with MFVI 0x2376a7b5 and Identification 0x12345678 and the OAL ingress tunnel endpoint has an NCE for the OAL egress tunnel endpoint with MFVI 0xacdeb12 and Identification 0x98765432, the OAL source prepares the carrier packets using compressed/uncompressed OAL headers that include the MFVI and Identification corresponding to the OAL destination and with L2 header information addressed to the next hop toward the ingress tunnel endpoint. When the ingress tunnel endpoint receives the carrier packet, it recognizes the current MFVI included by the OAL source and determines the correct next hop MFVI.

The ingress tunnel endpoint then discards the L2 headers from the previous hop and encapsulates the original compressed/uncompressed OAL header within a second compressed/uncompressed OAL header while including the next-hop MFVI in the outer OAL encapsulation header and omitting the MFVI in the inner header. The ingress tunnel endpoint then includes L2 encapsulation headers with destinations appropriate for the next hop on the path to the egress tunnel endpoint. The encapsulation appears as shown in Figure 9:



Carrier packet following OAL ingress
(re)encapsulation before fragmentation

Figure 9: Carrier Packet in Carrier Packet Encapsulation

Note that only a single OAL-in-OAL encapsulation layer is supported, and that MFVIs appear only in the outer OAL header (i.e., either within a CRH-32 routing header when a full OAL header is used or within an OCH header with X set to 0). The inner OAL header should omit the CRH-32 header or use an OCH header with X set to 1, respectively.

Note that OAL/OCH encapsulation may cause the payloads of OAL packets produced by the ingress tunnel endpoint to exceed the minimum MPS by a small amount. If the ingress has assurance that the path to the egress will include only links capable of transiting the resulting (slightly larger) carrier packets it should forward without further fragmentation. Otherwise, the ingress must perform fragmentation following encapsulation to produce two fragments such that the size of the first fragment matches the size of the original OAL packet, and with the remainder in a second fragment. The egress tunnel endpoint must then reassemble then decapsulate to arrive at the original OAL packet which is then subject to further forwarding.

6.6. OAL Identification Window Maintenance

The OAL encapsulates each original IP packet as an OAL packet then performs fragmentation to produce one or more carrier packets with the same 32-bit Identification value. In environments where spoofing is not considered a threat, OMNI interfaces send OAL packets with Identifications beginning with an unpredictable Initial Send Sequence (ISS) value [RFC7739] monotonically incremented (modulo 2^{32}) for each successive OAL packet sent to either a specific neighbor or to any neighbor. (The OMNI interface may later change to a new unpredictable ISS value as long as the Identifications are assured unique within a timeframe that would prevent the fragments of a first OAL packet from becoming associated with the reassembly of a second OAL packet.) In other environments, OMNI interfaces should maintain explicit per-neighbor send and receive windows to detect and exclude spurious carrier packets that might clutter the reassembly cache as discussed below.

OMNI interface neighbors use TCP-like synchronization to maintain windows with unpredictable ISS values incremented (modulo 2^{32}) for each successive OAL packet and re-negotiate windows often enough to maintain an unpredictable profile. OMNI interface neighbors exchange IPv6 ND messages with OMNI options that include TCP-like information fields to manage streams of OAL packets instead of streams of octets. As a link-layer service, the OAL provides low-persistence best-effort retransmission with no mitigations for duplication, reordering or deterministic delivery. Since the service model is best-effort and only control message sequence numbers are acknowledged, OAL nodes can select unpredictable new initial sequence numbers outside of the current window without delaying for the Maximum Segment Lifetime (MSL).

OMNI interface neighbors maintain current and previous window state in IPv6 ND neighbor cache entries (NCEs) to support dynamic rollover to a new window while still sending OAL packets and accepting carrier packets from the previous windows. Each NCE is indexed by the neighbor's ULA, while the OAL encapsulation ULA (which may be different) provides context for Identification verification. OMNI interface neighbors synchronize windows through asymmetric and/or symmetric IPv6 ND message exchanges. When a node receives an IPv6 ND message with new window information, it resets the previous window state based on the current window then resets the current window based on new and/or pending information.

The IPv6 ND message OMNI option header extension sub-option includes TCP-like information fields including Sequence Number, Acknowledgement Number, Window and flags (see: Section 12). OMNI interface neighbors maintain the following TCP-like state variables in the NCE:

Send Sequence Variables (current, previous and pending)

- SND.NXT - send next
- SND.WND - send window
- ISS - initial send sequence number

Receive Sequence Variables (current and previous)

- RCV.NXT - receive next
- RCV.WND - receive window
- IRS - initial receive sequence number

OMNI interface neighbors "OAL A" and "OAL B" exchange IPv6 ND messages per [RFC4861] with OMNI options that include TCP-like information fields. When OAL A synchronizes with OAL B, it maintains both a current and previous SND.WND beginning with a new unpredictable ISS and monotonically increments SND.NXT for each successive OAL packet transmission. OAL A initiates synchronization by including the new ISS in the Sequence Number of an authentic IPv6 ND message with the SYN flag set and with Window set to M (up to $2^{*}24$) as a tentative receive window size while creating a NCE in the INCOMPLETE state if necessary. OAL A caches the new ISS as pending, uses the new ISS as the Identification for OAL encapsulation, then sends the resulting OAL packet to OAL B and waits up to RetransTimer milliseconds to receive an IPv6 ND message response with the ACK flag set (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).

When OAL B receives the SYN, it creates a NCE in the STALE state if necessary, resets its RCV variables, caches the tentative (send) window size M, and selects a (receive) window size N (up to $2^{*}24$) to indicate the number of OAL packets it is willing to accept under the current RCV.WND. (The RCV.WND should be large enough to minimize control message overhead yet small enough to provide an effective filter for spurious carrier packets.) OAL B then prepares an IPv6 ND message with the ACK flag set, with the Acknowledgement Number set to OAL A's next sequence number, and with Window set to N. Since OAL B does not assert an ISS of its own, it uses the IRS it has cached for OAL A as the Identification for OAL encapsulation then sends the ACK to OAL A.

When OAL A receives the ACK, it notes that the Identification in the OAL header matches its pending ISS. OAL A then sets the NCE state to REACHABLE and resets its SND variables based on the Window size and Acknowledgement Number (which must include the sequence number following the pending ISS). OAL A can then begin sending OAL packets to OAL B with Identification values within the (new) current SND.WND for up to ReachableTime milliseconds or until the NCE is updated by a new IPv6 ND message exchange. This implies that OAL A must send a new SYN before sending more than N OAL packets within the current SND.WND, i.e., even if ReachableTime is not nearing expiration. After OAL B returns the ACK, it accepts carrier packets received from OAL A within either the current or previous RCV.WND as well as any new authentic NS/RS SYN messages received from OAL A even if outside the windows.

OMNI interface neighbors can employ asymmetric window synchronization as described above using two independent (SYN -> ACK) exchanges (i.e., a four-message exchange), or they can employ symmetric window synchronization using a modified version of the TCP three-way handshake as follows:

- * OAL A prepares a SYN with an unpredictable ISS not within the current SND.WND and with Window set to M as a tentative receive window size. OAL A caches the new ISS and Window size as pending information, uses the pending ISS as the Identification for OAL encapsulation, then sends the resulting OAL packet to OAL B and waits up to RetransTimer milliseconds to receive an ACK response (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).
- * OAL B receives the SYN, then resets its RCV variables based on the Sequence Number while caching OAL A's tentative receive Window size M and a new unpredictable ISS outside of its current window as pending information. OAL B then prepares a response with Sequence Number set to the pending ISS and Acknowledgement Number set to OAL A's next sequence number. OAL B then sets both the SYN and ACK flags, sets Window to N and sets the OPT flag according to whether an explicit concluding ACK is optional or mandatory. OAL B then uses the pending ISS as the Identification for OAL encapsulation, sends the resulting OAL packet to OAL A and waits up to RetransTimer milliseconds to receive an acknowledgement (retransmitting up to MAX_UNICAST_SOLICIT times if necessary).
- * OAL A receives the SYN/ACK, then resets its SND variables based on the Acknowledgement Number (which must include the sequence number following the pending ISS) and OAL B's advertised Window N. OAL A then resets its RCV variables based on the Sequence Number and marks the NCE as REACHABLE. If the OPT flag is clear, OAL A next prepares an immediate solicited NA message with the ACK flag set,

the Acknowledgement Number set to OAL B's next sequence number, with Window set a value that may be the same as or different than M, and with the OAL encapsulation Identification to SND.NXT, then sends the resulting OAL packet to OAL B. If the OPT flag is set and OAL A has OAL packets queued to send to OAL B, it can optionally begin sending their carrier packets under the (new) current SND.WND as implicit acknowledgements instead of returning an explicit ACK. In that case, the tentative Window size M becomes the current receive window size.

- * OAL B receives the implicit/explicit acknowledgement(s) then resets its SND state based on the pending/advertised values and marks the NCE as REACHABLE. If OAL B receives an explicit acknowledgement, it uses the advertised Window size and abandons the tentative size. (Note that OAL B sets the OPT flag in the SYN/ACK to assert that it will interpret timely receipt of carrier packets within the (new) current window as an implicit acknowledgement. Potential benefits include reduced delays and control message overhead, but use case analysis is outside the scope of this specification.)

Following synchronization, OAL A and OAL B hold updated NCEs and can exchange OAL packets with Identifications set to SND.NXT while the state remains REACHABLE and there is available window capacity. Either neighbor may at any time send a new SYN to assert a new ISS. For example, if OAL A's current SND.WND for OAL B is nearing exhaustion and/or ReachableTime is nearing expiration, OAL A continues to send OAL packets under the current SND.WND while also sending a SYN with a new unpredictable ISS. When OAL B receives the SYN, it resets its RCV variables and may optionally return either an asymmetric ACK or a symmetric SYN/ACK to also assert a new ISS. While sending SYNs, both neighbors continue to send OAL packets with Identifications set to the current SND.NXT then reset the SND variables after an acknowledgement is received.

While the optimal symmetric exchange is efficient, anomalous conditions such as receipt of old duplicate SYNs can cause confusion for the algorithm as discussed in Section 3.4 of [RFC0793]. For this reason, the OMNI option header includes an RST flag which OAL nodes set in solicited NA responses to ACKs received with incorrect acknowledgement numbers. The RST procedures (and subsequent synchronization recovery) are conducted exactly as specified in [RFC0793].

OMNI interfaces may set the PNG ("ping") flag when a reachability confirmation outside the context of the IPv6 ND protocol is needed (OMNI interfaces therefore most often set the PNG flag in advertisement messages and ignore it in solicitation messages). When

an OMNI interface receives a PNG, it returns an unsolicited NA (uNA) ACK with the PNG message Identification in the Acknowledgment, but without updating RCV state variables. OMNI interfaces return unicast uNA ACKs even for multicast PNG destination addresses, since OMNI link multicast is based on unicast emulation.

OMNI interfaces that employ the window synchronization procedures described above observe the following requirements:

- * OMNI interfaces MUST select new unpredictable ISS values that are at least a full window outside of the current SND.WND.
- * OMNI interfaces MUST set the initial SYN message Window field to a tentative value to be used only if no concluding NA ACK is sent.
- * OMNI interfaces that receive advertisements with the PNG and/or SYN flag set MUST NOT set the PNG and/or SYN flag in uNA responses.
- * OMNI interfaces that send advertisements with the PNG and/or SYN flag set MUST ignore uNA responses with the PNG and/or SYN flag set.
- * OMNI interfaces MUST send IPv6 ND messages used for window synchronization securely while using unpredictable initial Identification values until synchronization is complete.

Note: Although OMNI interfaces employ TCP-like window synchronization and support uNA ACK responses to SYNs and PNGs, all other aspects of the IPv6 ND protocol (e.g., control message exchanges, NCE state management, timers, retransmission limits, etc.) are honored exactly per [RFC4861].

Note: Recipients of OAL-encapsulated IPv6 ND messages index the NCE based on the message source address, which also determines the carrier packet Identification window. However, IPv6 ND messages may contain a message source address that does not match the OMNI encapsulation source address when the recipient acts as a proxy.

Note: OMNI interface neighbors apply the same send and receive windows for all of their (multilink) underlay interface pairs that exchange carrier packets. Each interface pair represents a distinct underlay network path, and the set of paths traversed may be highly diverse when multiple interface pairs are used. OMNI intermediate nodes therefore SHOULD NOT cache window synchronization parameters in IPv6 ND messages they forward since there is no way to ensure network-wide middlebox state consistency.

6.7. OAL Fragment Retransmission

When the OAL source sends carrier packets to an OAL destination, it should cache recently sent packets in case timely best-effort selective retransmission is requested. The OAL destination in turn maintains a checklist for the (Source, Destination, Identification)-tuple of recently received carrier packets and notes the ordinal numbers of OAL packet fragments already received (i.e., as Frag #0, Frag #1, Frag #2, etc.). The timeframe for maintaining the OAL source and destination caches determines the link persistence (see: [RFC3366]).

If the OAL destination notices some fragments missing after most other fragments within the same link persistence timeframe have already arrived, it may issue an Automatic Repeat Request (ARQ) with Selective Repeat (SR) by sending a uNA message to the OAL source. The OAL destination creates a uNA message with an OMNI option with one or more Fragmentation Report (FRAGREP) sub-options that include a list of (Identification, Bitmap)-tuples for fragments received and missing from this OAL source (see: Section 12 and [I-D.templin-6man-fragrep]). The OAL destination includes an authentication signature if necessary, performs OAL encapsulation (with the its own address as the OAL source and the source address of the message that prompted the uNA as the OAL destination) and sends the message to the OAL source.

When the OAL source receives the uNA message, it authenticates the message then examines the FRAGREP. For each (Source, Destination, Identification)-tuple, the OAL source determines whether it still holds the corresponding carrier packets in its cache and retransmits any for which the Bitmap indicates a loss event. For example, if the Bitmap indicates that ordinal fragments #3, #7, #10 and #13 from the OAL packet with Identification 0x12345678 are missing the OAL source only retransmits carrier packets containing those fragments. When the OAL destination receives the retransmitted carrier packets, it admits the enclosed fragments into the reassembly cache and updates its checklist. If some fragments are still missing, the OAL destination may send a small number of additional uNA ARQ/SRs within the link persistence timeframe.

The OAL therefore provides a link-layer low-to-medium persistence ARQ/SR service consistent with [RFC3366] and Section 8.1 of [RFC3819]. The service provides the benefit of timely best-effort link-layer retransmissions which may reduce packet loss and avoid some unnecessary end-to-end delays. This best-effort network-based service therefore compliments higher layer end-to-end protocols responsible for true reliability.

6.8. OAL MTU Feedback Messaging

When the OMNI interface forwards original IP packets from the network layer, it invokes the OAL and returns internally-generated ICMPv4 Fragmentation Needed [RFC1191] or ICMPv6 Path MTU Discovery (PMTUD) Packet Too Big (PTB) [RFC8201] messages as necessary. This document refers to both of these ICMPv4/ICMPv6 message types simply as "PTBs", and introduces a distinction between PTB "hard" and "soft" errors as discussed below and also in [I-D.templin-6man-fragrep].

Ordinary PTB messages with ICMPv4 header "unused" field or ICMPv6 header Code field value 0 are hard errors that always indicate that a packet has been dropped due to a real MTU restriction. However, the OMNI interface can also forward large original IP packets via OAL encapsulation and fragmentation while at the same time returning PTB soft error messages (subject to rate limiting) if it deems the original IP packet too large according to factors such as link performance characteristics, number of fragments needed, reassembly congestion, etc. This ensures that the path MTU is adaptive and reflects the current path used for a given data flow. The OMNI interface can therefore continuously forward packets without loss while returning PTB soft error messages recommending a smaller size if necessary. Original sources that receive the soft errors in turn reduce the size of the packets they send (i.e., the same as for hard errors), but can soon resume sending larger packets if the soft errors subside.

An OAL source sends PTB soft error messages by setting the ICMPv4 header "unused" field or ICMPv6 header Code field to the value 1 if the packet was dropped or 2 if the packet was forwarded successfully. The OAL source sets the PTB destination address to the original IP packet source, and sets the source address to one of its OMNI interface addresses that is routable from the perspective of the original source. The OAL source then sets the MTU field to a value smaller than the original packet size but no smaller than 576 for ICMPv4 or 1280 for ICMPv6, writes the leading portion of the original IP packet first fragment into the "packet in error" field, and returns the PTB soft error to the original source. When the original source receives the PTB soft error, it temporarily reduces the size of the packets it sends the same as for hard errors but may seek to increase future packet sizes dynamically while no further soft errors are arriving. (If the original source does not recognize the soft error code, it regards the PTB the same as a hard error but should heed the retransmission advice given in [RFC8201] suggesting retransmission based on normal packetization layer retransmission timers.)

An OAL destination may experience reassembly cache congestion, and can return uNA messages to the OAL source that originated the fragments (subject to rate limiting) that include OMNI encapsulated PTB messages with code 1 or 2. The OAL destination creates a uNA message with an OMNI option containing an authentication message sub-option if necessary followed optionally by a ICMPv6 Error sub-option that encodes a PTB message with a reduced value and with the leading portion an OAL first fragment containing the header of an original IP packet whose source must be notified (see: Section 12). The OAL destination encapsulates the leading portion of the OAL first fragment (beginning with the OAL header) in the PTB "packet in error" field, signs the message if an authentication sub-option is included, performs OAL encapsulation (with the its own address as the OAL source and the source address of the message that prompted the uNA as the OAL destination) and sends the message to the OAL source.

When the OAL source receives the uNA message, it sends a corresponding network layer PTB soft error to the original source to recommend a smaller size. The OAL source crafts the PTB by extracting the leading portion of the original IP packet from the OMNI encapsulated PTB message (i.e., not including the OAL header) and writes it in the "packet in error" field of a network layer PTB with destination set to the original IP packet source and source set to one of its OMNI interface addresses that is routable from the perspective of the original source.

Original sources that receive PTB soft errors can dynamically tune the size of the original IP packets they to send to produce the best possible throughput and latency, with the understanding that these parameters may change over time due to factors such as congestion, mobility, network path changes, etc. The receipt or absence of soft errors should be seen as hints of when increasing or decreasing packet sizes may be beneficial. The OMNI interface supports continuous transmission and reception of packets of various sizes in the face of dynamically changing network conditions. Moreover, since PTB soft errors do not indicate a hard limit, original sources that receive soft errors can resume sending larger packets without waiting for the recommended 10 minutes specified for PTB hard errors [RFC1191][RFC8201]. The OMNI interface therefore provides an adaptive service that accommodates MTU diversity especially well-suited for dynamic multilink environments.

6.9. OAL Super-Packets

By default, the OAL source includes a 40-octet IPv6 encapsulation header for each original IP packet during OAL encapsulation. The OAL source also calculates then performs fragmentation such that a copy of the 40-octet IPv6 header plus an 8-octet IPv6 Fragment Header is included in each OAL fragment (when a Routing Header is added, the OAL encapsulation headers become larger still). However, these encapsulations may represent excessive overhead in some environments. OAL header compression can dramatically reduce the amount of encapsulation overhead, however a complimentary technique known as "packing" (see: [I-D.ietf-intarea-tunnels]) supports encapsulation of multiple original IP packets and/or control messages within a single OAL "super-packet".

When the OAL source has multiple original IP packets to send to the same OAL destination with total length no larger than the OAL destination MRU, it can concatenate them into a super-packet encapsulated in a single OAL header. Within the OAL super-packet, the IP header of the first original IP packet (iHa) followed by its data (iDa) is concatenated immediately following the OAL header, then the IP header of the next original packet (iHb) followed by its data (iDb) is concatenated immediately following the first original packet, etc. with a trailing checksum field included in the final fragment. The OAL super-packet format is transposed from [I-D.ietf-intarea-tunnels] and shown in Figure 10:

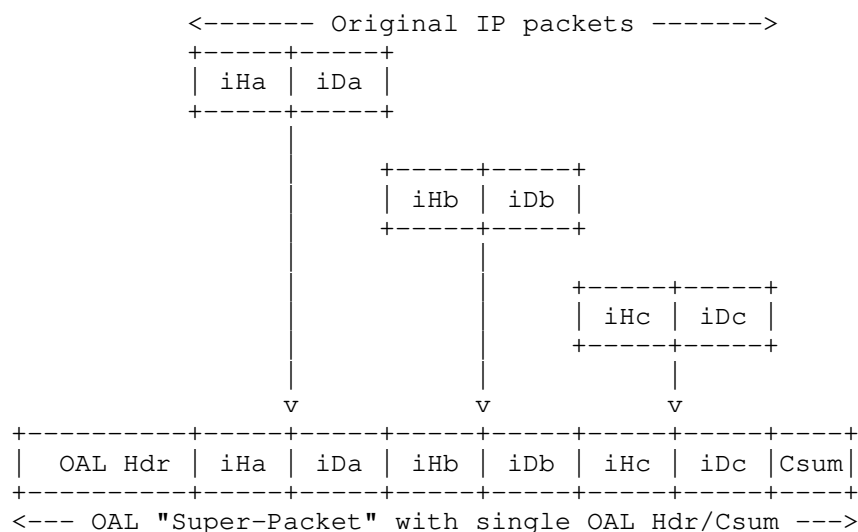


Figure 10: OAL Super-Packet Format

When the OAL source prepares a super-packet, it applies OAL fragmentation, includes a trailing checksum in the final fragment, applies L2 encapsulation to each fragment then sends the resulting carrier packets to the OAL destination. When the OAL destination receives the super-packet it sets aside the trailing checksum, reassembles if necessary, then verifies the checksum while regarding the remaining OAL header Payload Length as the sum of the lengths of all payload packets. The OAL destination then selectively extracts each original IP packet (e.g., by setting pointers into the super-packet buffer and maintaining a reference count, by copying each packet into a separate buffer, etc.) and forwards each packet to the network layer. During extraction, the OAL determines the IP protocol version of each successive original IP packet 'j' by examining the four most-significant bits of iH(j), and determines the length of the packet by examining the rest of iH(j) according to the IP protocol version.

When an OAL source prepares a super-packet that includes an IPv6 ND message with an authentication signature or ICMPv6 message checksum as the first original IP packet (i.e., iHa/iDa), it calculates the authentication signature or checksum over the remainder of super-packet. Security and integrity for forwarding initial protocol data packets in conjunction with IPv6 ND messages used to establish NCE state are therefore supported. (A common use case entails a path MPS probe beginning with a signed IPv6 ND message followed by a NULL IPv6 packet with a suitably large (Jumbo) Payload Length but with Next Header set to 59 for "No Next Header".)

6.10. OAL Bubbles

OAL sources may send NULL OAL packets known as "bubbles" for the purpose of establishing Network Address Translator (NAT) state on the path to the OAL destination. The OAL source prepares a bubble by crafting an OAL header with appropriate IPv6 source and destination ULAs, with the IPv6 Next Header field set to the value 59 ("No Next Header" - see [RFC8200]) and with only the trailing OAL Checksum field (i.e., and no protocol data) immediately following the IPv6 header.

The OAL source includes a random Identification value then encapsulates the OAL packet in L2 headers destined to either the mapped address of the OAL destination's first-hop ingress NAT or the L2 address of the OAL destination itself. When the OAL source sends the resulting carrier packet, any egress NATs in the path toward the L2 destination will establish state based on the activity but the bubble will be harmlessly discarded by either an ingress NAT on the path to the OAL destination or by the OAL destination itself.

The bubble concept for establishing NAT state originated in [RFC4380] and was later updated by [RFC6081]. OAL bubbles may be employed by mobility services such as [I-D.templin-6man-aero].

6.11. OAL Requirements

In light of the above, OAL sources, destinations and intermediate nodes observe the following normative requirements:

- * OAL sources MUST forward original IP packets either larger than the OMNI interface MRU or smaller than the minimum MPS minus the trailing checksum size as atomic fragments (i.e., and not as multiple fragments).
- * OAL sources MUST produce non-final fragments with payloads no smaller than the minimum MPS during fragmentation.
- * OAL intermediate nodes SHOULD and OAL destinations MUST unconditionally drop any non-final OAL fragments with payloads smaller than the minimum MPS.
- * OAL destinations MUST drop any new OAL fragments with offset and length that would overlap with other fragments and/or leave holes smaller than the minimum MPS between fragments that have already been received.

Note: Under the minimum MPS, ordinary 1500 octet original IP packets would require at most 4 OAL fragments, with each non-final fragment containing 400 payload octets and the final fragment containing 302 payload octets (i.e., the final 300 octets of the original IP packet plus the 2 octet trailing checksum). For all packet sizes, the likelihood of successful reassembly may improve when the OMNI interface sends all fragments of the same fragmented OAL packet consecutively over the same underlay interface pair instead of spread across multiple underlay interface pairs. Finally, an assured minimum/path MPS allows continuous operation over all paths including those that traverse bridged L2 media with dissimilar MTUs.

Note: Certain legacy network hardware of the past millennium was unable to accept packet "bursts" resulting from an IP fragmentation event - even to the point that the hardware would reset itself when presented with a burst. This does not seem to be a common problem in the modern era, where fragmentation and reassembly can be readily demonstrated at line rate (e.g., using tools such as 'iperf3') even over fast links on ordinary hardware platforms. Even so, while the OAL destination is reporting reassembly congestion (see: Section 6.8) the OAL source could impose "pacing" by inserting an inter-fragment delay and increasing or decreasing the delay according to congestion indications.

6.12. OAL Fragmentation Security Implications

As discussed in Section 3.7 of [RFC8900], there are four basic threats concerning IPv6 fragmentation; each of which is addressed by effective mitigations as follows:

1. Overlapping fragment attacks - reassembly of overlapping fragments is forbidden by [RFC8200]; therefore, this threat does not apply to the OAL.
2. Resource exhaustion attacks - this threat is mitigated by providing a sufficiently large OAL reassembly cache and instituting "fast discard" of incomplete reassemblies that may be part of a buffer exhaustion attack. The reassembly cache should be sufficiently large so that a sustained attack does not cause excessive loss of good reassemblies but not so large that (timer-based) data structure management becomes computationally expensive. The cache should also be indexed based on the arrival underlay interface such that congestion experienced over a first underlay interface does not cause discard of incomplete reassemblies for uncongested underlay interfaces.
3. Attacks based on predictable fragment identification values - in environments where spoofing is possible, this threat is mitigated through the use of Identification windows beginning with unpredictable values per Section 6.6. By maintaining windows of acceptable Identifications, OAL neighbors can quickly discard spurious carrier packets that might otherwise clutter the reassembly cache. The OAL additionally provides an integrity check to detect corruption that may be caused by spurious fragments received with in-window Identification values.
4. Evasion of Network Intrusion Detection Systems (NIDS) - since the OAL source employs a robust MPS, network-based firewalls can inspect and drop OAL fragments containing malicious data thereby disabling reassembly by the OAL destination. However, since OAL

fragments may take different paths through the network (some of which may not employ a firewall) each OAL destination must also employ a firewall.

IPv4 includes a 16-bit Identification (IP ID) field with only 65535 unique values such that at high data rates the field could wrap and apply to new carrier packets while the fragments of old packets using the same IP ID are still alive in the network [RFC4963]. Since carrier packets sent via an IPv4 path with DF=0 are normally no larger than 576 octets, IPv4 fragmentation is possible only at small-MTU links in the path which should support data rates low enough for safe reassembly [RFC3819]. (IPv4 carrier packets larger than 576 octets with DF=0 may incur high data rate reassembly errors in the path, but the OAL checksum provides OAL destination integrity assurance.) Since IPv6 provides a 32-bit Identification value, IP ID wraparound at high data rates is not a concern for IPv6 fragmentation.

Fragmentation security concerns for large IPv6 ND messages are documented in [RFC6980]. These concerns are addressed when the OMNI interface employs the OAL instead of directly fragmenting the IPv6 ND message itself. For this reason, OMNI interfaces MUST NOT send IPv6 ND messages larger than the OMNI interface MTU, and MUST employ OAL encapsulation and fragmentation for IPv6 ND messages larger than the minimum/path MPS for this OAL destination.

Unless the path is secured at the network-layer or below (i.e., in environments where spoofing is possible), OMNI interfaces MUST NOT send ordinary carrier packets with Identification values outside the current window and MUST secure IPv6 ND messages used for address resolution or window state synchronization. OAL destinations SHOULD therefore discard without reassembling any out-of-window OAL fragments received over an unsecured path.

6.13. OMNI Hosts

OMNI Hosts are end systems that extend the OMNI link over ENET underlay interfaces (i.e., either as an OMNI interface or as a sublayer of the ENET interface itself). Each ENET is connected to the rest of the OMNI link by a Client that receives an MNP delegation. Clients delegate MNP addresses and/or sub-prefixes to ENET nodes (i.e., Hosts, other Clients, routers and non-OMNI hosts) using standard mechanisms such as DHCP [RFC8415][RFC2131] and IPv6 Stateless Address AutoConfiguration (SLAAC) [RFC4862]. Clients forward packets between their ENET Hosts and peers on external networks acting as routers and/or OAL intermediate nodes.

OMNI Hosts coordinate with Clients and/or other Hosts connected to the same ENET using IP-encapsulated IPv6 ND messages. The IP encapsulation headers and ND messages both use the MNP-based addresses assigned to ENET underlay interfaces as source and destination addresses (i.e., instead of ULAs). For IPv4 MNPs, the ND messages use IPv4-Compatible IPv6 addresses [RFC4291] in place of the IPv4 addresses. (Note that IPv4-Compatible IPv6 addresses are deprecated for all other uses by the aforementioned standard.)

Hosts discover Clients by sending encapsulated RS messages using an OMNI link IP anycast address (or the unicast address of the Client) as the RS L2 encapsulation destination as specified in Section 15. The Client configures the IPv4 and/or IPv6 anycast addresses for the OMNI link on its ENET interface and advertises the address(es) into the ENET routing system. The Client then responds to the encapsulated RS messages by sending an encapsulated RA message that uses its ENET unicast address as the source. (To differentiate itself from an INET border Proxy/Server, the Client sets the RA message OMNI Interface Attributes sub-option LHS field to 0 for the Host's interface index. When the RS message includes an L2 anycast destination address, the Client also includes an Interface Attributes sub-option for interface index 0 to inform the Host of its L2 unicast address - see: Section 15 for full details on the RS and RA message contents.)

Hosts coordinate with peer Hosts on the same ENET by sending encapsulated NS messages to receive an NA reply. (Hosts determine whether a peer is on the same ENET by matching the peer's IP address with the MNP (sub)-prefix for the ENET advertised in the Client's RA message [RFC8028].) Each ENET peer then creates a NCE and synchronizes Identification windows the same as for OMNI link neighbors, and the Host can then engage in OMNI link transactions with the Client and/or other ENET Hosts. By coordinating with the Client in this way, the Host treats the Client as if it were an ANET Proxy/Server, and the Client provides the same services that a Proxy/Server would provide. By coordinating with other Hosts, the peer hosts can exchange large IP packets or parcels over the ENET using IPv6 fragmentation if necessary.

When a Host prepares an IP packet or parcel, it uses the IP address of its native ENET interface as the source and the IP address of the (remote) peer as the destination. The Host next performs parcel segmentation if necessary (see: Section 6.14) then encapsulates the packet/parcel in an IP header of the version supported by the ENET while setting the source to the same address and destination to either the same address if the peer is on the local ENET, or to the IP address of the Client otherwise. The Host can then proceed to exchange packets/parcels with the destination, either directly or via the Client as an intermediate node.

The encapsulation procedures are coordinated per Section 6.1, except that the IP encapsulation header version matches the native ENET IP protocol version and uses IPv6 GUA or public/private IPv4 addresses instead of ULAs. The Host sets the encapsulation IP header {Protocol, Next-Header} field to TBD1 to indicate that this is an OAL encapsulation and not an ordinary IP-in-IP encapsulation. When the inner header is IPv4-based, the Host next translates the encapsulation header into an IPv6 header with IPv4-Compatible addresses while setting the [IPv6 Traffic Class, Payload Length, Next Header, Hop Limit] fields according to the IPv4 {Type of Service, Total Length, Protocol, TTL} fields, respectively, while setting Flow Label to 0. The Host then calculates an OAL checksum, writes the value as the final two octets of the encapsulated packet then applies IPv6 fragmentation to the encapsulated packet to produce IPv6 fragments no smaller than the MPS the same as described in Section 6.1. If the original encapsulation IP header was IPv4, the Host next translates the IPv6 encapsulation headers back to IPv4 headers with Protocol value set to 44 since the immediately next header is the IPv6 Fragment Header. The Host finally sends the IP encapsulated fragments to the ENET peer.

When the ENET peer receives IP encapsulated fragments, for IPv4 it first translates the encapsulation headers back to IPv6 headers with IPv4-Compatible addresses the same as above. The peer then reassembles and verifies the OAL checksum. If the checksum is correct, the peer next removes the encapsulation headers and applies parcel reassembly if necessary. The peer then either delivers the encapsulated packet/parcel to upper layers if the peer is the destination or forwards the packet/parcel toward the final destination if the peer is a Client acting as an intermediate node.

Hosts and Clients that initiate OMNI-based packet/parcel transactions should first test the path toward the final destination using the parcel path qualification procedure specified in [I-D.templin-intarea-parcels]. An OMNI Host that sends and receives parcels need not implement the full OMNI interface abstraction but MUST implement enough of the OAL to be capable of fragmenting and reassembling maximum-length encapsulated IP packets/parcels and sub-parcels as discussed above and in the following section.

6.14. IP Parcels

IP parcels are specified in [I-D.templin-intarea-parcels], while details for their application over OMNI interfaces is specified here. IP parcels are formed by an OMNI Host or Client upper layer protocol entity (identified by the "5-tuple" source IP address/port number, destination IP address/port number and protocol number) when it produces a protocol data unit containing the concatenation of up to 64 upper layer protocol segments. All non-final segments MUST be equal in length while the final segment MUST NOT be larger but MAY be smaller. Each non-final segment MUST be no larger than 65535 minus the length of the IP header plus extensions, minus the length of the OAL encapsulation header and trailer. The upper layer protocol then presents the buffer and non-final segment size to the IP layer which appends a single IP header (plus any extension headers) before presenting the parcel to the OMNI Interface.

For IPv4, the IP layer prepares the parcel by appending an IPv4 header with a Jumbo Payload option (see: Section 5.1) where "Jumbo Payload Length" is a 32-bit unsigned integer value (in network byte order) set to the lengths of the IPv4 header plus all concatenated segments. The IP layer next sets the IPv4 header DF bit to 1, then sets the IPv4 header Total Length field to the length of the IPv4 header plus the length of the first segment only. (Note: the IP layer can form true IPv4 jumbograms (as opposed to parcels) by instead setting the Total Length field to the length of the IPv4 header only.)

For IPv6, the IP layer forms a parcel by appending an IPv6 header with a Jumbo Payload option the same as for IPv4 above where "Jumbo Payload Length" is set to the lengths of the IPv6 Hop-by-Hop Options header and any other extension headers present plus all concatenated segments. The IP layer next sets the IPv6 header Payload Length field to the lengths of the IPv6 Hop-by-Hop Options header and any other extension headers present plus the length of the first segment only. (Note: the IP layer can form true IPv6 jumbograms (as opposed to parcels) by instead setting the Payload Length field to 0.)

An IP parcel therefore has the following structure:

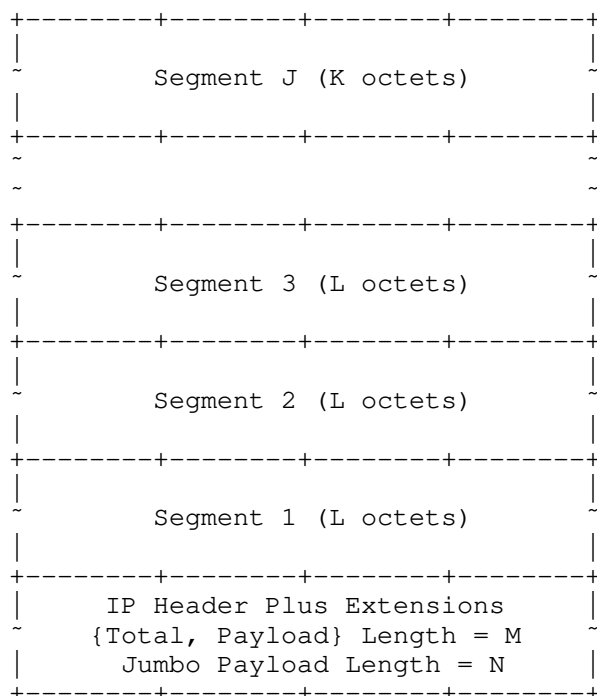


Figure 11: OMNI Interface IP Parcels

where J is the total number of segments (between 1 and 64), L is the length of each non-final segment which MUST NOT be larger than 65535 (minus headers as above) and K is the length of the final segment which MUST NOT be larger than L. The values M and N are then set to the length of the IP header plus extensions for IPv4 or to the length of the extensions only for IPv6, then further calculated as follows:

$$M = M + ((J-1) ? L : K)$$

$$N = N + (((J-1) * L) + K)$$

Note: a "singleton" parcel is one that includes only the IP header plus extensions with a single segment of length K, while a "null" parcel is a singleton with K=0, i.e., a parcel consisting of only the IP header plus extensions with no octets beyond.

When the IP layer forwards a parcel, the OMNI interface invokes the OAL which forwards it to either a Client as an intermediate node or the final destination itself. The OAL source first assigns a monotonically-incrementing (modulo 127) "Parcel ID" and subdivides the parcel into sub-parcels no larger than the maximum of the path

MTU to the next hop or 64KB (minus the length of encapsulation headers). The OAL source determines the number of segments of length L that can fit into each sub-parcel under these size constraints, e.g. if the OAL source determines that a sub-parcel can contain 3 segments of length L, it creates sub-parcels with the first containing segments 1-3, the second containing segments 4-6, etc. and with the final containing any remaining segments. The OAL source then appends an identical IP header plus extensions to each sub-parcel while resetting M and N in each according to the above equations with J set to 3 and K set to L for each non-final sub-parcel and with J set to the remaining number of segments for the final sub-parcel.

The OAL source next performs encapsulation on each sub-parcel with destination set to the next hop address. If the next hop is reached via an ANET/INET interface, the OAL source inserts an OAL header the same as discussed in Section 6.1 and sets the destination to the ULA-MNP of the target Client. If the next hop is reached via an ENET interface, the OAL source instead inserts an IP header of the appropriate protocol version for the underlay ENET (i.e., even if the encapsulation header is IPv4) and sets the destination to the ENET IP address of the next hop. The OAL source inserts the encapsulation header even if no actual fragmentation is needed and/or even if the Jumbo Payload option is present.

The OAL source next assigns an Identification number that is monotonically-incremented for each consecutive sub-parcel, calculates and appends the OAL checksum, then performs IPv6 fragmentation over the sub-parcel if necessary to create fragments small enough to traverse the path to the next hop. (If the encapsulation header is IPv4, the OAL source first translates the encapsulation header into an IPv6 header with IPv4-Compatible IPv6 addresses before performing the fragmentation/reassembly operation while inserting an IPv6 Fragment Header.) The OAL source then writes the "Parcel ID" and sets/clears the "(P)arcel" and "(More) (S)ub-Parcels" bits in the Fragment Header of the first fragment (see: Figure 5). (The OAL source sets P to 1 for a parcel or to 0 for a non-parcel. When P is 1, the OAL next sets S to 1 for non-final sub-parcels or to 0 if the sub-parcel contains the final segment.) The OAL source then forwards each IP encapsulated packet/fragment to the next hop (i.e., after first translating the IPv6 encapsulation header back to IPv4 if necessary).

When the next hop receives the encapsulated IP fragments or whole packets, it acts as an OAL destination and reassembles if necessary (i.e., after first translating the IPv4 encapsulation header to IPv6 if necessary). If the P flag in the first fragment is 0, the OAL destination then processes the reassembled entity as an ordinary IP

packet; otherwise it continues processing as a sub-parcel. If the OAL destination is not the final destination, it retains the sub-parcels along with their Parcel ID and Identification values for a brief time in hopes of re-combining with peer sub-parcels of the same original parcel identified by the 4-tuple consisting of the IP encapsulation source and destination, Identification and Parcel ID. The OAL destination re-combines peers by concatenating the segments included in sub-parcels with the same Parcel ID and with Identification values within 64 of one another to create a larger sub-parcel possibly even as large as the entire original parcel. Order of concatenation is not important, with the exception that the final sub-parcel (i.e., the one with S set to 0) must occur as the final concatenation before transmission. The OAL destination then appends a common IP header plus extensions to each re-combined sub-parcel while resetting M and N in each according to the above equations with J, K and L set accordingly.

When the current OAL destination is an intermediate node, it next becomes an OAL source to forward the re-combined (sub-)parcel(s) to the next hop toward the final destination using encapsulation/translation the same as specified above. (Each such intermediate node MUST ensure that the S flag remains set to 0 in the sub-parcel that contains the final segment.) When the parcel or sub-parcels arrive at the final OAL destination, it re-combines them into the largest possible (sub)-parcels while honoring the S flag then delivers them to upper layers which act on the enclosed 5-tuple information supplied by the original source.

Note: while the final destination may be tempted to re-combine the sub-parcels of multiple different parcels with identical upper layer protocol 5-tuples and with non-final segments of identical length, this process could become complicated when the different parcels each have final segments of diverse lengths. Since this could possibly defeat any perceived performance advantages, the decision of whether and how to perform inter-parcel concatenation is an implementation matter.

7. Frame Format

When the OMNI interface forwards original IP packets from the network layer it first invokes the OAL to create OAL packets/fragments if necessary, then includes any L2 encapsulations and finally engages the native frame format of the underlay interface. For example, for Ethernet-compatible interfaces the frame format is specified in [RFC2464], for aeronautical radio interfaces the frame format is specified in standards such as ICAO Doc 9776 (VDL Mode 2 Technical Manual), for various forms of tunnels the frame format is found in the appropriate tunneling specification, etc.

See Figure 2 for a map of the various L2 layering combinations possible. For any layering combination, the final layer (e.g., UDP, IP, Ethernet, etc.) must have an assigned number and frame format representation that is compatible with the selected underlay interface.

8. Link-Local Addresses (LLAs)

[RFC4861] requires that nodes assign Link-Local Addresses (LLAs) to all interfaces, and that routers use their LLAs as the source address for RA and Redirect messages. OMNI interfaces honor the first requirement, but do not honor the second since the OMNI link could consist of the concatenation of multiple links with diverse ULA prefixes (see Section 9) but for which multiple nodes might configure identical interface identifiers (IIDs). OMNI interface LLAs are therefore considered only as context for IID formation as discussed below and have no other operational role.

OMNI interfaces assign IPv6 LLAs through pre-service administrative actions. Clients assign "LLA-MNPs" with IIDs that embed the Client's unique MNP, while Proxy/Servers assign "LLA-RNDs" that include a randomly-generated IIDs generated as specified in [RFC7217]. LLAs are configured as follows:

- * IPv6 LLA-MNPs encode the most-significant 64 bits of an MNP within the least-significant 64 bits of the IPv6 link-local prefix `fe80::/64`, i.e., in the IID portion of the LLA. The LLA prefix length is determined by adding 64 to the MNP prefix length. e.g., for the MNP `2001:db8:1000:2000::/56` the corresponding LLA-MNP prefix is `fe80::2001:db8:1000:2000/120`. (The base LLA-MNP for each "/N" prefix sets the final 128-N bits to 0, but all LLA-MNPs that match the prefix are also accepted.) Non-MNP IPv6 prefix-based LLAs are also represented the same as for LLA-MNPs, but include a GUA prefix that is not properly covered by the MSP.
- * IPv4-Compatible LLA-MNPs are constructed as `fe80::{IPv4-Prefix}`, i.e., the IID consists of 32 '0' bits followed by a 32 bit IPv4 address/prefix, which may be either public or private in correspondence with the network layer addressing plan. The IPv4-Compatible LLA-MNP prefix length is determined by adding 96 to the IPv4 prefix length. For example, the IPv4-Compatible LLA-MNP for `192.0.2.0/24` is `fe80::192.0.2.0/120`, also written as `fe80::c000:0200/120`. (The base LLA-MNP for each "/N" prefix sets the final 128-N bits to 0, but all LLA-MNPs that match the prefix are also accepted.) Non-MNP IPv4 prefix-based LLAs are also represented the same as for LLA-MNPs, but include a GUA prefix that is not properly covered by the MSP.

- * LLA-RNDs are randomly-generated and assigned to Proxy/Servers and other SRT infrastructure elements. They may also be assigned by Clients to support the MNP delegation process. The upper 72 bits of the LLA-RND encode the prefix fe80::/72, and the lower 56 bits include a randomly-generated candidate pseudo-random value configured as specified in [RFC7217]; if the most significant 24 bits of the 56 bit candidate encodes the value '0', the node generates a new candidate to obtain one with a different most significant 24 bits to avoid overlap with IPv4-Compatible LLAs.
- * The address fe80::/128 (i.e., the LLA /64 prefix followed by an all-zero IID) is considered the LLA Subnet Router Anycast address

Since the prefix 0000::/8 is "Reserved by the IETF" [RFC4291], no MNPs can be allocated from that block ensuring that there is no possibility for overlap between the different MNP and RND LLA constructs discussed above.

Since LLA-MNPs are based on the distribution of administratively assured unique MNPs, and since LLA-RNDs are assumed unique through pseudo-random assignment, OMNI interfaces set the autoconfiguration variable DupAddrDetectTransmits to 0 [RFC4862].

Note: If future protocol extensions relax the 64-bit boundary in IPv6 addressing, the additional prefix bits of an MNP could be encoded in bits 16 through 63 of the LLA-MNP. (The most-significant 64 bits would therefore still be in bits 64-127, and the remaining bits would appear in bits 16 through 48.) However, this would interfere with the relationship between OMNI LLAs and ULAs (see: Section 9) and render many OMNI functions inoperable. The analysis provided in [RFC7421] furthermore suggests that the 64-bit boundary will remain in the IPv6 architecture for the foreseeable future.

9. Unique-Local Addresses (ULAs)

OMNI links use IPv6 Unique-Local Addresses (ULAs) as the source and destination addresses in both IPv6 ND messages and OAL packet IPv6 encapsulation headers. ULAs are routable only within the scope of an OMNI link, and are derived from the IPv6 Unique Local Address prefix fd00::/8 (i.e., the prefix fc00::/7 followed by the L bit set to 1). When the first 16 bits of the ULA encode the value fd00::/16, the address is considered as either a Temporary ULA (TLA) or an eXtended ULA (XLA) - see below. For all other ULAs, the 56 bits following fd00::/8 encode a 40-bit Global ID followed by a 16-bit Subnet ID as specified in Section 3 of [RFC4193]. All OMNI link ULA types finally include a 64-bit value in the IID portion of the address ULA::/64 as specified below.

When a node configures a ULA for OMNI, it selects a 40-bit Global ID for the OMNI link initialized to a candidate pseudo-random value as specified in Section 3 of [RFC4193]; if the most significant 8 bits of the candidate encodes the value '0', the node selects a new candidate until it obtains one with a different most significant 8 bits. All nodes on the same OMNI link use the same Global ID, and statistical uniqueness of the pseudo-random Global ID provides a unique OMNI link identifier allowing different links to be joined together in the future without requiring renumbering.

Next, for each logical segment of the same OMNI link the node selects a 16-bit Subnet ID value between 0x0000 and 0xffff. Nodes on the same logical segment configure the same Subnet ID, but nodes on different segments of the same OMNI link can still exchange IPv6 ND messages as single-hop neighbors even if they configure different Subnet IDs. When a node moves to a different OMNI link segment, it resets the Global ID and Subnet ID value according to the new segment but need not change the IID.

ULAs and their associated prefix lengths are configured in correspondence with LLAs through stateless prefix translation where "ULA-MNPs" simply copy the IIDs of their corresponding LLA-MNPs and "ULA-RNDs" simply copy the IIDs of their corresponding LLA-RNDs. For example, for the OMNI link ULA prefix `fd{Global}:{Subnet}::/64`:

- * the ULA-MNP corresponding to the LLA-MNP `fe80::2001:db8:1:2` with a 56-bit MNP length is simply `fd{Global}:{Subnet}:2001:db8:1:2/120` (where, the ULA prefix length becomes 64 plus the IPv6 MNP length).
- * the ULA-MNP corresponding to `fe80::192.0.2.0` with a 28-bit MNP length is simply `fd{Global}:{Subnet}::192.0.2.0/124` (where, the ULA prefix length becomes 96 plus the IPv4 MNP length).
- * the ULA-RND corresponding to `fe80::0012:3456:789a:bcde` is simply `fd{Global}:{Subnet}::0012:3456:789a:bcde/128`.
- * the Subnet Router Anycast ULA corresponding to `fe80::/128` is simply `fd{Global}:{Subnet}::/128`.

The ULA presents an IPv6 address format that is routable within the OMNI link routing system and can be used to convey link-scoped (i.e., single-hop) IPv6 ND messages across multiple hops through IPv6 encapsulation [RFC2473]. The OMNI link extends across one or more underlying Internetworks to include all Proxy/Servers and other service nodes. All Clients are also considered to be connected to the OMNI link, however unnecessary encapsulations are omitted whenever possible to conserve bandwidth (see: Section 14).

Clients can configure TLAs when they have no other ULA addresses by setting the ULA prefix to fd00::/16 followed by a 48-bit randomly-generated number followed by a random or MNP-based IID as discussed in Section 8. XLAs are a special-case TLA that use the prefix fd00::/64. (Note that XLAs can also be formed from LLAs simply by inverting bits 7 and 8 of 'fe80' to form 'fd00'.)

OMNI nodes use XLA-MNPs as "default" ULAs for representing MNPs in the OMNI link routing system. Clients use {TLA,XLA}-MNPs when they already know their MNP but need to express it outside the context of a specific ULA prefix, and Proxy/Servers advertise XLA-MNPs into the OMNI link routing system instead of advertising fully-qualified {TLA,ULA}-MNPs and/or non-routable LLA-MNPs.

{TLAs,XLAs} provide initial "bootstrapping" addresses while the Client is in the process of procuring an MNP and/or identifying the ULA prefix for the OMNI link segment; TLAs are not advertised into the OMNI link routing system but can be used for Client-to-Client communications within a single {A,I,E}NET when no OMNI link infrastructure is present. Within each individual {A,I,E}NET, TLAs employ optimistic DAD principles [RFC4429] since they are statistically unique.

Each OMNI link may be subdivided into SRT segments that often correspond to different administrative domains or physical partitions. Each SRT segment is identified by a different Subnet ID within the same ULA ::/48 prefix. Multiple distinct OMNI links with different ULA ::/48 prefixes can also be joined together into a single unified OMNI link through simple interconnection without requiring renumbering. In that case, the (larger) unified OMNI link routing system may carry multiple distinct ULA prefixes.

OMNI nodes can use Segment Routing [RFC8402] to support efficient forwarding to destinations located in other OMNI link segments. A full discussion of Segment Routing over the OMNI link appears in [I-D.templin-6man-aero].

Note: IPv6 ULAs taken from the prefix fc00::/7 followed by the L bit set to 0 (i.e., as fc00::/8) are never used for OMNI OAL addressing, however the range could be used for MSP/MNP addressing under certain limiting conditions (see: Section 10). When used within the context of OMNI, ULAs based on the prefix fc00::/8 are referred to as "ULA-C's".

Note: When they appear in the OMNI link routing table, ULA-RNDs always use prefix lengths between /48 and /64 (or, /128) while XLA-MNPs always use prefix lengths between /65 and /128. {TLA,ULA}-MNPs and {TLA,XLA}-RNDs should never appear in the OMNI link routing table, but may appear in {A,I,E}NET routing tables.

10. Global Unicast Addresses (GUAs)

OMNI domains use IP Global Unicast Address (GUA) prefixes [RFC4291] as Mobility Service Prefixes (MSPs) from which Mobile Network Prefixes (MNP) are delegated to Clients. Fixed correspondent node networks reachable from the OMNI link are represented by non-MNP GUA prefixes that are not derived from the MSP, but are treated in all other ways the same as for MNPs.

For IPv6, GUA MSPs are assigned by IANA [IPV6-GUA] and/or an associated Regional Internet Registry (RIR) such that the OMNI link can be interconnected to the global IPv6 Internet without causing inconsistencies in the routing system. An OMNI link could instead use ULAs with the 'L' bit set to 0 (i.e., from the prefix fc00::/8) [RFC4193], however this would require IPv6 NAT if the domain were ever connected to the global IPv6 Internet.

For IPv4, GUA MSPs are assigned by IANA [IPV4-GUA] and/or an associated RIR such that the OMNI link can be interconnected to the global IPv4 Internet without causing routing inconsistencies. An OMNI ANET/ENET could instead use private IPv4 prefixes (e.g., 10.0.0.0/8, etc.) [RFC3330], however this would require IPv4 NAT at the INET-to-ANET/ENET boundary. OMNI interfaces advertise IPv4 MSPs into IPv6 routing systems as IPv4-Compatible IPv6 prefixes [RFC4291] (e.g., the IPv6 prefix for the IPv4 MSP 192.0.2.0/24 is ::192.0.2.0/120).

OMNI interfaces assign the IPv4 anycast address TBD3 (see: IANA Considerations), and IPv4 routers that configure OMNI interfaces advertise the prefix TBD3/N into the routing system of other networks (see: IANA Considerations). OMNI interfaces also configure global IPv6 anycast addresses formed according to [RFC3056] as:

2002:TBD3{32}:MSP{64}:Link-ID{16}

where TBD3{32} is the 32 bit IPv4 anycast address and MSP{64} encodes an MSP zero-padded to 64 bits (if necessary). For example, the OMNI IPv6 anycast address for MSP 2001:db8::/32 is 2002:TBD3{32}:2001:db8:0:0:{Link-ID}, the OMNI IPv6 anycast address for MSP 192.0.2.0/24 is 2002:TBD3{32}::c000:0200:{Link-ID}, etc.).

The 16-bit Link-ID in the OMNI IPv6 anycast address identifies a specific OMNI link within the domain that services the MSP. The special Link-ID value '0' is a wildcard that matches all links, while all other values identify specific links. Mappings between Link-ID values and the ULA Global IDs assigned to OMNI links are outside the scope of this document.

OMNI interfaces assign OMNI IPv6 anycast addresses, and IPv6 routers that configure OMNI interfaces advertise the corresponding prefixes into the routing systems of other networks. An OMNI IPv6 anycast prefix is formed the same as for any IPv6 prefix; for example, the prefix 2002:TBD3{32}:2001:db8::/80 matches all OMNI IPv6 anycast addresses covered by the prefix. When IPv6 routers advertise OMNI IPv6 anycast prefixes in this way, Clients can locate and associate with either a specific OMNI link or any OMNI link within the domain that services the MSP of interest.

OMNI interfaces use OMNI IPv6 and IPv4 anycast addresses to support Service Discovery in the spirit of [RFC7094], i.e., the addresses are not intended for use in long-term transport protocol sessions. Specific applications for OMNI IPv6 and IPv4 anycast addresses are discussed throughout the document as well as in [I-D.templin-6man-aero].

11. Node Identification

OMNI Clients and Proxy/Servers that connect over open Internet networks include a unique node identification value for themselves in the OMNI options of their IPv6 ND messages (see: Section 12.2.12). An example identification value alternative is the Host Identity Tag (HIT) as specified in [RFC7401], while Hierarchical HITs (HHITs) [I-D.ietf-drip-rid] may be more appropriate for certain domains such as the Unmanned (Air) Traffic Management (UTM) service for Unmanned Air Systems (UAS). Another example is the Universally Unique Identifier (UUID) [RFC4122] which can be self-generated by a node without supporting infrastructure with very low probability of collision.

When a Client is truly outside the context of any infrastructure, it may have no MNP information at all. In that case, the Client can use a TLA or (H)HIT as an IPv6 source/destination address for sustained communications in Vehicle-to-Vehicle (V2V) and (multihop) Vehicle-to-Infrastructure (V2I) scenarios. The Client can also propagate the ULA/(H)HIT into the multihop routing tables of (collective) Mobile/Vehicular Ad-hoc Networks (MANETs/VANETs) using only the vehicles themselves as communications relays.

When a Client connects via a protected-spectrum ANET, an alternate form of node identification (e.g., MAC address, serial number, airframe identification value, VIN, etc.) embedded in a ULA may be sufficient. The Client can then include OMNI "Node Identification" sub-options (see: Section 12.2.12) in IPv6 ND messages should the need to transmit identification information over the network arise.

12. Address Mapping - Unicast

OMNI interfaces maintain a neighbor cache for tracking per-neighbor state and use the link-local address format specified in Section 8. IPv6 Neighbor Discovery (ND) [RFC4861] messages sent over OMNI interfaces without encapsulation observe the native underlay interface Source/Target Link-Layer Address Option (S/TLLAO) format (e.g., for Ethernet the S/TLLAO is specified in [RFC2464]). IPv6 ND messages sent over OMNI interfaces using encapsulation do not include S/TLLAOs, but instead include a new option type that encodes encapsulation addresses, interface attributes and other OMNI link information. Hence, this document does not define an S/TLLAO format but instead defines a new option type termed the "OMNI option" designed for these purposes. (Note that OMNI interface IPv6 ND messages sent without encapsulation may include both OMNI options and S/TLLAOs, but the information conveyed in each is mutually exclusive.)

OMNI interfaces prepare IPv6 ND messages that include one or more OMNI options (and any other IPv6 ND options) then completely populate all option information. If the OMNI interface includes an authentication signature, it sets the IPv6 ND message Checksum field to 0 and calculates the authentication signature over the length of the entire OAL packet or super-packet (beginning with a pseudo-header of the IPv6 ND message IPv6 header) but does not calculate/include the IPv6 ND message checksum itself. Otherwise, the OMNI interface calculates the standard IPv6 ND message checksum over the entire OAL packet or super-packet and writes the value in the Checksum field. OMNI interfaces verify authentication and/or integrity of each IPv6 ND message received according to the specific check(s) included, and process the message further only following verification.

OMNI interface Clients such as aircraft typically have multiple wireless data link types (e.g. satellite-based, cellular, terrestrial, air-to-air directional, etc.) with diverse performance, cost and availability properties. The OMNI interface would therefore appear to have multiple L2 connections, and may include information for multiple underlay interfaces in a single IPv6 ND message exchange. OMNI interfaces manage their dynamically-changing multilink profiles by including OMNI options in IPv6 ND messages as discussed in the following subsections.

12.1. The OMNI Option

OMNI options appear in IPv6 ND messages formatted as shown in Figure 12:

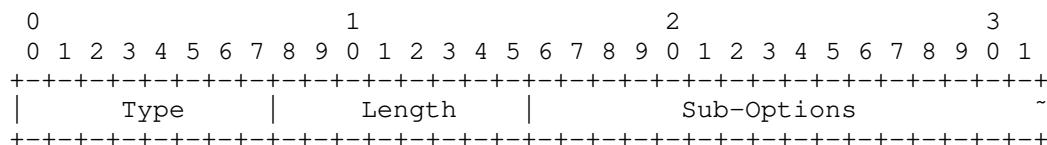


Figure 12: OMNI Option Format

In this format:

- * Type is set to TBD4 (see: IANA Considerations).
- * Length is set to the number of 8 octet blocks in the option. The value 0 is invalid, while the values 1 through 255 (i.e., 8 through 2040 octets, respectively) indicate the total length of the OMNI option. If multiple OMNI option instances appear in the same IPv6 ND message, the union of the contents of all OMNI options is accepted unless otherwise qualified for specific sub-options below.
- * Sub-Options is a Variable-length field padded if necessary such that the complete OMNI Option is an integer multiple of 8 octets long. Sub-Options contains zero or more sub-options as specified in Section 12.2.

The OMNI option is included in all OMNI interface IPv6 ND messages; the option is processed by receiving interfaces that recognize it and otherwise ignored. The OMNI interface processes all OMNI option instances received in the same IPv6 ND message in the consecutive order in which they appear. The OMNI option(s) included in each IPv6 ND message may include full or partial information for the neighbor. The OMNI interface therefore retains the union of the information in the most recently received OMNI options in the corresponding NCE.

12.2. OMNI Sub-Options

Each OMNI option includes a Sub-Options block containing zero or more individual sub-options. Each consecutive sub-option is concatenated immediately following its predecessor. All sub-options except Pad1 (see below) are in an OMNI-specific type-length-value (TLV) format encoded as follows:

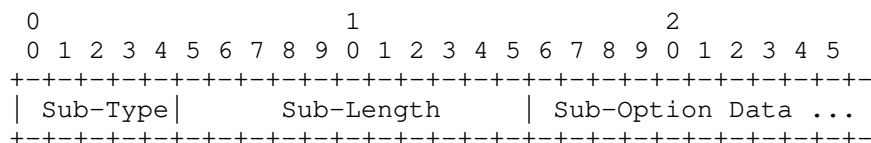


Figure 13: Sub-Option Format

- * Sub-Type is a 5-bit field that encodes the sub-option type. Sub-option types defined in this document are:

Sub-Option Name	Sub-Type
Pad1	0
PadN	1
Neighbor Coordination	2
Interface Attributes	3
Multilink Forwarding Params	4
Traffic Selector	5
Geo Coordinates	6
DHCPv6 Message	7
HIP Message	8
PIM-SM Message	9
Fragmentation Report	10
Node Identification	11
ICMPv6 Error	12
QUIC-TLS Message	13
Proxy/Server Departure	14
Sub-Type Extension	30

Figure 14

Sub-Types 15-29 are available for future assignment for major protocol functions, while Sub-Type 30 supports scalable extension to include other functions. Sub-Type 31 is reserved by IANA.

- * Sub-Length is an 11-bit field that encodes the length of the Sub-Option Data in octets.
- * Sub-Option Data is a block of data with format determined by Sub-Type and length determined by Sub-Length. Note that each individual sub-option may end on an arbitrary octet boundary, whereas the OMNI option itself must include padding if necessary for 8-octet alignment.

The OMNI interface codes each sub-option with a 2 octet header that includes Sub-Type in the most significant 5 bits followed by Sub-Length in the next most significant 11 bits. Each sub-option encodes a maximum Sub-Length value of 2038 octets minus the lengths of the

OMNI option header and any preceding sub-options. This allows ample Sub-Option Data space for coding large objects (e.g., ASCII strings, domain names, protocol messages, security codes, etc.), while a single OMNI option is limited to 2040 octets the same as for any IPv6 ND option.

The OMNI interface codes initial sub-options in a first OMNI option instance and subsequent sub-options in additional instances in the same IPv6 ND message in the intended order of processing. The OMNI interface can then code any remaining sub-options in additional IPv6 ND messages if necessary. Implementations must observe these size limits and refrain from sending IPv6 ND messages larger than the OMNI interface MTU.

The OMNI interface processes all OMNI option Sub-Options received in an IPv6 ND message while skipping over and ignoring any unrecognized sub-options. The OMNI interface processes the Sub-Options of all OMNI option instances in the consecutive order in which they appear in the IPv6 ND message, beginning with the first instance and continuing through any additional instances to the end of the message. If an individual sub-option length would cause processing to exceed the OMNI option instance and/or IPv6 ND message lengths, the OMNI interface accepts any sub-options already processed and ignores the remainder of that instance. The interface then processes any remaining OMNI option instances in the same fashion to the end of the IPv6 ND message.

When an OMNI interface includes an authentication sub-option (e.g., see: Section 12.2.9), it MUST appear as the first sub-option of the first OMNI option which must appear immediately following the IPv6 ND message header (all other authentication sub-options are ignored). If the IPv6 ND message is the first packet in a combined OAL super-packet, the OMNI interface calculates the authentication signature over the entire length of the super-packet, i.e., and not just to the end of the IPv6 ND message itself. When the first sub-option is not authentication, the OMNI interface instead calculates the IPv6 ND message checksum over the entire length of the packet/super-packet.

When a Client OMNI interface prepares a secured unicast RS message, it includes an Interface Attributes sub-option specific to the underlay interface that will transmit the RS (see: Section 12.2.4) immediately following the authentication and header extension sub-options if present; otherwise as the first sub-option of the first OMNI option which must appear immediately following the IPv6 ND message header. When a Client OMNI interface prepares a secured unicast NS message, it instead includes a Multilink Forwarding Parameters sub-option specific to the underlay interface that will transmit the NS (see: Section 12.2.5).

Note: large objects that exceed the maximum Sub-Option Data length are not supported under the current specification; if this proves to be limiting in practice, future specifications may define support for fragmenting large sub-options across multiple OMNI options within the same IPv6 ND message (or even across multiple IPv6 ND messages, if necessary).

The following sub-option types and formats are defined in this document:

12.2.1. Pad1

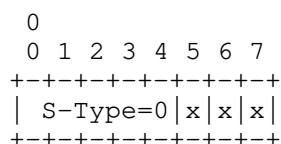


Figure 15: Pad1

- * Sub-Type is set to 0. If multiple instances appear in OMNI options of the same message all are processed.
- * Sub-Type is followed by 3 'x' bits, set to any value on transmission (typically all-zeros) and ignored on reception. Pad1 therefore consists of 1 octet with the most significant 5 bits set to 0, and with no Sub-Length or Sub-Option Data fields following.

If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

12.2.2. PadN

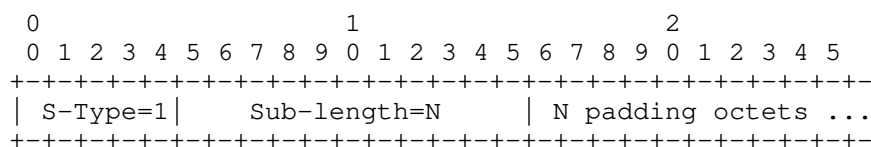


Figure 16: PadN

- * Sub-Type is set to 1. If multiple instances appear in OMNI options of the same message all are processed.
- * Sub-Length is set to N that encodes the number of padding octets that follow.

- * Sub-Option Data consists of N octets, set to any value on transmission (typically all-zeros) and ignored on receipt.

When a proxy forwards an IPv6 ND message with OMNI options, it can employ PadN to cancel any sub-options (other than Pad1) that should not be processed by the next hop by simply writing the value '1' over the Sub-Type. When the proxy alters the IPv6 ND message contents in this way, any included authentication and integrity checks are invalidated. See: Appendix B for a discussion of IPv6 ND message authentication and integrity.

12.2.3. Neighbor Coordination

IPv6 ND messages used for Prefix Length assertion, service coordination and/or Window Synchronization include a Neighbor Coordination sub-option. If a Neighbor Coordination sub-option is included, it must appear immediately after the authentication sub-option if present; otherwise, as the first (non-padding) sub-option of the first OMNI option. If multiple Neighbor Coordination sub-options are included (whether in a single OMNI option or multiple), only the first is processed and all others are ignored.

The Neighbor Coordination sub-option is formatted as follows:

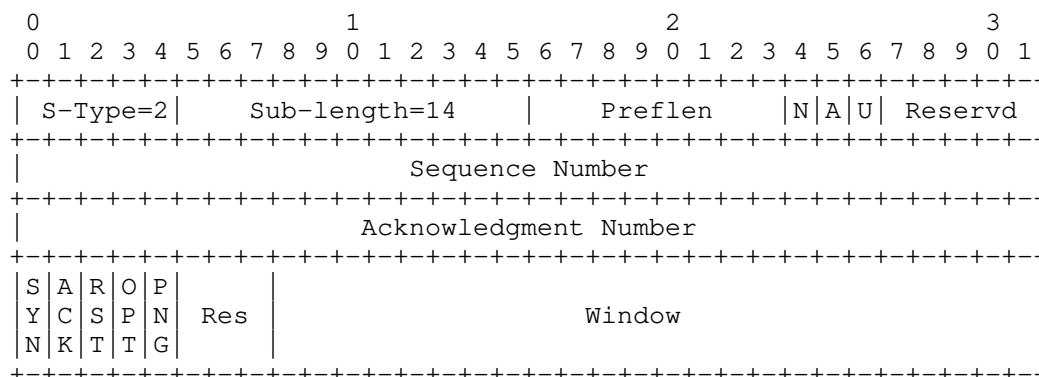


Figure 17: Neighbor Coordination

- * Sub-Type is set to 2.
- * Sub-Length is set to 14.
- * The first two octets of Sub-Option Data contains a 1-octet Prefix Length followed by a 1-octet flags field interpreted as follows:

- Preflen is an 8 bit field that determines the length of prefix associated with a ULA. Values 0 through 128 specify a valid prefix length (if any other value appears the OMNI option must be ignored). For IPv6 ND messages sent from a Client to the MS, Preflen applies to the IPv6 source ULA and provides the length that the Client is requesting from or asserting to the MS. For IPv6 ND messages sent from the MS to the Client, Preflen applies to the IPv6 destination ULA and indicates the length that the MS is granting to the Client. For IPv6 ND messages sent between MS endpoints, Preflen provides the length associated with the source/target Client MNP that is subject of the ND message. When an IPv6 ND RS/RA message sets Preflen to 0, the recipient regards the message as a prefix release indication.
- The N/A/U flags are set or cleared in Client RS messages as directives to FHS and Hub Proxy/Servers and ignored in all other IPv6 ND messages. When an FHS Proxy/Server forwards or processes an RS with the N flag set, it responds directly to NS Neighbor Unreachability Detection (NUD) messages by returning NA(NUD) replies; otherwise, it forwards NS(NUD) messages to the Client. When the Hub Proxy/Server receives an RS with the A flag set, it responds directly to NS Address Resolution (AR) messages by returning NA(AR) replies; otherwise, it forwards NS(AR) messages to the Client. When the Hub Proxy/Server receives an RS with the U flag set, it maintains a Report List of recent NS(AR) message sources for this Client and sends uNA messages to all list members if any aspects of the Client's underlay interfaces change. Proxy/Servers function according to the N/A/U flag settings received in the most recent RS message to support dynamic Client updates. In all IPv6 ND messages, the remaining 5 flag bits are set to 0 on transmission and ignored on reception.
- * The remainder of Sub-Option Data contains a 4-octet Sequence Number, followed by a 4-octet Acknowledgement Number, followed by a 1-octet flags field followed by a 3-octet Window size modeled from the Transmission Control Protocol (TCP) header specified in Section 3.1 of [RFC0793]. The (SYN, ACK, RST) flags are used for TCP-like window synchronization, while the TCP (URG, PSH, FIN) flags are not used and therefore omitted. The (OPT, PNG) flags are OMNI-specific, and the remaining flags are Reserved. Together, these fields support the asymmetric and symmetric OAL window synchronization services specified in Section 6.6.

12.2.4. Interface Attributes

The Interface Attributes sub-option provides neighbors with forwarding information for the multilink conceptual sending algorithm discussed in Section 14. Neighbors use the forwarding information to selecting among potentially multiple candidate underlay interfaces that can be used to forward carrier packets to the neighbor based on factors such as traffic selectors and link quality. Interface Attributes further include link-layer address information to be used for either direct INET encapsulation for targets in the local SRT segment or spanning tree forwarding for targets in remote SRT segments.

OMNI nodes include Interface Attributes for some/all of a target Client's underlay interfaces in NS/NA and uNA messages used to publish Client information (see: [I-D.templin-6man-aero]). At most one Interface Attributes sub-option for each distinct omIndex may be included; if an NS/NA message includes multiple Interface Attributes sub-options for the same omIndex, the first is processed and all others are ignored. OMNI nodes that receive NS/NA messages can use all of the included Interface Attributes and/or Traffic Selectors to formulate a map of the prospective target node as well as to seed the information to be populated in a Multilink Forwarding Parameters sub-option (see: Section 12.2.5).

OMNI Clients and Proxy/Servers also include Interface Attributes sub-options in RS/RA messages used to initialize, discover and populate routing and addressing information. Each RS message MUST contain exactly one Interface Attributes sub-option with an omIndex corresponding to the Client's underlay interface used to transmit the message, and each RA message MUST echo the same Interface Attributes sub-option with any (proxied) information populated by the FHS Proxy/Server to provide operational context.

OMNI Client RS and Proxy/Server RA messages MUST include the Interface Attributes sub-option for the Client underlay interface in the first OMNI option immediately following the Neighbor Coordination and/or authentication sub-option(s) if present; otherwise, immediately following the OMNI header. When an FHS Proxy/Server receives an RS message destined to an anycast L2 address, it MUST include an Interface Attributes sub-option with omIndex '0' that encodes its unicast L2 address relative to the Client's underlay interface immediately after the Interface Attributes sub-option in the solicited RA response. Any additional Interface Attributes sub-options that appear in RS/RA messages are ignored.

The Interface Attributes sub-options are formatted as shown below:

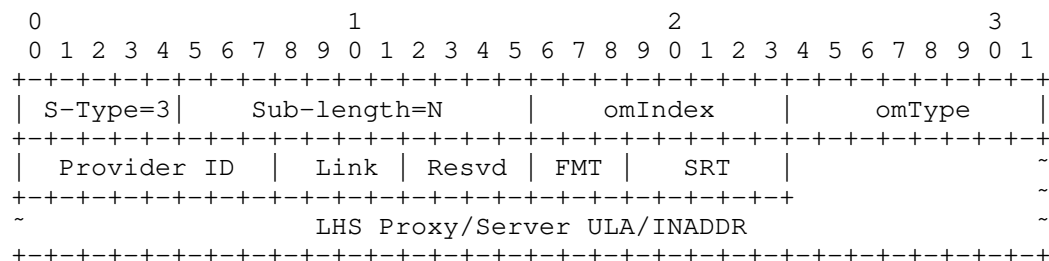


Figure 18: Interface Attributes

- * Sub-Type is set to 3.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.
- * Sub-Option Data contains an "Interface Attributes" option encoded as follows:
 - omIndex is a 1-octet value corresponding to a specific underlay interface. Client OMNI interfaces MUST number each distinct underlay interface with an omIndex value between '1' and '255' that represents a Client-specific 8-bit mapping for the actual ifIndex value assigned by network management [RFC2863], then set omIndex to either a specific omIndex value or '0' to denote "unspecified".
 - omType is set to an 8-bit integer value corresponding to the underlay interface identified by omIndex. The value represents an OMNI interface-specific 8-bit mapping for the actual IANA ifType value registered in the 'IANAifType-MIB' registry [<http://www.iana.org>].
 - Provider ID is set to an OMNI interface-specific 8-bit ID value for the network service provider associated with this omIndex.
 - Link encodes a 4-bit link metric. The value '0' means the link is DOWN, and the remaining values mean the link is UP with metric ranging from '1' ("lowest") to '15' ("highest").
 - Resvd is a 4-bit Reserved field set to 0 on transmission and ignored on reception.
 - FMT - a 3-bit "Forward/Mode/Type" code interpreted as follows:

- o The most significant two bits (i.e., "FMT-Forward" and "FMT-Mode") are interpreted in conjunction with one another. When FMT-Forward is clear, the LHS Proxy/Server performs OAL reassembly and decapsulation to obtain the original IP packet before forwarding. If the FMT-Mode bit is clear, the LHS Proxy/Server then forwards the original IP packet at layer 3; otherwise, it invokes the OAL to re-encapsulate, re-fragment and forwards the resulting carrier packets to the Client via the selected underlay interface. When FMT-Forward is set, the LHS Proxy/Server forwards unsecured OAL fragments to the Client without reassembling, while reassembling secured OAL fragments before re-fragmenting and forwarding to the Client. If FMT-Mode is clear, all carrier packets destined to the Client must always be forwarded through the LHS Proxy/Server; otherwise the Client is eligible for direct forwarding over the open INET where it may be located behind one or more NATs.
- o The least significant bit (i.e., "FMT-Type") determines the length of the LHS Proxy/Server INADDR field. If FMT-Type is clear, INADDR includes a 4-octet IPv4 address; otherwise, a 16-octet IPv6 address. (Note: the INADDR "short form" minimizes overhead for ND messages that include many Interface Attributes sub-options with IPv4 addresses.)
- SRT - a 5-bit Segment Routing Topology prefix length value between 0 and 16 that (when added to 48) determines the prefix length associated with the LHS ULA Subnet ID. For example, the value 5 corresponds to the prefix ULA::/53.
- LHS Proxy/Server ULA/INADDR - the first 15 octets following the "FMT/SRT" octet includes the 120 least significant bits of the ULA of the LHS Proxy/Server on the path from a source neighbor to the target Client's underlay interface. (Note that the FMT/SRT code is replaced with the value "fd" after processing to form a proper Proxy/Server ULA.) When SRT and ULA are both set to 0, the LHS Proxy/Server is considered unspecified in this IPv6 ND message. FMT, SRT and LHS together provide guidance for the OMNI interface forwarding algorithm. Specifically, if SRT/LHS is located in the local OMNI link segment, then the source can reach the target Client either through its dependent Proxy/Server or through direct encapsulation following NAT traversal according to FMT. Otherwise, the target Client is located on a different SRT segment and the path from the source must employ a combination of route optimization and spanning tree hop traversals. INADDR identifies the LHS Proxy/Server's INET-facing interface not located behind NATs, therefore no UDP port number is included since port number 8060 is used when the

L2 encapsulation includes a UDP header. Instead, INADDR includes only a 4-octet IPv4 or 16-octet IPv6 address with type and length determined by FMT-Type. The IP address is recorded in network byte order in ones-compliment "obfuscated" form per [RFC4380].

12.2.5. Multilink Forwarding Parameters

OMNI nodes include the Multilink Forwarding Parameters sub-option in NS/NA messages used to coordinate with multilink route optimization targets. If an NS message includes the sub-option, the solicited NA response must also include the sub-option. The OMNI node MUST include the sub-option in the first OMNI option immediately following the Neighbor Coordination and/or authentication message sub-option(s) if present. Otherwise, the OMNI node MUST include the sub-option immediately following the OMNI header. Each NS/NA message may contain at most one Multilink Forwarding Parameters sub-option; if an NS/NA message contains additional Multilink Forwarding Parameters sub-options, the first is processed and all others are ignored.

When an NS/NA message includes the sub-option, the FHS Client omIndex MUST correspond to the underlay interface used to transmit the message. When the NS/NA message also includes Interface Attributes sub-options any that include the same FHS/LHS Client omIndex are ignored while all others are processed.

The Multilink Forwarding Parameters sub-option includes the necessary state for establishing Multilink Forwarding Vectors (MFVs) in the Multilink Forwarding Information Bases (MFIBs) of the OAL source, destination and intermediate nodes in the path. The sub-option also records addressing information for FHS/LHS nodes on the path, including "INADDRs" which MUST be unicast IP encapsulation addresses (i.e., and not anycast/multicast). The manner for populating multilink forwarding information is specified in detail in [I-D.templin-6man-aero].

The Multilink Forwarding Parameters sub-option is formatted as shown in Figure 19:

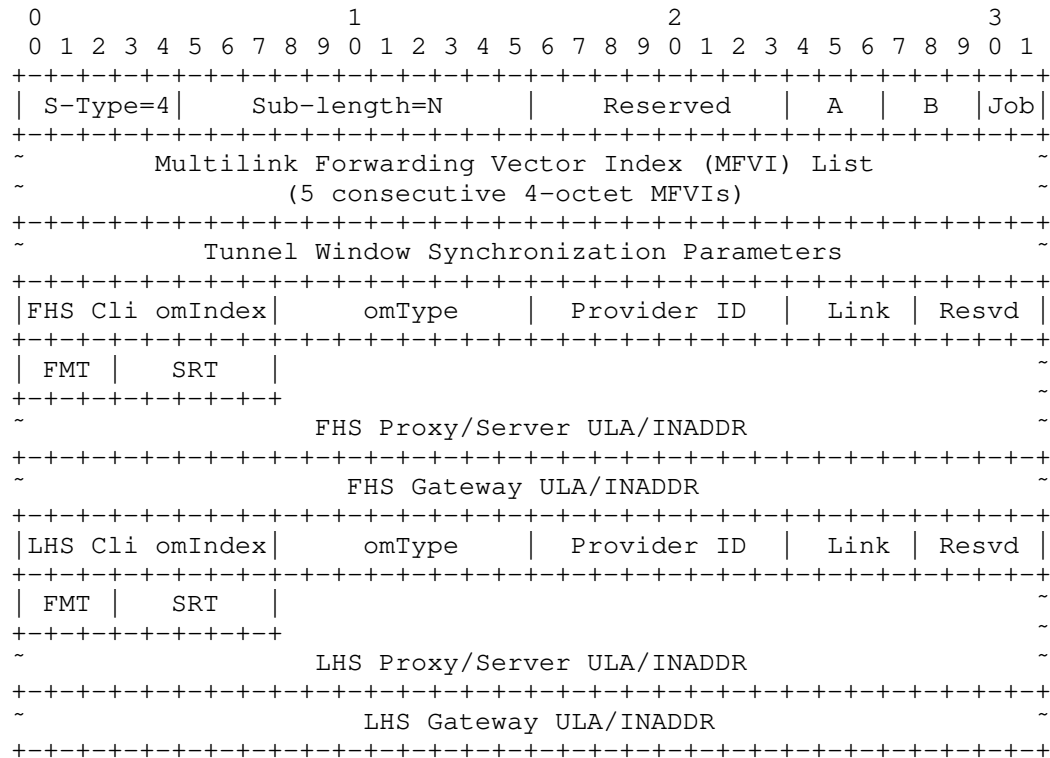


Figure 19: Multilink Forwarding Parameters

- * Sub-Type is set to 4. If multiple instances appear in the same message (i.e., whether in a single OMNI option or multiple) the first instance is processed and all others are ignored.
- * Sub-Length encodes the number of Sub-Option Data octets that follow. The length includes all fields up to and including the Tunnel Window Synchronization Parameters for all Job codes, while including the remaining fields only for Job codes "0" and "1" (see below).
- * Sub-Option Data contains Multilink Forwarding Parameters as follows:
 - Reserved is a 1-octet reserved field set to 0 on transmission and ignored on receipt.

- A/B and Job are fields that determine per-hop processing of the MFVI List, where A is a 3-bit count of the number of "A" MFVI List entries and B is a 3-bit count of the number of "B" MFVI List entries (valid A/B values are 0-5). Job is a 2-bit code interpreted as follows:
 - o '00' - "Initialize; Build B" - the FHS source sets this code in an NS used to initialize MFV state (any other messages that include this code MUST be dropped). The FHS source first sets A/B to 0, and the FHS source and each intermediate node along the path to the LHS destination that processes the message creates a new MFV. Each node that processes the message then assigns a unique 4-octet "B" MFVI to the MVE and also writes the value into list entry B, then increments B. When the message arrives at the LHS destination, B will contain the number of MFVI List "B" entries, with the FHS source entry first, followed by entries for each consecutive intermediate node and ending with an entry for the final intermediate node (i.e., the list is populated in the forward direction).
 - o '01' - "Follow B; Build A" - the LHS source sets this code in a solicited NA response to a solicitation with Job code "0" (any other messages that include this code MUST be dropped). The LHS source first copies the MFVI List and B value from the code "0" solicitation into these fields and sets A to 0. The LHS source and each intermediate node along the path to the FHS destination that processes the message then uses MFVI List entry B to locate the corresponding MFV. Each node that processes the message then assigns a unique 4-octet "A" MFVI to the MVE and also writes the value into list entry B, then increments A and decrements B. When the message arrives at the FHS destination, A will contain the number of MFVI List "A" entries, with the LHS source entry last, preceded by entries for each consecutive intermediate node and beginning with an entry for the final intermediate node (i.e., the list is populated in the reverse direction).
 - o '10' - "Follow A; Record B" - the FHS node that sent the original code "0" solicitation and received the corresponding code "1" advertisement sets this code in any subsequent NS/NA messages sent to the same LHS destination. The FHS source copies the MFVI List and A value from the code "1" advertisement into these fields and sets B to 0. The FHS source and each intermediate node along the path to the LHS destination that processes the message then uses the "A" MFVI found at list entry B to locate the corresponding

MFV. Each node that processes the message then writes the MFV's "B" MFVI into list entry B, then decrements A and increments B. When the message arrives at the LHS destination, B will contain the number of MFVI List "B" entries populated in the forward direction.

- o '11' - "Follow B; Record A" - the LHS node that received the original code "0" solicitation and sent the corresponding code "1" advertisement sets this code in any subsequent NS/NA messages sent to the same FHS destination. The LHS source copies the MFVI List and B values from the code "0" solicitation into these fields and sets A to 0. The LHS source and each intermediate node along the path to the FHS destination that processes the message then uses the "B" MFVI List entry found at list entry B to locate the corresponding MFV. Each node that processes the message then writes the MFV's "A" MFVI into list entry B, then increments A and decrements B. When the message arrives at the FHS destination, A will contain the number of MFVI List "A" entries populated in the reverse direction.

Job and A/B together determine the per-hop behavior at each FHS/LHS source, intermediate node and destination that processes an IPv6 ND message. When a Job code specifies "Initialize", each FHS/LHS node that processes the message creates a new MFV. When a Job code specifies "Build", each node that processes the message assigns a new MFVI. When a Job code specifies "Follow", each node that processes the message uses an A/B MFVI List entry to locate an MFV (if the MFV cannot be located, the node returns a parameter problem and drops the message). Using this algorithm, FHS sources that send code '00' solicitations and receive code '01' advertisements discover only "A" information, while LHS sources that receive code '00' solicitations and return code '01' advertisements discover only "B" information. FHS/LHS intermediate nodes can instead examine A, B and the MFVI List to determine the number of previous hops, the number of remaining hops, and the A/B MFVIs associated with the previous/remaining hops. However, no intermediate nodes will discover inappropriate A/B MFVIs for their location in the multihop forwarding chain. See: [I-D.templin-6man-aero] for further discussion on A/B MFVI processing.

- Multilink Forwarding Vector Index (MFVI) List is a 20-octet block that contains 5 consecutive 4-octet MFVI entries. The FHS/LHS source and each intermediate node on the path to the destination processes the list according to the Job and A/B codes (see above). Note that the reason the MFVI list contains

at most 5 entries is that only the FHS (Client, Proxy/Server, Gateway) and LHS (Client, Proxy/Server, Gateway) nodes are eligible for OMNI link route optimization resulting in at most 5 MFVIs "hops" that must be exposed. All other OMNI link nodes (i.e., downstream Clients that connect via an FHS/LHS Client) must forward through their upstream-dependent OMNI link neighbors without applying OMNI link route optimization.

- Tunnel Window Synchronization Parameters is a 12-octet block that consists of a 4-octet Sequence Number followed by a 4-octet Acknowledgement Number followed by a 1-octet Flags field followed by a 3-octet Window field (i.e., the same as for the OMNI header parameters). Tunnel endpoints use these parameters for simultaneous middlebox window synchronization in a single solicitation/advertisement message exchange.
- For Job codes '00' and '01' only, two trailing state variable blocks are included for First-Hop Segment (FHS) followed by Last-Hop Segment (LHS) network elements. When present, each block encodes the following information:
 - o Client omIndex, omType, Provider ID and Resvd/Link are 1-octet fields (at offset 0 from the beginning of the Sub-Option Data) that include link parameters for the Client underlay interface. These fields are populated based on information discovered in Interface Attributes sub-options included in earlier RS/RA and/or NS/NA exchanges.
 - o FMT/SRT is a 1-octet field with a 5-bit SRT prefix length that applies to all elements in the segment. The FMT-Forward/Mode bits determine the characteristics of the Proxy/Server relationship for this specific Client underlay interface (i.e., the same as described in Section 12.2.4), and the FMT-Type bits determine the IP address version for all INADDR fields relative to this SRT segment. Unlike the case for Interface Attributes, all INADDR fields are always 16 bits in length regardless of the IP protocol version with IPv4 INADDRs encoded as IPv4-Compatible IPv6 addresses [RFC4291]. (Note: the INADDR "long-form" is used exclusively since there may be no a priori knowledge of the IP address version used at each hop.) The IP address is recoded in network byte order, and in ones-complement "obfuscated" form the same as described in Section 12.2.4.
 - o Proxy/Server ULA/INADDR includes a 15 octet value that encodes the 120 least significant bits of the Proxy/Server ULA followed by a 16 octet INADDR. (Note that the FMT/SRT code is replaced with the value "fd" after processing to

form a proper Proxy/Server ULA.) INADDR identifies an open INET interface not located behind NATs, therefore no UDP port number is included since port number 8060 is used when the L2 encapsulation includes a UDP header.

- o Gateway ULA/INADDR encodes a 16 octet ULA followed by a 16 octet INADDR exactly as for the Proxy/Server ULA/INADDR. (Note that the Gateway ULA simply encodes the value "fd" in the most significant bits, since the FMT/SRT code applies to both the Proxy/Server and Gateway.)

12.2.6. Traffic Selector

When used in conjunction with Interface Attributes and/or Multilink Forwarding Parameters information, the Traffic Selector sub-option provides forwarding information for the multilink conceptual sending algorithm discussed in Section 14.

IPv6 ND messages include Traffic Selectors for some or all of the source/target Client's underlay interfaces. Traffic Selectors for some or all of a target Client's underlay interfaces are also included in uNA messages used to publish Client information changes. See: [I-D.templin-6man-aero] for more information.

Traffic Selectors must be honored by all implementations in the format shown below:

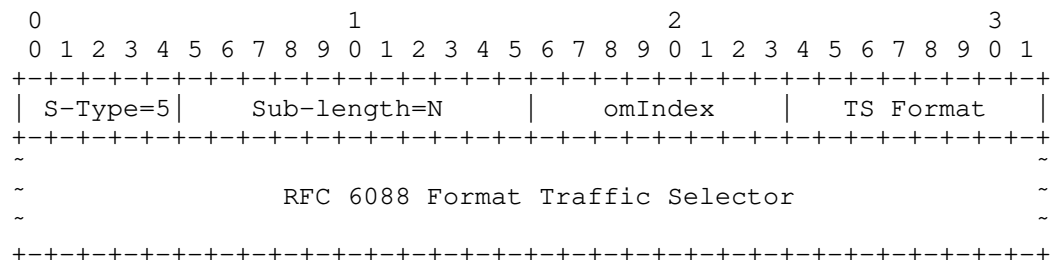


Figure 20: Traffic Selector

- * Sub-Type is set to 5. Each IPv6 ND message may contain zero or more Traffic Selectors for each omIndex; when multiple Traffic Selectors for the same omIndex appear, all are processed and the cumulative information from all is accepted.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.
- * Sub-Option Data contains a "Traffic Selector" encoded as follows:

- omIndex is a 1-octet value corresponding to a specific underlay interface the same as specified above for Interface Attributes and Multilink Forwarding Parameters above. The OMNI options of a single message may include multiple Traffic Selector sub-options; each with the same or different omIndex values.
- TS Format is a 1-octet field that encodes a Traffic Selector version per [RFC6088]. If TS Format encodes the value 1 or 2, the Traffic Selector includes IPv4 or IPv6 information, respectively. If TS Format encodes any other value, the sub-option is ignored.
- The remainder of the sub-option includes a traffic selector formatted per [RFC6088] beginning with the "Flags (A-N)" field, and with the Traffic Selector IP protocol version coded in the TS Format field. If a single interface identified by omIndex requires Traffic Selectors for multiple IP protocol versions, or if a Traffic Selector block would exceed the available space, the remaining information is coded in additional Traffic Selector sub-options that all encode the same omIndex.

12.2.7. Geo Coordinates

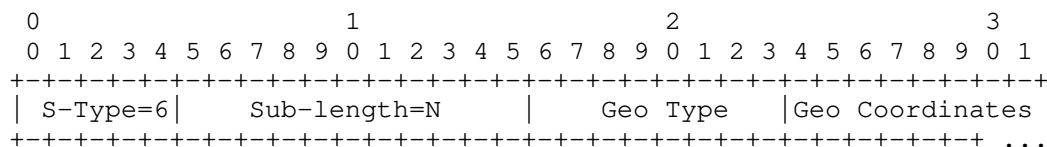


Figure 21: Geo Coordinates Sub-option

- * Sub-Type is set to 6. If multiple instances appear in OMNI options of the same message all are processed.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.
- * Geo Type is a 1 octet field that encodes a type designator that determines the format and contents of the Geo Coordinates field that follows. The following types are currently defined:
 - 0 - NULL, i.e., the Geo Coordinates field is zero-length.
- * A set of Geo Coordinates of length up to the remaining available space for this OMNI option. New formats to be specified in future documents and may include attributes such as latitude/longitude, altitude, heading, speed, etc.

12.2.8. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Message

The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) sub-option may be included in the OMNI options of Client RS messages and Proxy/Server RA messages. FHS Proxy/Servers that forward RS/RA messages between a Client and an LHS Proxy/Server also forward DHCPv6 Sub-Options unchanged. Note that DHCPv6 messages do not include a Checksum field since integrity is protected by the IPv6 ND message checksum, authentication signature and/or lower-layer authentication and integrity checks.

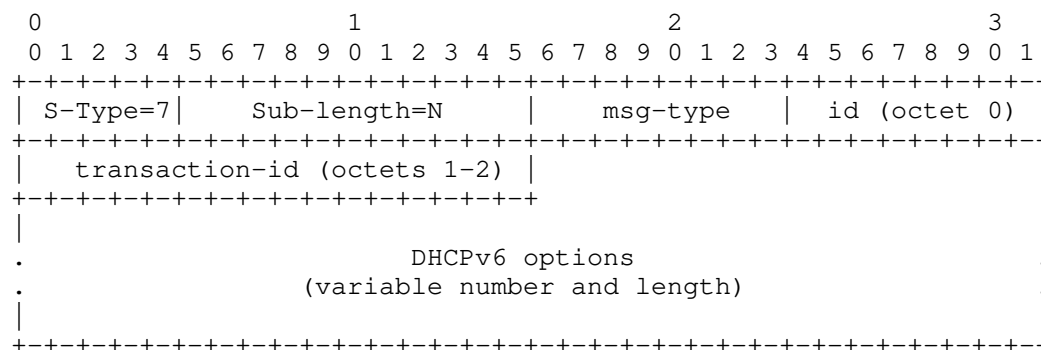


Figure 22: DHCPv6 Message Sub-option

- * Sub-Type is set to 7. If multiple instances appear in OMNI options of the same message the first is processed and all others are ignored.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The 'msg-type' and 'transaction-id' fields are always present; hence, the length of the DHCPv6 options is limited by the remaining available space for this OMNI option.
- * 'msg-type' and 'transaction-id' are coded according to Section 8 of [RFC8415].
- * A set of DHCPv6 options coded according to Section 21 of [RFC8415] follows.

12.2.9. Host Identity Protocol (HIP) Message

The Host Identity Protocol (HIP) Message sub-option (when present) provides authentication for IPv6 ND messages exchanged between Clients and FHS Proxy/Servers over an open Internetwork. FHS Proxy/Servers authenticate the HIP authentication signatures in source Client IPv6 ND messages before securely forwarding them to other OMNI nodes. LHS Proxy/Servers that receive secured IPv6 ND messages from other OMNI nodes that do not already include a security sub-option insert HIP authentication signatures before forwarding them to the target Client.

OMNI interfaces MUST include the HIP message (when present) as the first sub-option of the first OMNI option, which MUST appear immediately following the IPv6 ND message header. OMNI interfaces can therefore easily locate the HIP message and verify the authentication signature without applying deep inspection. OMNI interfaces that receive IPv6 ND messages without a HIP (or other authentication) sub-option as the first OMNI sub-option instead verify the IPv6 ND message checksum.

OMNI interfaces include the HIP message sub-option when they forward IPv6 ND messages that require security over INET underlay interfaces, i.e., where authentication and integrity is not already assured by lower layers. The OMNI interface calculates the authentication signature over the entire length of the OAL packet (or super-packet) beginning with a pseudo-header of the IPv6 ND message header and extending over the remainder of the OAL packet. OMNI interfaces that process OAL packets that contain secured IPv6 ND messages verify the signature then either process the rest of the message locally or forward a proxied copy to the next hop.

When a FHS Client inserts a HIP message sub-option in an NS/NA message destined to a target in a remote spanning tree segment, it must ensure that the insertion does not cause the message to exceed the OMNI interface MTU. When the remote segment LHS Proxy/Server forwards the NS/NA message from the spanning tree to the target Client, it inserts a new HIP message sub-option if necessary while overwriting or cancelling the (now defunct) HIP message sub-option supplied by the FHS Client.

If the defunct HIP sub-option size was smaller than the space needed for the LHS Client HIP message (or, if no defunct HIP sub-option is present), the LHS Proxy/Server adjusts the space immediately following the OMNI header by copying the preceding portion of the IPv6 ND message into buffer headroom free space or copying the remainder of the IPv6 ND message into buffer tailroom free space. The LHS Proxy/Server then insets the new HIP sub-option immediately after the OMNI header and immediately before the next sub-option while properly overwriting the defunct sub-option if present.

If the defunct HIP sub-option size was larger than the space needed for the LHS Client HIP message, the LHS Proxy/Server instead overwrites the existing sub-option and writes a single Pad1 or PadN sub-option over the next 1-2 octets to cancel the remainder of the defunct sub-option. If the LHS Proxy/Server cannot create sufficient space through any means without causing the OMNI option to exceed 2040 octets or causing the IPv6 ND message to exceed the OMNI interface MTU, it returns a suitable error (see: Section 12.2.13) and drops the message.

The HIP message sub-option is formatted as shown below:

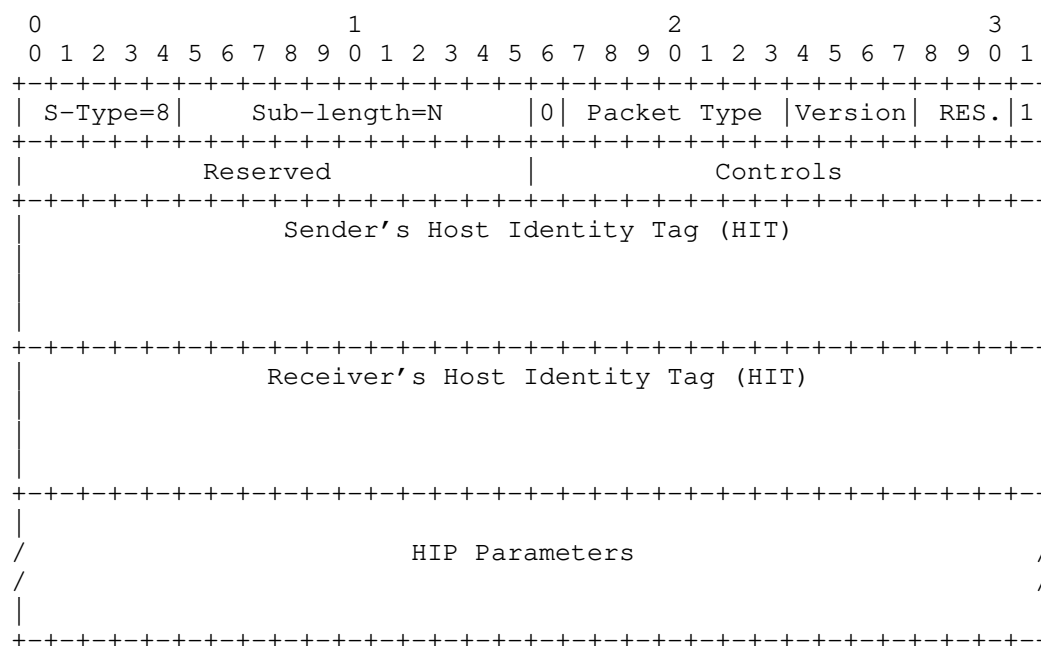


Figure 23: HIP Message Sub-option

- * Sub-Type is set to 8. If multiple instances appear in OMNI options of the same message the first is processed and all others are ignored.
- * Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the HIP parameters. The length of the entire HIP message is therefore limited by the remaining available space for this OMNI option.
- * The HIP message is coded per Section 5 of [RFC7401], except that the OMNI "Sub-Type" and "Sub-Length" fields replace the first 2 octets of the HIP message header (i.e., the Next Header and Header Length fields). Also, since the IPv6 ND message is already protected by the authentication signature and/or lower-layer authentication and integrity checks, the HIP message Checksum field is replaced by a Reserved field set to 0 on transmission and ignored on reception.

Note: In some environments, maintenance of a Host Identity Tag (HIT) namespace may be unnecessary for securely associating an OMNI node with an IPv6 address-based identity. In that case, IPv6 ULAs can be used instead of HITs in the authentication signature as long as the address can be uniquely associated with the Sender/Receiver.

12.2.10. PIM-SM Message

The Protocol Independent Multicast - Sparse Mode (PIM-SM) Message sub-option may be included in the OMNI options of IPv6 ND messages. PIM-SM messages are formatted as specified in Section 4.9 of [RFC7761], with the exception that the Checksum field is replaced by a Reserved field (set to 0) since the IPv6 ND message is already protected by the IPv6 ND message checksum, authentication signature and/or lower-layer authentication and integrity checks. The PIM-SM message sub-option format is shown in Figure 24:

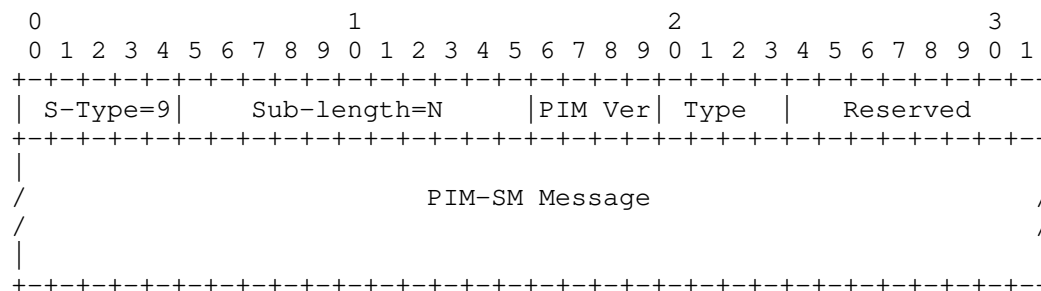


Figure 24: PIM-SM Message Option Format

- * Sub-Type is set to 9. If multiple instances appear in OMNI options of the same message all are processed.
- * Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the PIM-SM message. The length of the entire PIM-SM message is therefore limited by the remaining available space for this OMNI option.
- * The PIM-SM message is coded exactly as specified in Section 4.9 of [RFC7761], except that the Checksum field is replaced by a Reserved field set to 0 on transmission and ignored on reception. The "PIM Ver" field MUST encode the value 2, and the "Type" field encodes the PIM message type. (See Section 4.9 of [RFC7761] for a list of PIM-SM message types and formats.)

12.2.11. Fragmentation Report (FRAGREP)

Fragmentation Report (FRAGREP) sub-options may be included in the OMNI options of uNA messages sent from an OAL destination to an OAL source. The message consists of $(N / 20)$ -many (Identification, Bitmap)-tuples which include the Identification values of OAL fragments received plus a Bitmap marking the ordinal positions of individual fragments received and fragments missing.

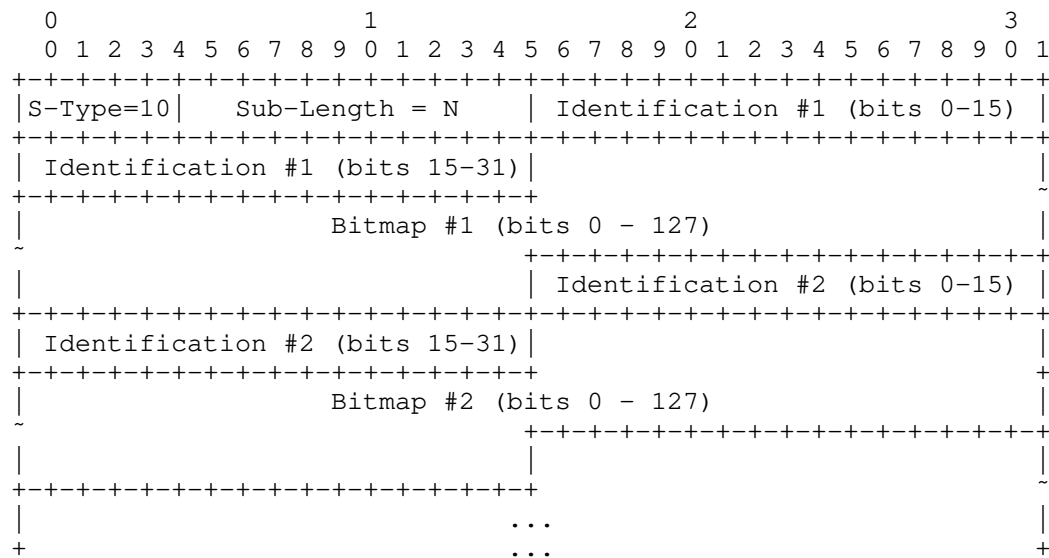


Figure 25: Fragmentation Report (FRAGREP)

- * Sub-Type is set to 10. If multiple instances appear in OMNI options of the same message all are processed.
- * Sub-Length is set to N, i.e., the length of the option in octets beginning immediately following the Sub-Length field and extending to the end of the sub-option. If N is not an integral multiple of 20 octets, the sub-option is ignored. The length of the entire sub-option should not cause the entire IPv6 ND message to exceed the minimum IPv6 MTU.
- * Identification (i) includes the IPv6 Identification value found in the Fragment Header of a received OAL fragment. (Only those Identification values included represent fragments for which loss was unambiguously observed; any Identification values not included correspond to fragments that were either received in their entirety or may still be in transit.)
- * Bitmap (i) includes an ordinal checklist of up to 128 fragments, with each bit set to 1 for a fragment received or 0 for a fragment missing. For example, for a 20-fragment OAL packet with ordinal fragments #3, #10, #13 and #17 missing and all other fragments received, Bitmap (i) encodes the following:

```

      0               1               2
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|1|1|1|0|1|1|1|1|1|1|0|1|1|0|1|1|1|0|1|1|0|0|0|...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 26

(Note that loss of an OAL atomic fragment is indicated by a Bitmap(i) with all bits set to 0.)

12.2.12. Node Identification

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|S-Type=11| Sub-length=N | ID-Type | ~
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
~ Node Identification Value (N-1 octets) ~
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 27: Node Identification

- * Sub-Type is set to 11. If multiple instances appear in OMNI options of the same IPv6 ND message the first instance of a specific ID-Type is processed and all other instances of the same ID-Type are ignored. (It is therefore possible for a single IPv6 ND message to convey multiple distinct Node Identifications - each with a different ID-Type.)
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The ID-Type field is always present; hence, the maximum Node Identification Value length is limited by the remaining available space in this OMNI option.
- * ID-Type is a 1 octet field that encodes the type of the Node Identification Value. The following ID-Type values are currently defined:
 - 0 - Universally Unique Identifier (UUID) [RFC4122]. Indicates that Node Identification Value contains a 16 octet UUID.
 - 1 - Host Identity Tag (HIT) [RFC7401]. Indicates that Node Identification Value contains a 16 octet HIT.
 - 2 - Hierarchical HIT (HHIT) [I-D.ietf-drip-rid]. Indicates that Node Identification Value contains a 16 octet HHIT.
 - 3 - Network Access Identifier (NAI) [RFC7542]. Indicates that Node Identification Value contains an N-1 octet NAI.
 - 4 - Fully-Qualified Domain Name (FQDN) [RFC1035]. Indicates that Node Identification Value contains an N-1 octet FQDN.
 - 5 - IPv6 Address. Indicates that Node Identification contains a 16-octet IPv6 address that is not a (H)HIT. The IPv6 address type is determined according to the IPv6 addressing architecture [RFC4291].
 - 6 - 252 - Unassigned.
 - 253-254 - Reserved for experimentation, as recommended in [RFC3692].
 - 255 - reserved by IANA.
- * Node Identification Value is an (N - 1) octet field encoded according to the appropriate the "ID-Type" reference above.

OMNI interfaces code Node Identification Values used for DHCPv6 messaging purposes as a DHCP Unique Identifier (DUID) using the "DUID-EN for OMNI" format with enterprise number 45282 (see: Section 25) as shown in Figure 28:

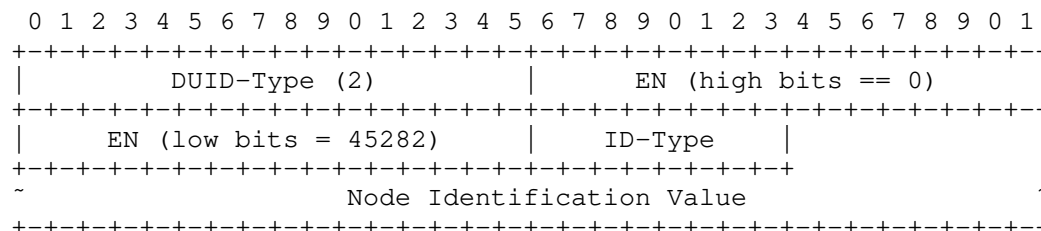


Figure 28: DUID-EN for OMNI Format

In this format, the OMNI interface codes the ID-Type and Node Identification Value fields from the OMNI sub-option following a 6 octet DUID-EN header, then includes the entire "DUID-EN for OMNI" in a DHCPv6 message per [RFC8415].

12.2.13. ICMPv6 Error

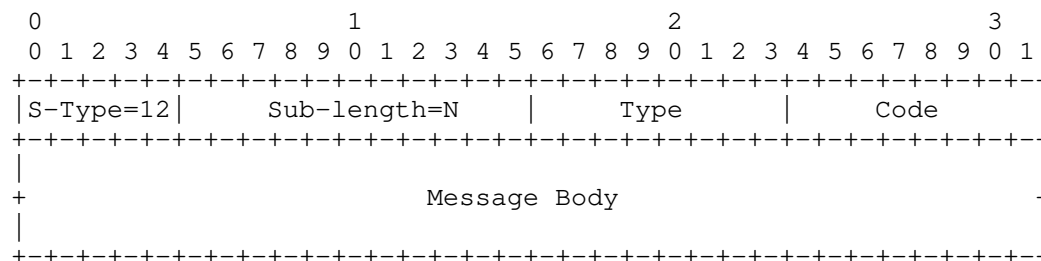


Figure 29: ICMPv6 Error

- * Sub-Type is set to 12. If multiple instances appear in OMNI options of the same IPv6 ND message all are processed.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.
- * Sub-Option Data includes a one octet Type followed by a one octet Code followed by an (N-2)-octet Message Body encoded exactly as per Section 2.1 of [RFC4443]. OMNI interfaces include as much of the ICMPv6 error message body in the sub-option as possible without causing the entire IPv6 ND message to exceed the minimum IPv6 MTU. While all ICMPv6 error message types are supported, OAL destinations in particular may include ICMPv6 PTB messages in uNA

messages to provide MTU feedback information via the OAL source (see: Section 6.8). Note: ICMPv6 informational messages must not be included and must be ignored if received.

12.2.14. QUIC-TLS Message

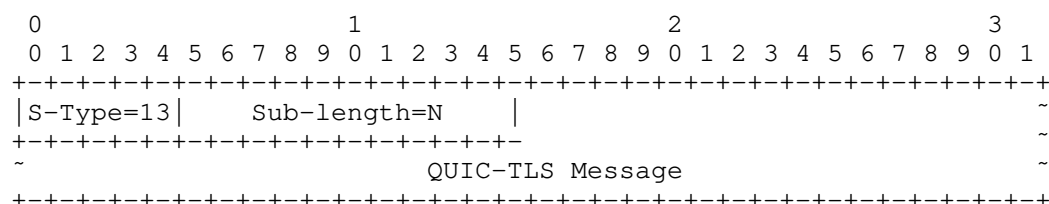


Figure 30: QUIC-TLS Message

- * Sub-Type is set to 13. If multiple instances appear in OMNI options of the same IPv6 ND message, the first is processed and all others are ignored.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow.
- * The QUIC-TLS message [RFC9000][RFC9001][RFC9002] encodes the QUIC and TLS message parameters necessary to support QUIC connection establishment.

When present, the QUIC-TLS Message sub-option MUST appear immediately after the header of the first OMNI option in the IPv6 ND message; if the sub-option appears in any other location it MUST be ignored. IPv6 ND solicitation and advertisement messages serve as couriers to transport the QUIC and TLS parameters necessary to establish a secured QUIC connection.

12.2.15. Proxy/Server Departure

OMNI Clients include a Proxy/Server Departure sub-option in RS messages when they associate with a new FHS and/or Hub Proxy/Server and need to send a departure indication to an old FHS and/or Hub Proxy/Server. The Proxy/Server Departure sub-option is formatted as shown below:

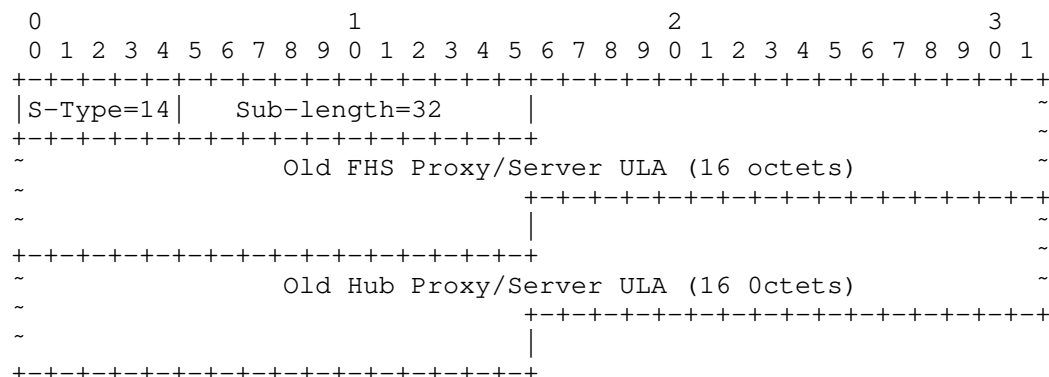


Figure 31: Proxy/Server Departure

- * Sub-Type is set to 14.
- * Sub-Length is set to 32.
- * Sub-Option Data contains the 16 octet ULA for the "Old FHS Proxy/Server" followed by a 16 octet ULA for an "Old Hub Proxy/Server". (If the Old FHS/Hub is unspecified, the corresponding ULA instead includes the value 0.)

12.2.16. Sub-Type Extension

Since the Sub-Type field is only 5 bits in length, future specifications of major protocol functions may exhaust the remaining Sub-Type values available for assignment. This document therefore defines Sub-Type 30 as an "extension", meaning that the actual Sub-Type type is determined by examining a 1 octet "Extension-Type" field immediately following the Sub-Length field. The Sub-Type Extension is formatted as shown in Figure 32:

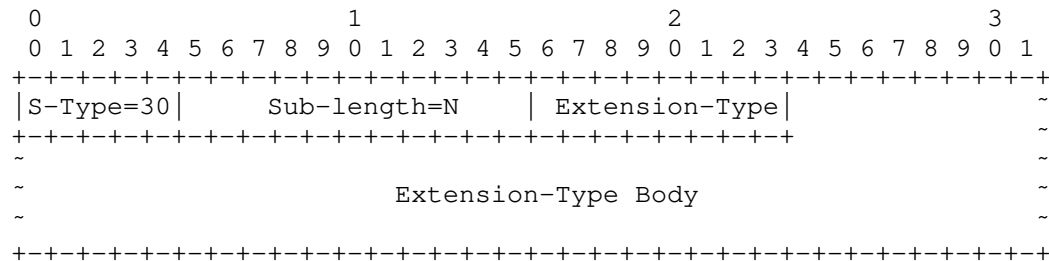


Figure 32: Sub-Type Extension

- * Sub-Type is set to 30. If multiple instances appear in OMNI options of the same message all are processed, where each individual extension defines its own policy for processing multiple of that type.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The Extension-Type field is always present, and the maximum Extension-Type Body length is limited by the remaining available space in this OMNI option.
- * Extension-Type contains a 1 octet Sub-Type Extension value between 0 and 255.
- * Extension-Type Body contains an N-1 octet block with format defined by the given extension specification.

Extension-Type values 0 and 1 are defined in the following subsections, while Extension-Type values 2 through 252 are available for assignment by future specifications which must also define the format of the Extension-Type Body and its processing rules. Extension-Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is reserved by IANA.

12.2.16.1. RFC4380 Header Extension Option

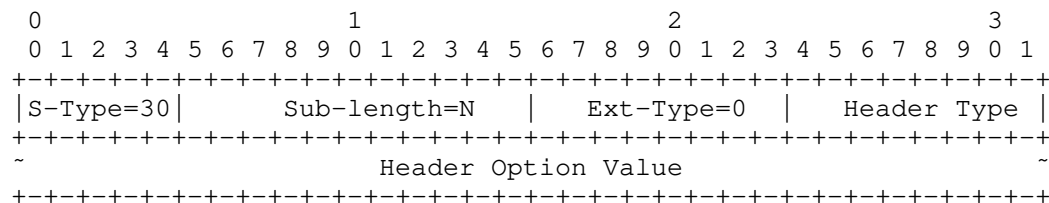


Figure 33: RFC4380 Header Extension Option (Extension-Type 0)

- * Sub-Type is set to 30.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The Extension-Type and Header Type fields are always present, and the Header Option Value is limited by the remaining available space in this OMNI option.
- * Extension-Type is set to 0. Each instance encodes exactly one header option per Section 5.1.1 of [RFC4380], with Ext-Type and Header Type representing the first two octets of the option. If multiple instances of the same Header Type appear in OMNI options of the same message the first instance is processed and all others are ignored. If Header Type indicates an Authentication

Encapsulation (see below), the entire sub-option MUST appear as the first sub-option of the first OMNI option, which MUST appear immediately following the IPv6 ND message header.

- * Header Type and Header Option Value are coded exactly as specified in Section 5.1.1 of [RFC4380]; the following types are currently defined:
 - 0 - Origin Indication (IPv4) - value coded as a UDP port number followed by a 4-octet IPv4 address both in "obfuscated" form per Section 5.1.1 of [RFC4380].
 - 1 - Authentication Encapsulation - value coded per Section 5.1.1 of [RFC4380].
 - 2 - Origin Indication (IPv6) - value coded per Section 5.1.1 of [RFC4380], except that the address is a 16-octet IPv6 address instead of a 4-octet IPv4 address.
- * Header Type values 3 through 252 are available for assignment by future specifications, which must also define the format of the Header Option Value and its processing rules. Header Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is Reserved by IANA.

12.2.16.2. RFC6081 Trailer Extension Option

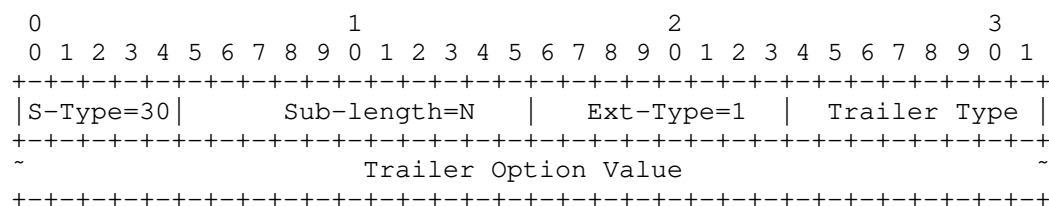


Figure 34: RFC6081 Trailer Extension Option (Extension-Type 1)

- * Sub-Type is set to 30.
- * Sub-Length is set to N that encodes the number of Sub-Option Data octets that follow. The Extension-Type and Trailer Type fields are always present, and the maximum-length Trailer Option Value is limited by the remaining available space in this OMNI option.
- * Extension-Type is set to 1. Each instance encodes exactly one trailer option per Section 4 of [RFC6081]. If multiple instances of the same Trailer Type appear in OMNI options of the same message the first instance is processed and all others ignored.

- * Trailer Type and Trailer Option Value are coded exactly as specified in Section 4 of [RFC6081]; the following Trailer Types are currently defined:
 - 0 - Unassigned
 - 1 - Nonce Trailer - value coded per Section 4.2 of [RFC6081].
 - 2 - Unassigned
 - 3 - Alternate Address Trailer (IPv4) - value coded per Section 4.3 of [RFC6081].
 - 4 - Neighbor Discovery Option Trailer - value coded per Section 4.4 of [RFC6081].
 - 5 - Random Port Trailer - value coded per Section 4.5 of [RFC6081].
 - 6 - Alternate Address Trailer (IPv6) - value coded per Section 4.3 of [RFC6081], except that each address is a 16-octet IPv6 address instead of a 4-octet IPv4 address.
- * Trailer Type values 7 through 252 are available for assignment by future specifications, which must also define the format of the Trailer Option Value and its processing rules. Trailer Type values 253 and 254 are reserved for experimentation, as recommended in [RFC3692], and value 255 is Reserved by IANA.

13. Address Mapping - Multicast

The multicast address mapping of the native underlay interface applies. The Client mobile router also serves as an IGMP/MLD Proxy for its ENETs and/or hosted applications per [RFC4605].

The Client uses Multicast Listener Discovery (MLDv2) [RFC3810] to coordinate with Proxy/Servers, and underlay network elements use MLD snooping [RFC4541]. The Client can also employ multicast routing protocols to coordinate with network-based multicast sources as specified in [I-D.templin-6man-aero].

Since the OMNI link model is NBMA, OMNI links support link-scoped multicast through iterative unicast transmissions to individual multicast group members (i.e., unicast/multicast emulation).

14. Multilink Conceptual Sending Algorithm

The Client's IPv6 layer selects the outbound OMNI interface according to SBM considerations when forwarding original IP packets from local or ENET applications to external correspondents. Each OMNI interface maintains a neighbor cache the same as for any IPv6 interface, but includes additional state for multilink coordination. Each Client OMNI interface maintains default routes via Proxy/Servers discovered as discussed in Section 15, and may configure more-specific routes discovered through means outside the scope of this specification.

For each original IP packet it forwards, the OMNI interface selects one or more source underlay interfaces based on PBM factors (e.g., traffic attributes, cost, performance, message size, etc.) and one or more target underlay interfaces for the neighbor based on Interface Attributes received in IPv6 ND messages (see: Section 12.2.4). Multilink forwarding may also direct packet replication across multiple underlay interface pairs for increased reliability at the expense of duplication. The set of all Interface Attributes and Traffic Selectors received in IPv6 ND messages determines the multilink forwarding profile for selecting target underlay interfaces.

When the OMNI interface sends an original IP packet over a selected source underlay interface, it first employs OAL encapsulation and fragmentation as discussed in Section 5, then performs L2 encapsulation as directed by the appropriate MFV. The OMNI interface also performs L2 encapsulation (following OAL encapsulation) when the nearest Proxy/Server is located multiple hops away as discussed in Section 15.2.

OMNI interface multilink service designers MUST observe the BCP guidance in Section 15 [RFC3819] in terms of implications for reordering when original IP packets from the same flow may be spread across multiple underlay interfaces having diverse properties.

14.1. Multiple OMNI Interfaces

Clients may connect to multiple independent OMNI links within the same or different OMNI domains to support SBM. The Client configures a separate OMNI interface for each link so that multiple interfaces (e.g., omni0, omni1, omni2, etc.) are exposed to the IP layer. Each OMNI interface configures one or more OMNI anycast addresses (see: Section 10), and the Client injects the corresponding anycast prefixes into the ENET routing system. Multiple distinct OMNI links can therefore be used to support fault tolerance, load balancing, reliability, etc.

Applications in ENETs can use Segment Routing to select the desired OMNI interface based on SBM considerations. The application writes an OMNI anycast address into the original IP packet's destination address, and writes the actual destination (along with any additional intermediate hops) into the Segment Routing Header. Standard IP routing directs the packet to the Client's mobile router entity, where the anycast address identifies the correct OMNI interface for next hop forwarding. When the Client receives the packet, it replaces the IP destination address with the next hop found in the Segment Routing Header and forwards the message via the OMNI interface identified by the anycast address.

Note: The Client need not configure its OMNI interface indexes in one-to-one correspondence with the global OMNI Link-IDs configured for OMNI domain administration since the Client's indexes (i.e., omni0, omni1, omni2, etc.) are used only for its own local interface management.

14.2. Client-Proxy/Server Loop Prevention

After a Proxy/Server has registered an MNP for a Client (see: Section 15), the Proxy/Server will forward all packets destined to an address within the MNP to the Client. The Client will under normal circumstances then forward the packet to the correct destination within its connected (downstream) ENETs.

If at some later time the Client loses state (e.g., after a reboot), it may begin returning packets with destinations corresponding to its MNP to the Proxy/Server as its default router. The Proxy/Server therefore drops any original IP packets received from the Client with a destination address that corresponds to the Client's MNP (i.e., whether ULA or GUA), and drops any carrier packets with both source and destination address corresponding to the same Client's MNP regardless of their origin.

15. Router Discovery and Prefix Registration

Clients engage the MS by sending RS messages with OMNI options under the assumption that one or more Proxy/Server will process the message and respond. The RS message is received by a FHS Proxy/Server, which may in turn forward a proxied copy of the RS to a Hub Proxy/Server located on the same or different SRT segment. The Hub Proxy/Server then returns an RA message either directly to the Client or via an FHS Proxy/Server acting as a proxy.

Clients and FHS Proxy/Servers include an authentication signature in their RS/RA exchanges when necessary; otherwise, they calculate and include a valid IPv6 ND message checksum (see: Section 12 and

Appendix B). FHS and Hub Proxy/Server RS/RA message exchanges over the SRT secured spanning tree instead always include the checksum and omit the authentication signature. Clients and Proxy/Servers use the information included in RS/RA messages to establish NCE state and OMNI link autoconfiguration information as discussed in this section.

For each underlay interface, the Client sends RS messages with OMNI options to coordinate with a (potentially) different FHS Proxy/Server for each interface but with a single Hub Proxy/Server. All Proxy/Servers are identified by their ULA-RNDs and accept carrier packets addressed to their anycast/unicast L2 INADDRs; the Hub Proxy/Server may be chosen among any of the Client's FHS Proxy/Servers or may be any other Proxy/Server for the OMNI link. Example ULA/INADDR discovery methods are given in [RFC5214] and include data link login parameters, name service lookups, static configuration, a static "hosts" file, etc. In the absence of other information, the Client can resolve the DNS Fully-Qualified Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the OMNI link (e.g., "example.com"). The name resolution will retain a set of DNS resource records with the addresses of Proxy/Servers for the domain.

Each FHS Proxy/Server configures an ULA-RND based on a /64 ULA prefix for the link/segment with randomly-generated Global ID to assure global uniqueness then administratively assigned to FHS Proxy/Servers for the link to assure global consistency. The Client can then configure ULA-MNPs derived from the 64-bit ULA prefix assigned to a FHS Proxy/Server for each underlay interface. The FHS Proxy/Servers discovered over multiple of the Client's underlay interfaces may configure the same or different ULA prefixes, and the Client's ULA-MNP for each underlay interface will fall within the ULA (multilink) subnet relative to each FHS Proxy/Server.

Clients configure OMNI interfaces that observe the properties discussed in previous sections. The OMNI interface and its underlay interfaces are said to be in either the "UP" or "DOWN" state according to administrative actions in conjunction with the interface connectivity status. An OMNI interface transitions to UP or DOWN through administrative action and/or through state transitions of the underlay interfaces. When a first underlay interface transitions to UP, the OMNI interface also transitions to UP. When all underlay interfaces transition to DOWN, the OMNI interface also transitions to DOWN.

When a Client OMNI interface transitions to UP, it sends RS messages to register its MNP and an initial set of underlay interfaces that are also UP. The Client sends additional RS messages to refresh lifetimes and to register/deregister underlay interfaces as they

transition to UP or DOWN. The Client's OMNI interface sends initial RS messages over an UP underlay interface with its {TLA,XLA}-MNP as the source (or with a {TLA,XLA}-RND as the source if it does not yet have an MNP) and with destination set to link-scoped All-Routers multicast or the ULA of a specific (Hub) Proxy/Server. The OMNI interface includes an OMNI option per Section 12 with an OMNI Neighbor Coordination sub-option with (Preflen assertion, N/A/U flags and Window Synchronization parameters), an Interface Attributes sub-option for the underlay interface, a DHCPv6 Solicit sub-option if necessary, and with any other necessary OMNI sub-options such as authentication, Proxy/Server Departure, etc.

The Client then calculates the authentication signature or checksum and prepares to forward the RS over the underlay interface using OAL encapsulation and fragmentation if necessary. If the Client uses OAL encapsulation for RS messages sent to an unsynchronized FHS Proxy/Server over an INET interface, the entire RS message must fit within a single carrier packet (i.e., an atomic fragment) so that the FHS Proxy/Server can verify the authentication signature without having to reassemble. The OMNI interface selects an Identification value (see: Section 6.6), sets the OAL source address to the ULA-MNP corresponding to the RS source if known (otherwise to a {TLA,XLA}-RND), sets the OAL destination to an OMNI IPv6 anycast address or a known Proxy/Server ULA, optionally includes a Nonce and/or Timestamp, then performs fragmentation if necessary. When L2 encapsulation is used, the Client includes the discovered FHS Proxy/Server INADDR or an anycast address as the L2 destination then forwards the resulting carrier packet(s) into the underlay network. Note that the Client does not yet create a NCE, but instead remembers the Identification, Nonce and/or Timestamp values included in its RS message transmissions to match against any received RA messages.

When an FHS Proxy/Server receives the carrier packets containing an RS it sets aside the L2 headers, verifies the Identifications and reassembles if necessary, sets aside the OAL header, then verifies the RS authentication signature or checksum. The FHS Proxy/Server then creates/updates a NCE indexed by the Client's RS source address and caches the OMNI Interface Attributes and any Traffic Selector sub-options while also caching the L2 (UDP/IP) and OAL source and destination address information. The FHS Proxy/Server next caches the OMNI Neighbor Coordination sub-option Window Synchronization parameters and N flag to determine its role in processing NS(NUD) messages (see: Section 12.1) then examines the RS destination address. If the destination matches its own ULA, the FHS Proxy/Server assumes the Hub role and acts as the sole entry point for injecting the Client's XLA-MNP into the OMNI link routing system (i.e., after performing any necessary prefix delegation operations) while including a prefix length and setting the prefix to fd00::/64

and suffix to the 64-bit MNP. The FHS/Hub Proxy/Server then caches the OMNI Neighbor Coordination sub-option A/U flags to determine its role in processing NS(AR) messages and generating uNA messages (see: Section 12.1).

The FHS/Hub Proxy/Server then prepares to return an RA message directly to the Client by first populating the Cur Hop Limit, Flags, Router Lifetime, Reachable Time and Retrans Timer fields with values appropriate for the OMNI link. The FHS/Hub Proxy/Server next includes as the first RA message option an OMNI option with a neighbor coordination sub-option with Window Synchronization information, an authentication sub-option if necessary and a (proxied) copy of the Client's original Interface Attributes sub-option with its INET-facing interface information written in the FMT/SRT and LHS Proxy/Server ULA/INADDR fields. If the RS L2 destination IP address was anycast, the FHS/Hub Proxy/Server next includes a second Interface Attributes sub-option with omIndex set to '0' and with a unicast L2 IP address for its Client-facing interface in the INADDR field.

The FHS/Hub Proxy/Server next includes an Origin Indication sub-option that includes the RS L2 source INADDR information (see: Section 12.2.16.1), then includes any other necessary OMNI sub-options (either within the same OMNI option or in additional OMNI options). Following the OMNI option(s), the FHS/Hub Proxy/Server next includes any other necessary RA options such as PIOs with (A; L=0) that include the OMNI link MSPs [RFC8028], RIOs [RFC4191] with more-specific routes, Nonce and Timestamp options, etc. The FHS/Hub Proxy/Server then sets the RA source address to its own ULA and destination address to the Client's ULA-MNP (i.e., relative to the ULA /64 prefix for its Client-facing underlay interface) while also recording the corresponding XLA-MNP as an (alternate) index to the Client NCE, then calculates the authentication signature or checksum. The FHS/Hub Proxy/Server finally performs OAL encapsulation with source set to its own ULA and destination set to the OAL source that appeared in the RS, then fragments if necessary, encapsulates each fragment in appropriate L2 headers with source and destination address information reversed from the RS L2 information and returns the resulting carrier packets to the Client over the same underlay interface the RS arrived on.

When an FHS Proxy/Server receives an RS with a valid authentication signature or checksum and with destination set to link-scoped All-Routers multicast, it can either assume the Hub role itself the same as above or act as a proxy and select the ULA of another Proxy/Server to serve as the Hub. When an FHS Proxy/Server assumes the proxy role or receives an RS with destination set to the ULA of another Proxy/Server, it forwards the message while acting as a proxy. The FHS

Proxy/Server creates/updates a NCE for the Client (i.e., based on the RS source address) and caches the OAL source, Window Synchronization, N flag, Interface Attributes addressing information as above then writes its own INET-facing FMT/SRT and LHS Proxy/Server ULA/INADDR information into the appropriate Interface Attributes sub-option fields. The FHS Proxy/Server then calculates and includes the checksum, performs OAL encapsulation with source set to its own ULA and destination set to the ULA of the Hub Proxy/Server, fragments if necessary, encapsulates each fragment in appropriate L2 headers and sends the resulting carrier packets into the SRT secured spanning tree.

When the Hub Proxy/Server receives the carrier packets, it discards the L2 headers, reassembles if necessary to obtain the proxied RS, then performs DHCPv6 Prefix Delegation (PD) to obtain the Client's MNP if the RS source is a {TLA,XLA}-RND. The Hub Proxy/Server then creates/updates a NCE for the Client's XLA-MNP and caches any state (including the A/U flags, OAL addresses, Interface Attributes information and Traffic Selectors), then finally performs routing protocol injection. The Hub Proxy/Server then returns an RA that echoes the Client's (proxied) Interface Attributes sub-option and with any RA parameters the same as specified for the FHS/Hub Proxy/Server case above. The Hub Proxy/Server then sets the RA source address to its own ULA and destination address to the RS source address; if the RS source address is a {TLA,XLA}-RND, the Hub Proxy/Server also includes the MNP in a DHCPv6 PD Reply OMNI sub-option. The Hub Proxy/Server next calculates the checksum, then encapsulates the RA as an OAL packet with source set to its own ULA and destination set to the ULA of the FHS Proxy/Server that forwarded the RS. The Hub Proxy/Server finally fragments if necessary, encapsulates each fragment in appropriate L2 headers and sends the resulting carrier packets into the secured spanning tree.

When the FHS Proxy/Server receives the carrier packets it discards the L2 headers, reassembles if necessary to obtain the RA message, verifies the checksum then updates the OMNI interface NCE for the Client and creates/updates a NCE for the Hub. The FHS Proxy/Server then sets the P flag in the RA flags field [RFC4389] and proxys the RA by changing the OAL source to its own ULA, changing the OAL destination to the OAL address found in the Client's NCE, and changing the RA destination address to the ULA-MNP of the Client relative to its own /64 ULA prefix while also recording the corresponding XLA-MNP as an alternate index into the Client NCE. (If the RA destination address was a {TLA,XLA}-RND, the FHS Proxy Server determines the MNP by consulting the DHCPv6 PD Reply message sub-option.) The FHS Proxy/Server next includes Window Synchronization parameters responsive to those in the Client's RS, an Interface Attributes sub-option with omIndex '0' and with its unicast L2 IP

address if necessary (see above), an Origin Indication sub-option with the Client's cached INADDR and an authentication sub-option if necessary. The FHS Proxy/Server finally selects an Identification value per Section 6.6, calculates the authentication signature or checksum, fragments if necessary, encapsulates each fragment in L2 headers with addresses taken from the Client's NCE and returns the resulting carrier packets via the same underlay interface over which the RS was received.

When the Client receives the carrier packets, it discards the L2 headers, reassembles if necessary and removes the OAL header to obtain the RA message. The Client next verifies the authentication signature or checksum, then matches the RA message with its previously-sent RS by comparing the RS Sequence Number with the RA Acknowledgement Number and also comparing the Nonce and/or Timestamp values if present. If the values match, the Client then creates/updates OMNI interface NCEs for both the Hub and FHS Proxy/Server and caches the information in the RA message. In particular, the Client caches the RA source address as the Hub Proxy/Server ULA and uses the OAL source address to configure both an underlay interface-specific ULA for the Hub Proxy/Server and the ULA of this FHS Proxy/Server. The Client then uses the ULA-MNP in the RA destination address to configure its address within the ULA (multilink) subnet prefix of the FHS Proxy/Server. If the Client has multiple underlay interfaces, it creates additional FHS Proxy/Server NCEs and ULA-MNPs as necessary when it receives RAs over those interfaces (noting that multiple of the Client's underlay interfaces may be serviced by the same or different FHS Proxy/Servers). The Client finally adds the Hub Proxy/Server ULA to the default router list if necessary.

For each underlay interface, the Client next caches the (filled-out) Interface Attributes for its own omIndex and Origin Indication information that it received in an RA message over that interface so that it can include them in future NS/NA messages to provide neighbors with accurate FMT/SRT/LHS information. (If the message includes an Interface Attributes sub-option with omIndex '0', the Client also caches the INADDR as the underlay network-local unicast address of the FHS Proxy//Server via that underlay interface.) The Client then compares the Origin Indication INADDR information with its own underlay interface addresses to determine whether there may be NATs on the path to the FHS Proxy/Server; if the INADDR information differs, the Client is behind a NAT and must supply the Origin information in IPv6 ND message exchanges with prospective neighbors on the same SRT segment. The Client finally configures default routes and assigns the OMNI Subnet Router Anycast address corresponding to the MNP (e.g., 2001:db8:1:2::) to the OMNI interface.

Following the initial exchange, the FHS Proxy/Server MAY later send additional periodic and/or event-driven unsolicited RA messages per [RFC4861]. (The unsolicited RAs may be initiated either by the FHS Proxy/Server itself or by the Hub via the FHS as a proxy.) The Client then continuously manages its underlay interfaces according to their states as follows:

- * When an underlay interface transitions to UP, the Client sends an RS over the underlay interface with an OMNI option with sub-options as specified above.
- * When an underlay interface transitions to DOWN, the Client sends unsolicited NA messages over any UP underlay interface with an OMNI option containing Interface Attributes sub-options for the DOWN underlay interface with Link set to '0'. The Client sends isolated unsolicited NAs when reliability is not thought to be a concern (e.g., if redundant transmissions are sent on multiple underlay interfaces), or may instead set the PNG flag in the OMNI header to trigger a uNA reply.
- * When the Router Lifetime for the Hub Proxy/Server nears expiration, the Client sends an RS over any underlay interface to receive a fresh RA from the Hub. If no RA messages are received over a first underlay interface (i.e., after retrying), the Client marks the underlay interface as DOWN and should attempt to contact the Hub Proxy/Server via a different underlay interface. If the Hub Proxy/Server is unresponsive over additional underlay interfaces, the Client sends an RS message with destination set to the ULA of another Proxy/Server which will then assume the Hub role.
- * When all of a Client's underlay interfaces have transitioned to DOWN (or if the prefix registration lifetime expires), the Hub Proxy/Server withdraws the MNP the same as if it had received a message with a release indication.

The Client is responsible for retrying each RS exchange up to MAX_RTR_SOLICITATIONS times separated by RTR_SOLICITATION_INTERVAL seconds until an RA is received. If no RA is received over an UP underlay interface (i.e., even after attempting to contact alternate Proxy/Servers), the Client declares this underlay interface as DOWN. When changing to a new FHS or Hub Proxy/Server, the Client also includes a Proxy/Server Departure OMNI sub-option in new RS messages; the (new) FHS Proxy/Server will in turn send uNA messages to the old FHS and/or Hub Proxy/Server to announce the Client's departure as discussed in [I-D.templin-6man-aero].

The IPv6 layer sees the OMNI interface as an ordinary IPv6 interface. Therefore, when the IPv6 layer sends an RS message the OMNI interface returns an internally-generated RA message as though the message originated from an IPv6 router. The internally-generated RA message contains configuration information consistent with the information received from the RAs generated by the Hub Proxy/Server. Whether the OMNI interface IPv6 ND messaging process is initiated from the receipt of an RS message from the IPv6 layer or independently of the IPv6 layer is an implementation matter. Some implementations may elect to defer the OMNI interface internal RS/RA messaging process until an RS is received from the IPv6 layer, while others may elect to initiate the process proactively. Still other deployments may elect to administratively disable IPv6 layer RS/RA messaging over the OMNI interface, since the messages are not required to drive the OMNI interface internal RS/RA process. (Note that this same logic applies to IPv4 implementations that employ "ICMP Router Discovery" [RFC1256].)

Note: The Router Lifetime value in RA messages indicates the time before which the Client must send another RS message over this underlay interface (e.g., 600 seconds), however that timescale may be significantly longer than the lifetime the MS has committed to retain the prefix registration (e.g., REACHABLETIME seconds). Proxy/Servers are therefore responsible for keeping MS state alive on a shorter timescale than the Client may be required to do on its own behalf.

Note: On certain multicast-capable underlay interfaces, Clients should send periodic unsolicited multicast NA messages and Proxy/Servers should send periodic unsolicited multicast RA messages as "beacons" that can be heard by other nodes on the link. If a node fails to receive a beacon after a timeout value specific to the link, it can initiate Neighbor Unreachability Detection (NUD) exchanges to test reachability.

Note: If a single FHS Proxy/Server services multiple of a Client's underlay interfaces, Window Synchronization will initially be repeated for the RS/RA exchange over each underlay interface, i.e., until the Client discovers the many-to-one relationship. This will naturally result in a single window synchronization that applies over the Client's multiple underlay interfaces for the same FHS Proxy/Server.

Note: Although the Client's FHS Proxy/Server is a first-hop segment node from its own perspective, the Client stores the Proxy/Server's FMT/SRT/ULA/INADDR as last-hop segment (LHS) information to supply to neighbors. This allows both the Client and Hub Proxy/Server to supply the information to neighbors that will perceive it as LHS information on the return path to the Client.

Note: The Hub Proxy/Server injects Client XLA-MNP into the OMNI link routing system by simply creating a route-to-interface forwarding table entry for fd00::{MNP}/N via the OMNI interface. The dynamic routing protocol will notice the new entry and propagate the route to its peers. If the Hub receives additional RS messages, it need not re-create the forwarding table entry (nor disturb the dynamic routing protocol) if an entry is already present. If the Hub ceases to receive RS messages from any of the Client's interfaces, it removes the Client XLA-MNP from the forwarding table (i.e., after a short delay) resulting in its removal also from the routing system.

Note: If the Client's initial RS message includes an anycast L2 destination address, the FHS Proxy/Server returns the solicited RA using the same anycast address as the L2 source while including an Interface Attributes sub-option with omIndex '0' and its true unicast address in the INADDR. When the Client sends additional RS messages, it includes this FHS Proxy/Server unicast address as the L2 destination and the FHS Proxy/Server returns the solicited RA using the same unicast address as the L2 source. This will ensure that RS/RA exchanges are not impeded by any NATs on the path while avoiding long-term exposure of messages that use an anycast address as the source.

Note: The Origin Indication sub-option is included only by the FHS Proxy/Server and not by the Hub (unless the Hub is also serving as an FHS).

Note: Clients should set the N/A/U flags consistently in successive RS messages and only change those settings when an FHS/Hub Proxy/Server service profile update is necessary.

Note: After a Client has discovered its ULA-MNPs for a given set of FHS Proxy/Servers, it should begin using its XLA-MNP as the IPv6 ND message source address and ULA-MNP as the OAL source address in future IPv6 ND messages and refrain from further use of TLAs. In any case, the Client SHOULD NOT gratuitously configure and use large numbers of additional TLAs, as doing so would simply result in address change churn in neighbor cache entries with no operational advantages.

Note: Although the Client adds the Hub Proxy/Server ULA to the default router list, it also caches the ULAs of the FHS Proxy/Servers on the path to the Hub over each underlying interface. When the Client needs to send a packet to a default router, it therefore selects an ULA corresponding to the selected interface which directs the packet to an FHS Proxy/Server for that interface. The FHS Proxy/Server then forwards the packet without disturbing the Hub.

15.1. Window Synchronization

In environments where Identification window synchronization is necessary, the RS/RA exchanges discussed above observe the principles specified in Section 6.6. Window synchronization is conducted between the Client and each FHS Proxy/Server used to contact the Hub Proxy/Server, i.e., and not between the Client and the Hub. This is due to the fact that the Hub Proxy/Server is responsible only for forwarding control and data messages via the secured spanning tree to FHS Proxy/Servers, and is not responsible for forwarding messages directly to the Client under a synchronized window. Also, in the reverse direction the FHS Proxy/Servers handle all default forwarding actions without forwarding Client-initiated data to the Hub.

When a Client needs to perform window synchronization via a new FHS Proxy/Server, it sets the RS source address to its own {TLA,XLA}-MNP (or a {TLA,XLA}-RND) and destination address to the ULA of the Hub Proxy/Server (or to All-Routers multicast in an initial RS), then sets the SYN flag and includes an initial Sequence Number for Window Synchronization. The Client then performs OAL encapsulation using its own ULA-MNP (or the TLA-RND) as the source and the ULA of the FHS Proxy/Server as the destination and includes an Interface Attributes sub-option then forwards the resulting carrier packets to the FHS Proxy/Server. The FHS Proxy/Server then extracts the RS message and caches the Window Synchronization parameters then re-encapsulates with its own ULA as the source and the ULA of the Hub Proxy/Server as the target.

The FHS Proxy/Server then forwards the resulting carrier packets via the secured spanning tree to the Hub Proxy/Server, which updates the Client's Interface Attributes and returns a unicast RA message with source set to its own ULA and destination set to the RS source address and with the Client's Interface Attributes echoed. The Hub Proxy/Server then performs OAL encapsulation using its own ULA as the source and the ULA of the FHS Proxy/Server as the destination, then forwards the carrier packets via the secured spanning tree to the FHS Proxy/Server. The FHS Proxy/Server then proxys the message as discussed in the previous section and includes responsive Window Synchronization information. The FHS Proxy/Server then forwards the message to the Client which updates its window synchronization information for the FHS Proxy/Server as necessary.

Following the initial RS/RA-driven window synchronization, the Client can re-assert new windows with specific FHS Proxy/Servers by performing NS/NA exchanges between its own XLA-MNPs and the ULAs of the FHS Proxy/Servers without having to disturb the Hub.

15.2. Router Discovery in IP Multihop and IPv4-Only Networks

On some *NETs, a Client may be located multiple IP hops away from the nearest OMNI link Proxy/Server. Forwarding through IP multihop *NETs is conducted through the application of a routing protocol (e.g., a MANET/VANET routing protocol over omni-directional wireless interfaces, an inter-domain routing protocol in an enterprise network, etc.). Example routing protocols optimized for MANET/VANET operations include [RFC3684] and [RFC5614] which operate according to the link model articulated in [RFC5889] and subnet model articulated in [RFC5942].

A Client located potentially multiple *NET hops away from the nearest Proxy/Server prepares an RS message, sets the source address to its {TLA,XLA}-MNP (or to a {TLA,XLA}-RND if it does not yet have an MNP), and sets the destination to link-scoped All-Routers multicast or the unicast ULA of a Proxy/Server the same as discussed above. The OMNI interface then employs OAL encapsulation, sets the OAL source address to a TLA and sets the OAL destination to an OMNI IPv6 anycast address based on either a native IPv6 or IPv4-Compatible IPv6 prefix (see: Section 10).

For IPv6-enabled *NETs, if the underlay interface does not configure an IPv6 GUA the Client injects the TLA into the IPv6 multihop routing system and forwards the message without further encapsulation. Otherwise, the Client encapsulates the message in UDP/IPv6 L2 headers, sets the source to the underlay interface IPv6 address and sets the destination to the same OMNI IPv6 anycast address. The Client then forwards the message into the IPv6 multihop routing system which conveys it to the nearest Proxy/Server that advertises a matching OMNI IPv6 anycast prefix. If the nearest Proxy/Server is too busy, it should forward (without Proxying) the OAL-encapsulated RS to another nearby Proxy/Server connected to the same IPv6 (multihop) network that also advertises the matching OMNI IPv6 anycast prefix.

For IPv4-only *NETs, the Client encapsulates the RS message in UDP/IPv4 L2 headers, sets the source to the underlay interface IPv4 address and sets the destination to the OMNI IPv4 anycast address. The Client then forwards the message into the IPv4 multihop routing system which conveys it to the nearest Proxy/Server that advertises the corresponding IPv4 prefix. If the nearest Proxy/Server is too busy and/or does not configure the specified OMNI IPv6 anycast address, it should forward (without Proxying) the OAL-encapsulated RS to another nearby Proxy/Server connected to the same IPv4 (multihop) network that configures the OMNI IPv6 anycast address. (In environments where reciprocal RS forwarding cannot be supported, the first Proxy/Server should instead return an RA based on its own MSP(s).)

When an intermediate *NET hop that participates in the routing protocol receives the encapsulated RS, it forwards the message according to its routing tables (note that an intermediate node could be a fixed infrastructure element such as a roadside unit or another MANET/VANET node). This process repeats iteratively until the RS message is received by a penultimate *NET hop within single-hop communications range of a Proxy/Server, which forwards the message to the Proxy/Server.

When a Proxy/Server that configures the OMNI IPv6 anycast OAL destination receives the message, it decapsulates the RS and assumes either the Hub or FHS role (in which case, it forwards the RS to a candidate Hub). The Hub Proxy/Server then prepares an RA message with source address set to its own ULA and destination address set to the RS source address if it is acting only as the Hub (or to the Client ULA-MNP within its ULA subnet prefix if it is also acting as the FHS Proxy/Server). The Hub Proxy/Server then performs OAL encapsulation with the RA OAL source/destination set to the RS OAL destination/source and forwards the RA either to the FHS Proxy/Server or directly to the Client.

When the Hub or FHS Proxy/Server forwards the RA to the Client, it encapsulates the message in L2 encapsulation headers (if necessary) with (src, dst) set to the (dst, src) of the RS L2 encapsulation headers. The Proxy/Server then forwards the message to a *NET node within communications range, which forwards the message according to its routing tables to an intermediate node. The multihop forwarding process within the *NET continues repetitively until the message is delivered to the original Client, which decapsulates the message and performs autoconfiguration the same as if it had received the RA directly from a Proxy/Server on the same physical link. The Client then injects the ULA-MNP into the IPv6 multihop routing system if necessary, then begins using the ULA-MNP as its OAL source address and suspends use of its TLA since it now has a unique address within the FHS Proxy/Server's "Multilink Subnet".

Note: When the RS message includes anycast OAL and/or L2 encapsulation destinations, the FHS Proxy/Server must use the same anycast addresses as the OAL and/or L2 encapsulation sources to support forwarding of the RA message and any initial data packets over any NATs on the path. When the Client receives the RA, it will discover its unicast ULA-MNP and/or L2 encapsulation addresses and can forward future packets using the unicast (instead of anycast) addresses to populate NAT state in the forward path. (If the Client does not have immediate data to send to the FHS Proxy/Server, it can instead send an OAL "bubble" - see Section 6.10.) After the Client begins using unicast OAL/L2 encapsulation addresses in this way, the FHS Proxy/Server should also begin using the same unicast addresses in the reverse direction.

Note: When an OMNI interface configures a TLA, any nodes that forward an encapsulated RS message with the ULA as the OAL source must not consider the message as being specific to a particular OMNI link. TLAs can therefore also serve as the source and destination addresses of unencapsulated IPv6 data communications within the local routing region, and if the TLAs are injected into the local network routing protocol their prefix length must be set to 128.

15.3. DHCPv6-based Prefix Registration

When a Client is not pre-provisioned with an MNP (or, when the Client requires additional MNP delegations), it requests the MS to select MNPs on its behalf and set up the correct routing state. The DHCPv6 service [RFC8415] supports this requirement.

When a Client requires the MS to select MNPs, it sends an RS message with source set to a {TLA,XLA}-RND. If the Client requires only a single MNP delegation, it can then include a OMNI Node Identification sub-option plus an OMNI Neighbor Coordination sub-option with Preflen

set to the length of the desired MNP. If the Client requires multiple MNP delegations and/or more complex DHCPv6 services, it instead includes a DHCPv6 Message sub-option containing a Client Identifier, one or more IA_PD options and a Rapid Commit option then sets the 'msg-type' field to "Solicit", and includes a 3 octet 'transaction-id'. The Client then sets the RS destination to link-scoped All-Routers multicast and sends the message using OAL encapsulation and fragmentation if necessary as discussed above.

When the Hub Proxy/Server receives the RS message, it performs OAL reassembly if necessary. Next, if the RS source is a {TLA,XLA}-RND and/or the OMNI option includes a DHCPv6 message sub-option, the Hub Proxy/Server acts as a "Proxy DHCPv6 Client" in a message exchange with the locally-resident DHCPv6 server. If the RS did not contain a DHCPv6 message sub-option, the Hub Proxy/Server generates a DHCPv6 Solicit message on behalf of the Client using an IA_PD option with the prefix length set to the OMNI Neighbor Coordination header Preflen value and with a Client Identifier formed from the OMNI option Node Identification sub-option; otherwise, the Hub Proxy/Server uses the DHCPv6 Solicit message contained in the OMNI option. The Hub Proxy/Server then sends the DHCPv6 message to the DHCPv6 Server, which delegates MNPs and returns a DHCPv6 Reply message with PD parameters. (If the Hub Proxy/Server wishes to defer creation of Client state until the DHCPv6 Reply is received, it can instead act as a Lightweight DHCPv6 Relay Agent per [RFC6221] by encapsulating the DHCPv6 message in a Relay-forward/reply exchange with Relay Message and Interface ID options. In the process, the Hub Proxy/Server packs any state information needed to return an RA to the Client in the Relay-forward Interface ID option so that the information will be echoed back in the Relay-reply.)

When the Hub Proxy/Server receives the DHCPv6 Reply, it creates XLA-MNPs based on the delegated MNPs and creates OMNI interface XLA-MNP forwarding table entries (i.e., to prompt the dynamic routing protocol). The Hub Proxy/Server then sends an RA back to the FHS Proxy/Server with the DHCPv6 Reply message included in an OMNI DHCPv6 message sub-option. The Hub Proxy/Server sets the RA destination address to the RS source address, sets the RA source address to its own ULA, performs OAL encapsulation and fragmentation, performs L2 encapsulation and sends the RA to the Client via the FHS Proxy/Server as discussed above.

When the FHS Proxy/Server receives the RA, it changes the RA destination address to the ULA-MNP for the Client within its own ULA subnet prefix then forwards the RA to the Client. When the Client receives the RA, it reassembles and discards the OAL encapsulation then creates a default route, assigns Subnet Router Anycast addresses and uses the RA destination address or DHCPv6-delegated MNP to

automatically configure its primary ULA-MNP. The Client will then use these primary MNP-based addresses as the source address of any IPv6 ND messages it sends as long as it retains ownership of the MNP.

Note: when the Hub Proxy/Server is also the FHS Proxy/Server, it forwards the RA message directly to the Client with the destination set to the Client's ULA-MNP (i.e., instead of forwarding via another Proxy/Server).

15.4. OMNI Link Extension

Clients can provide an OMNI link ingress point for other nodes on their (downstream) ENETs that also act as Clients. When Client A has already coordinated with an (upstream) ANET/INET Proxy/Server, Client B on an ENET serviced by Client A can send OAL-encapsulated RS messages with addresses set the same as specified in Section 15.2. When Client A receives the RS message, it infers from the OAL encapsulation that Client B is seeking to establish itself as a Client instead of just a simple ENET Host.

Client A then returns an RA message the same as a Proxy/Server would do as specified in Section 15.2 except that it instead uses its own ULA-MNP as the RA and OAL source addresses and performs (recursive) DHCPv6 Prefix Delegation. The MNP delegation in the RA message must be a sub-MNP from the MNP delegated to Client A. For example, if Client A receives the MNP 2001:db8:1000::/48 it can provide a sub-delegation such as 2001:db8:1000:2000::/56 to Client B. Client B can in turn sub-delegate 2001:db8:1000:2000::/56 to its own ENET(s), where there may be a further prospective Client C that would in turn request OMNI link services via Client B.

To support this Client-to-Client chaining, Clients send IPv6 ND messages addressed to the OMNI link anycast address via their ANET/INET (i.e., upstream) interfaces, but advertise the OMNI link anycast address into their ENET (i.e., downstream) networks where there may be further prospective Clients wishing to join the chain. The ENET of the upstream Client is therefore seen as an ANET by downstream Clients, and the upstream Client is seen as a Proxy/Server by downstream Clients.

16. Secure Redirection

If the underlay network link model is multiple access, the FHS Proxy/Server is responsible for assuring that address duplication cannot corrupt the neighbor caches of other nodes on the link. When the Client sends an RS message on a multiple access underlay network, the Proxy/Server verifies that the Client is authorized to use the address and responds with an RA (or forwards the RS to the Hub) only

if the Client is authorized.

After verifying Client authorization and returning an RA, the Proxy/Server MAY return IPv6 ND Redirect messages to direct Clients located on the same underlay network to exchange packets directly without transiting the Proxy/Server. In that case, the Clients can exchange packets according to their unicast L2 addresses discovered from the Redirect message instead of using the dogleg path through the Proxy/Server. In some underlay networks, however, such direct communications may be undesirable and continued use of the dogleg path through the Proxy/Server may provide better performance. In that case, the Proxy/Server can refrain from sending Redirects, and/or Clients can ignore them.

17. Proxy/Server Resilience

*NETs SHOULD deploy Proxy/Servers in Virtual Router Redundancy Protocol (VRRP) [RFC5798] configurations so that service continuity is maintained even if one or more Proxy/Servers fail. Using VRRP, the Client is unaware which of the (redundant) FHS Proxy/Servers is currently providing service, and any service discontinuity will be limited to the failover time supported by VRRP. Widely deployed public domain implementations of VRRP are available.

Proxy/Servers SHOULD use high availability clustering services so that multiple redundant systems can provide coordinated response to failures. As with VRRP, widely deployed public domain implementations of high availability clustering services are available. Note that special-purpose and expensive dedicated hardware is not necessary, and public domain implementations can be used even between lightweight virtual machines in cloud deployments.

18. Detecting and Responding to Proxy/Server Failures

In environments where fast recovery from Proxy/Server failure is required, FHS Proxy/Servers SHOULD use proactive Neighbor Unreachability Detection (NUD) in a manner that parallels Bidirectional Forwarding Detection (BFD) [RFC5880] to track Hub Proxy/Server reachability. FHS Proxy/Servers can then quickly detect and react to failures so that cached information is re-established through alternate paths. Proactive NUD control messaging is carried only over well-connected ground domain networks (i.e., and not low-end links such as aeronautical radios) and can therefore be tuned for rapid response.

FHS Proxy/Servers perform proactive NUD for Hub Proxy/Servers for which there are currently active Clients. If a Hub Proxy/Server fails, the FHS Proxy/Server can quickly inform Clients of the outage

by sending multicast RA messages. The FHS Proxy/Server sends RA messages to Clients with source set to the ULA of the Hub, with destination address set to All-Nodes multicast (ff02::1) [RFC4291] and with Router Lifetime set to 0.

The FHS Proxy/Server SHOULD send MAX_FINAL_RTR_ADVERTISEMENTS RA messages separated by small delays [RFC4861]. Any Clients that have been using the (now defunct) Hub Proxy/Server will receive the RA messages.

19. Transition Considerations

When a Client connects to an *NET link for the first time, it sends an RS message with an OMNI option. If the first hop router recognizes the option, it responds according to the appropriate FHS/Hub Proxy/Server role resulting in an RA message with an OMNI option returned to the Client. The Client then engages this FHS Proxy/Server according to the OMNI link model specified above. If the first hop router is a legacy IPv6 router, however, it instead returns an RA message with no OMNI option and with a non-OMNI unicast source LLA as specified in [RFC4861]. In that case, the Client engages the *NET according to the legacy IPv6 link model and without the OMNI extensions specified in this document.

If the *NET link model is multiple access, there must be assurance that address duplication cannot corrupt the neighbor caches of other nodes on the link. When the Client sends an RS message on a multiple access *NET link with an OMNI option, first hop routers that recognize the option ensure that the Client is authorized to use the address and return an RA with a non-zero Router Lifetime only if the Client is authorized. First hop routers that do not recognize the OMNI option instead return an RA that makes no statement about the Client's authorization to use the source address. In that case, the Client should perform Duplicate Address Detection to ensure that it does not interfere with other nodes on the link.

An alternative approach for multiple access *NET links to ensure isolation for Client-Proxy/Server communications is through link-layer address mappings as discussed in Appendix D. This arrangement imparts a (virtual) point-to-point link model over the (physical) multiple access link.

20. OMNI Interfaces on Open Internetworks

Client OMNI interfaces configured over IPv6-enabled underlay interfaces on an open Internetwork without an OMNI-aware first-hop router receive IPv6 RA messages with no OMNI options, while OMNI interfaces configured over IPv4-only underlay interfaces receive no IPv6 RA messages at all (but may receive IPv4 RA messages [RFC1256]). Client OMNI interfaces that receive RA messages with OMNI options configure addresses, on-link prefixes, etc. on the underlay interface that received the RA according to standard IPv6 ND and address resolution conventions [RFC4861] [RFC4862]. Client OMNI interfaces configured over IPv4-only underlay interfaces configure IPv4 address information on the underlay interfaces using mechanisms such as DHCPv4 [RFC2131].

Client OMNI interfaces configured over underlay interfaces connected to open Internetworks can apply security services such as VPNs to connect to a Proxy/Server, or can establish a direct link to the Proxy/Server through some other means (see Section 4). In environments where an explicit VPN or direct link may be impractical or undesirable, Client OMNI interfaces can instead send IPv6 ND messages with OMNI options that include authentication signatures.

OMNI interfaces use UDP/IP as L2 encapsulation headers for transmission over open Internetworks with UDP service port number 8060 (see: Section 25.13 and Section 3.6 of [I-D.templin-6man-aero]) for both IPv4 and IPv6 underlay interfaces. The OMNI interface submits original IP packets for OAL encapsulation, then encapsulates the resulting OAL fragments in UDP/IP L2 headers to form carrier packets. (The first four bits following the UDP header determine whether the OAL headers are uncompressed/compressed as discussed in Section 6.4.) The OMNI interface sets the UDP length to the encapsulated OAL fragment length and sets the IP length to an appropriate value at least as large as the UDP datagram.

For Client-Proxy/Server (e.g., "Vehicle-to-Infrastructure (V2I)") neighbor exchanges, the source must include an OMNI option with an authentication sub-option in all IPv6 ND messages. The source can apply HIP security services per [RFC7401] using the IPv6 ND message OMNI option as a "shipping container" to convey an authentication signature in a (unidirectional) HIP "Notify" message. For Client-Client (e.g., "Vehicle-to-Vehicle (V2V)") neighbor exchanges, two Clients can exchange HIP "Initiator/Responder" messages coded in OMNI options of multiple IPv6 NS/NA messages for mutual authentication according to the HIP protocol. (Note: a simple Hashed Message Authentication Code (HMAC) such as specified in [RFC4380] or the QUIC-TLS connection-oriented service [RFC9000] can be used as an alternate authentication service in some environments.)

When an OMNI interface includes an authentication sub-option, it must appear as the first sub-option of the first OMNI option in the IPv6 ND message which must appear immediately following the IPv6 ND message header. When an OMNI interface prepares a HIP message sub-option, it includes its own (H)HIT as the Sender's HIT and the neighbor's (H)HIT if known as the Receiver's HIT (otherwise 0). If (H)HITs are not available within the OMNI operational environment, the source can instead include other IPv6 address types instead of (H)HITs as long as the Sender and Receiver have some way to associate information embedded in the IPv6 address with the neighbor; such information could include a node identifier, vehicle identifier, MAC address, etc.

Before calculating the authentication signature, the source includes any other necessary sub-options (such as Interface Attributes and Origin Indication) and sets both the IPv6 ND message Checksum and authentication signature fields to 0. The source then calculates the authentication signature over the full length of the IPv6 ND message beginning with a pseudo-header of the IPv6 header (i.e., the same as specified in [RFC4443]) and extending over the length of the message. (If the IPv6 ND message is part of an OAL super-packet, the source instead calculates the authentication signature over the remainder of the super-packet.) The source next writes the authentication signature into the sub-option signature field and forwards the message.

After establishing a VPN or preparing for UDP/IP encapsulation, OMNI interfaces send RS/RA messages for Client-Proxy/Server coordination (see: Section 15) and NS/NA messages for route optimization, window synchronization and mobility management (see: [I-D.templin-6man-aero]). These control plane messages must be authenticated while other control and data plane messages are delivered the same as for ordinary best-effort traffic with source address and/or Identification window-based data origin verification. Upper layer protocol sessions over OMNI interfaces that connect over open Internetworks without an explicit VPN should therefore employ transport- or higher-layer security to ensure authentication, integrity and/or confidentiality.

Clients should avoid using INET Proxy/Servers as general-purpose routers for steady streams of carrier packets that do not require authentication. Clients should instead perform route optimization to coordinate with other INET nodes that can provide forwarding services (or preferably coordinate directly with peer Clients directly) instead of burdening the Proxy/Server. Procedures for coordinating with peer Clients and discovering INET nodes that can provide better forwarding services are discussed in [I-D.templin-6man-aero].

Clients that attempt to contact peers over INET underlay interfaces often encounter NATs in the path. OMNI interfaces accommodate NAT traversal using UDP/IP encapsulation and the mechanisms discussed in [I-D.templin-6man-aero]. FHS Proxy/Servers include Origin Indications in RA messages to allow Clients to detect the presence of NATs.

Note: Following the initial IPv6 ND message exchange, OMNI interfaces configured over INET underlay interfaces maintain neighbor relationships by transmitting periodic IPv6 ND messages with OMNI options that include HIP "Update" and/or "Notify" messages. When HMAC authentication is used instead of HIP, the Client and Proxy/Server exchange all IPv6 ND messages with HMAC signatures included based on a shared-secret. When QUIC-TLS connections are used, the Client and Proxy/Server observe QUIC-TLS conventions [RFC9000][RFC9001].

Note: OMNI interfaces configured over INET underlay interfaces should employ the Identification window synchronization mechanisms specified in Section 6.6 in order to exclude spurious carrier packets that might otherwise clutter the reassembly cache. This is especially important in environments where carrier packet spoofing and/or corruption is a threat.

Note: NATs may be present on the path from a Client to its FHS Proxy/Server, but never on the path from the FHS Proxy/Server to the Hub where only INET and/or spanning tree hops occur. Therefore, the FHS Proxy/Server does not communicate Client origin information to the Hub where it would serve no purpose.

21. Time-Varying MNPs

In some use cases, it is desirable, beneficial and efficient for the Client to receive a constant MNP that travels with the Client wherever it moves. For example, this would allow air traffic controllers to easily track aircraft, etc. In other cases, however (e.g., intelligent transportation systems), the Client may be willing to sacrifice a modicum of efficiency in order to have time-varying MNPs that can be changed every so often to defeat adversarial tracking.

The prefix delegation services discussed in Section 15.3 allows Clients that desire time-varying MNPs to obtain short-lived prefixes to send RS messages with a {TLA,XLA}-RND source address and/or with an OMNI option with DHCPv6 Option sub-options. The Client would then be obligated to renumber its internal networks whenever its MNP (and therefore also its OMNI address) changes. This should not present a challenge for Clients with automated network renumbering services, but may disrupt persistent sessions that would prefer to use a constant address.

22. (H)HITs and Temporary ULA (TLA)s

Clients that generate (H)HITs but do not have pre-assigned MNPs can request MNP delegations by issuing IPv6 ND messages that use the (H)HIT instead of a TLA. For example, when a Client creates an RS message it can set the source to a (H)HIT and destination to link-scoped All-Routers multicast. The IPv6 ND message includes an OMNI option with a HIP message sub-option, and need not include a Node Identification sub-option if the Client's (H)HIT appears in the HIP message. The Client then encapsulates the message in an IPv6 header with the (H)HIT as the source address. The Client then sends the message as specified in Section 15.2.

When the Hub Proxy/Server receives the RS message, it notes that the source was a (H)HIT, then invokes the DHCPv6 protocol to request an MNP prefix delegation while using the (H)HIT (in the form of a DUID) as the Client Identifier. The Hub Proxy/Server then prepares an RA message with source address set to its own ULA and destination set to the source of the RS message. The Hub Proxy/Server next includes an OMNI option with a HIP message sub-option and any DHCPv6 prefix delegation parameters. The Proxy/Server finally encapsulates the RA in an OAL header with source address set to its own ULA and destination set to the RS OAL source address, then returns the encapsulated RA to the Client either directly or by way of the FHS Proxy/Server as a proxy.

Clients can also use (H)HITs and/or TLAs for direct Client-to-Client communications outside the context of any OMNI link supporting infrastructure. When two Clients encounter one another they can use their (H)HITs and/or TLAs as original IPv6 packet source and destination addresses to support direct communications. Clients can also inject their (H)HITs and/or TLAs into an IPv6 multihop routing protocol to enable multihop communications as discussed in Section 15.2. Clients can further exchange other IPv6 ND messages using their (H)HITs and/or TLAs as source and destination addresses.

Lastly, when Clients are within the coverage range of OMNI link infrastructure a case could be made for injecting (H)HITs and/or TLAs into the global MS routing system. For example, when the Client sends an RS to an FHS Proxy/Server it could include a request to inject the (H)HIT / TLA into the routing system instead of requesting an MNP prefix delegation. This would potentially enable OMNI link-wide communications using only (H)HITs or TLAs, and not MNPs. This document notes the opportunity, but makes no recommendation.

23. Address Selection

Clients assign LLAs to the OMNI interface, but do not use LLAs as IPv6 ND message source/destination addresses nor for addressing ordinary original IP packets exchanged with OMNI link neighbors.

Clients use ULA-MNPs as source/destination IPv6 addresses in the encapsulation headers of OAL packets and use XLA-MNPs as the IPv6 source addresses of the IPv6 ND messages themselves. Clients use TLAs when an MNP is not available, or as source/destination IPv6 addresses for communications within a MANET/VANET local area. Clients can also use (H)HITs instead of ULAs for local communications when operation outside the context of a specific ULA domain and/or source address attestation is necessary.

Clients use MNP-based GUAs as original IP packet source and destination addresses for communications with Internet destinations when they are within range of OMNI link supporting infrastructure that can inject the MNP into the routing system. Clients can also use MNP-based GUAs within multihop routing regions that are currently disconnected from infrastructure as long as the corresponding ULA-MNPs have been injected into the routing system.

Clients use anycast GUAs as OAL and/or L2 encapsulation destination addresses for RS messages used to discover the nearest FHS Proxy/Server. When the Proxy/Server returns a solicited RA, it must also use the same anycast address as the RA OAL/L2 encapsulation source in order to successfully traverse any NATs in the path. The Client should then immediately transition to using the FHS Proxy/Server's discovered unicast OAL/L2 address as the destination in order to minimize dependence on the Proxy/Server's use of an anycast source address.

24. Error Messages

An OAL destination or intermediate node may need to return ICMPv6-like error messages (e.g., Destination Unreachable, Packet Too Big, Time Exceeded, etc.) [RFC4443] to an OAL source. Since ICMPv6 error messages do not themselves include authentication codes, OAL nodes can return error messages as an OMNI ICMPv6 Error sub-option in a secured IPv6 ND uNA message.

25. IANA Considerations

The following IANA actions are requested in accordance with [RFC8126] and [RFC8726]:

25.1. "Protocol Numbers" Registry

The IANA is instructed to allocate an Internet Protocol number TBD1 from the 'protocol numbers' registry for the Overlay Multilink Network Interface (OMNI) protocol. Guidance is found in [RFC5237] (registration procedure is IESG Approval or Standards Action).

25.2. "IEEE 802 Numbers" Registry

During final publication stages, the IESG will be requested to procure an IEEE EtherType value TBD2 for OMNI according to the statement found at <https://www.ietf.org/about/groups/iesg/statements/ethertypes/>.

Following this procurement, the IANA is instructed to register the value TBD2 in the 'ieee-802-numbers' registry for Overlay Multilink Network Interface (OMNI) encapsulation on Ethernet networks. Guidance is found in [RFC7042] (registration procedure is Expert Review).

25.3. "IPv4 Special-Purpose Address" Registry

The IANA is instructed to assign TBD3/N as an "OMNI IPv4 anycast" address/prefix in the "IPv4 Special-Purpose Address" registry in a similar fashion as for [RFC3068]. The IANA is requested to work with the authors to obtain a TBD3/N public IPv4 prefix, whether through an RIR allocation, a delegation from IANA's "IPv4 Recovered Address Space" registry or through an unspecified third party donation.

25.4. "IPv6 Neighbor Discovery Option Formats" Registry

The IANA is instructed to allocate an official Type number TBD4 from the "IPv6 Neighbor Discovery Option Formats" registry for the OMNI option (registration procedure is RFC required). Implementations set Type to 253 as an interim value [RFC4727].

25.5. "Ethernet Numbers" Registry

The IANA is instructed to allocate one Ethernet unicast address TBD5 (suggested value '00-52-14') in the 'ethernet-numbers' registry under "IANA Unicast 48-bit MAC Addresses" (registration procedure is Expert Review). The registration should appear as follows:

Addresses	Usage	Reference
-----	-----	-----
00-52-14	Overlay Multilink Network (OMNI) Interface	[RFCXXXX]

Figure 35: IANA Unicast 48-bit MAC Addresses

25.6. "ICMPv6 Code Fields: Type 2 - Packet Too Big" Registry

The IANA is instructed to assign two new Code values in the "ICMPv6 Code Fields: Type 2 - Packet Too Big" registry (registration procedure is Standards Action or IESG Approval). The registry should appear as follows:

Code	Name	Reference
---	----	-----
0	PTB Hard Error	[RFC4443]
1	PTB Soft Error (loss)	[RFCXXXX]
2	PTB Soft Error (no loss)	[RFCXXXX]

Figure 36: ICMPv6 Code Fields: Type 2 - Packet Too Big Values

(Note: this registry also to be used to define values for setting the "unused" field of ICMPv4 "Destination Unreachable - Fragmentation Needed" messages.)

25.7. "OMNI Option Sub-Type Values" (New Registry)

The OMNI option defines a 5-bit Sub-Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI Option Sub-Type Values". Initial values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
-----	-----	-----
0	Pad1	[RFCXXXX]
1	PadN	[RFCXXXX]
2	Neighbor Coordination	[RFCXXXX]
3	Interface Attributes	[RFCXXXX]
4	Multilink Forwarding Params	[RFCXXXX]
5	Traffic Selector	[RFCXXXX]
6	Geo Coordinates	[RFCXXXX]
7	DHCPv6 Message	[RFCXXXX]
8	HIP Message	[RFCXXXX]
9	PIM-SM Message	[RFCXXXX]
10	Fragmentation Report	[RFCXXXX]
11	Node Identification	[RFCXXXX]
12	ICMPv6 Error	[RFCXXXX]
13	QUIC-TLS Message	[RFCXXXX]
14	Proxy/Server Departure	[RFCXXXX]
15-29	Unassigned	
30	Sub-Type Extension	[RFCXXXX]
31	Reserved by IANA	[RFCXXXX]

Figure 37: OMNI Option Sub-Type Values

25.8. "OMNI Geo Coordinates Type Values" (New Registry)

The OMNI Geo Coordinates sub-option (see: Section 12.2.7) contains an 8-bit Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI Geo Coordinates Type Values". Initial values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
-----	-----	-----
0	NULL	[RFCXXXX]
1-252	Unassigned	[RFCXXXX]
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 38: OMNI Geo Coordinates Type

25.9. "OMNI Node Identification ID-Type Values" (New Registry)

The OMNI Node Identification sub-option (see: Section 12.2.12) contains an 8-bit ID-Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI Node Identification ID-Type Values". Initial values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
-----	-----	-----
0	UUID	[RFCXXXX]
1	HIT	[RFCXXXX]
2	HHIT	[RFCXXXX]
3	Network Access Identifier	[RFCXXXX]
4	FQDN	[RFCXXXX]
5	IPv6 Address	[RFCXXXX]
6-252	Unassigned	[RFCXXXX]
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 39: OMNI Node Identification ID-Type Values

25.10. "OMNI Option Sub-Type Extension Values" (New Registry)

The OMNI option defines an 8-bit Extension-Type field for Sub-Type 30 (Sub-Type Extension), for which IANA is instructed to create and maintain a new registry entitled "OMNI Option Sub-Type Extension Values". Initial values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
-----	-----	-----
0	RFC4380 UDP/IP Header Option	[RFCXXXX]
1	RFC6081 UDP/IP Trailer Option	[RFCXXXX]
2-252	Unassigned	
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 40: OMNI Option Sub-Type Extension Values

25.11. "OMNI RFC4380 UDP/IP Header Option" (New Registry)

The OMNI Sub-Type Extension "RFC4380 UDP/IP Header Option" defines an 8-bit Header Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI RFC4380 UDP/IP Header Option". Initial registry values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
-----	-----	-----
0	Origin Indication (IPv4)	[RFC4380]
1	Authentication Encapsulation	[RFC4380]
2	Origin Indication (IPv6)	[RFCXXXX]
3-252	Unassigned	
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 41: OMNI RFC4380 UDP/IP Header Option

25.12. "OMNI RFC6081 UDP/IP Trailer Option" (New Registry)

The OMNI Sub-Type Extension for "RFC6081 UDP/IP Trailer Option" defines an 8-bit Trailer Type field, for which IANA is instructed to create and maintain a new registry entitled "OMNI RFC6081 UDP/IP Trailer Option". Initial registry values are given below (registration procedure is RFC required):

Value	Sub-Type name	Reference
-----	-----	-----
0	Unassigned	
1	Nonce	[RFC6081]
2	Unassigned	
3	Alternate Address (IPv4)	[RFC6081]
4	Neighbor Discovery Option	[RFC6081]
5	Random Port	[RFC6081]
6	Alternate Address (IPv6)	[RFCXXXX]
7-252	Unassigned	
253-254	Reserved for Experimentation	[RFCXXXX]
255	Reserved by IANA	[RFCXXXX]

Figure 42: OMNI RFC6081 Trailer Option

25.13. Additional Considerations

The IANA has assigned the UDP port number "8060" for an earlier experimental version of AERO [RFC6706]. This document reclaims the UDP port number "8060" for 'aero' as the service port for UDP/IP encapsulation. (Note that, although [RFC6706] is not widely implemented or deployed, any messages coded to that specification can be easily distinguished and ignored since they include an invalid ICMPv6 message type number '0'.) The IANA is therefore instructed to update the reference for UDP port number "8060" from "RFC6706" to "RFCXXXX" (i.e., this document) while retaining the existing name 'aero'.

The IANA has assigned a 4 octet Private Enterprise Number (PEN) code "45282" in the "enterprise-numbers" registry. This document is the normative reference for using this code in DHCP Unique IDentifiers based on Enterprise Numbers ("DUID-EN for OMNI Interfaces") (see: Section 11). The IANA is therefore instructed to change the enterprise designation for PEN code "45282" from "LinkUp Networks" to "Overlay Multilink Network Interface (OMNI)".

The IANA has assigned the ifType code "301 - omni - Overlay Multilink Network Interface (OMNI)" in accordance with Section 6 of [RFC8892]. The registration appears under the IANA "Structure of Management Information (SMI) Numbers (MIB Module Registrations) - Interface Types (ifType)" registry.

No further IANA actions are required.

26. Security Considerations

Security considerations for IPv4 [RFC0791], IPv6 [RFC8200] and IPv6 Neighbor Discovery [RFC4861] apply. OMNI interface IPv6 ND messages SHOULD include Nonce and Timestamp options [RFC3971] when transaction confirmation and/or time synchronization is needed. (Note however that when OAL encapsulation is used the (echoed) OAL Identification value can provide sufficient transaction confirmation.)

OMNI interfaces configured over secured ANET/ENET interfaces inherit the physical and/or link-layer security properties (i.e., "protected spectrum") of the connected networks. OMNI interfaces configured over open INET interfaces can use symmetric securing services such as VPNs or can by some other means establish a direct link. When a VPN or direct link may be impractical or undesirable, however, the security services specified in [RFC7401], [RFC4380] or [RFC9000] can be employed. While the OMNI link protects control plane messaging, applications must still employ end-to-end transport- or higher-layer security services to protect the data plane.

Strong network layer security for control plane messages and forwarding path integrity for data plane messages between Proxy/Servers MUST be supported. In one example, the AERO service [I-D.templin-6man-aero] constructs an SRT spanning tree with Proxy/Servers as leaf nodes and secures the spanning tree links with network layer security mechanisms such as IPsec [RFC4301] or WireGuard [WG]. Secured control plane messages are then constrained to travel only over the secured spanning tree paths and are therefore protected from attack or eavesdropping. Other control and data plane messages can travel over route optimized paths that do not strictly follow the secured spanning tree, therefore end-to-end sessions should employ transport- or higher-layer security services. Additionally, the OAL Identification value can provide a first level of data origin authentication to mitigate off-path spoofing in some environments.

Identity-based key verification infrastructure services such as iPSK may be necessary for verifying the identities claimed by Clients. This requirement should be harmonized with the manner in which (H)HITs are attested in a given operational environment.

Security considerations for specific access network interface types are covered under the corresponding IP-over-(foo) specification (e.g., [RFC2464], [RFC2492], etc.).

Security considerations for IPv6 fragmentation and reassembly are discussed in Section 6.12. In environments where spoofing is considered a threat, OMNI nodes SHOULD employ Identification window synchronization and OAL destinations SHOULD configure an (end-system-based) firewall.

27. Implementation Status

AERO/OMNI Release-3.2 was tagged on March 30, 2021, and is undergoing internal testing. Additional internal releases expected within the coming months, with first public release expected end of 1H2021.

Many AERO/OMNI functions are implemented and undergoing final integration. OAL fragmentation/reassembly buffer management code has been cleared for public release.

28. Document Updates

This document does not itself update other RFCs, but suggests that the following could be updated through future IETF initiatives:

- * [RFC1191]
- * [RFC2675]
- * [RFC4291]
- * [RFC4443]
- * [RFC8201]

Updates can be through, e.g., standards action, the errata process, etc. as appropriate.

29. Acknowledgements

The first version of this document was prepared per the consensus decision at the 7th Conference of the International Civil Aviation Organization (ICAO) Working Group-I Mobility Subgroup on March 22, 2019. Consensus to take the document forward to the IETF was reached at the 9th Conference of the Mobility Subgroup on November 22, 2019. Attendees and contributors included: Guray Acar, Danny Bharj, Francois D'Humieres, Pavel Drasil, Nikos Fistas, Giovanni Garofolo, Bernhard Haindl, Vaughn Maiolla, Tom McParland, Victor Moreno, Madhu

Niraula, Brent Phillips, Liviu Popescu, Jacky Pouzet, Aloke Roy, Greg Saccone, Robert Segers, Michal Skorepa, Michel Solery, Stephane Tamalet, Fred Templin, Jean-Marc Vacher, Bela Varkonyi, Tony Whyman, Fryderyk Wrobel and Dongsong Zeng.

The following individuals are acknowledged for their useful comments: Amanda Baber, Stuart Card, Donald Eastlake, Adrian Farrel, Michael Matyas, Robert Moskowitz, Madhu Niraula, Greg Saccone, Stephane Tamalet, Eliot Lear, Eduard Vasilenko, Eric Vyncke. Pavel Drasil, Zdenek Jaron and Michal Skorepa are especially recognized for their many helpful ideas and suggestions. Akash Agarwal, Madhuri Madhava Badgandi, Sean Dickson, Don Dillenburg, Joe Dudkowski, Vijayasarathy Rajagopalan, Ron Sackman, Bhargava Raman Sai Prakash and Katherine Tran are acknowledged for their hard work on the implementation and technical insights that led to improvements for the spec.

Discussions on the IETF 6man and atn mailing lists during the fall of 2020 suggested additional points to consider. The authors gratefully acknowledge the list members who contributed valuable insights through those discussions. Eric Vyncke and Erik Kline were the intarea ADs, while Bob Hinden and Ole Troan were the 6man WG chairs at the time the document was developed; they are all gratefully acknowledged for their many helpful insights. Many of the ideas in this document have further built on IETF experiences beginning in the 1990s, with insights from colleagues including Ron Bonica, Brian Carpenter, Ralph Droms, Christian Huitema, Thomas Narten, Dave Thaler, Joe Touch, Pascal Thubert, and many others who deserve recognition.

Early observations on IP fragmentation performance implications were noted in the 1986 Digital Equipment Corporation (DEC) "qe reset" investigation, where fragment bursts from NFS UDP traffic triggered hardware resets resulting in communication failures. Jeff Chase, Fred Glover and Chet Juzszczak of the Ultrix Engineering Group led the investigation, and determined that setting a smaller NFS mount block size reduced the amount of fragmentation and suppressed the resets. Early observations on L2 media MTU issues were noted in the 1988 DEC FDDI investigation, where Raj Jain, KK Ramakrishnan and Kathy Wilde represented architectural considerations for FDDI networking in general including FDDI/Ethernet bridging. Jeff Mogul (who led the IETF Path MTU Discovery working group) and other DEC colleagues who supported these early investigations are also acknowledged.

Throughout the 1990's and into the 2000's, many colleagues supported and encouraged continuation of the work. Beginning with the DEC Project Sequoia effort at the University of California, Berkeley, then moving to the DEC research lab offices in Palo Alto CA, then to Sterling Software at the NASA Ames Research Center, then to SRI in

Menlo Park, CA, then to Nokia in Mountain View, CA and finally to the Boeing Company in 2005 the work saw continuous advancement through the encouragement of many. Those who offered their support and encouragement are gratefully acknowledged.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the Boeing Information Technology (BIT) Mobility Vision Lab (MVL) program.

30. References

30.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.

- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <<https://www.rfc-editor.org/info/rfc4727>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <<https://www.rfc-editor.org/info/rfc6088>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.

30.2. Informative References

- [ATN] Maiolla, V., "The OMNI Interface - An IPv6 Air/Ground Interface for Civil Aviation, IETF Liaison Statement #1676, <https://datatracker.ietf.org/liaison/1676/>", 3 March 2020.
- [ATN-IPS] WG-I, ICAO., "ICAO Document 9896 (Manual on the Aeronautical Telecommunication Network (ATN) using Internet Protocol Suite (IPS) Standards and Protocol), Draft Edition 3 (work-in-progress)", 10 December 2020.
- [CKSUM] Stone, J., Greenwald, M., Partridge, C., and J. Hughes, "Performance of Checksums and CRC's Over Real Data, IEEE/ACM Transactions on Networking, Vol. 6, No. 5", October 1998.
- [CRC] Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI), IEEE Transactions on Communications", August 1990.
- [I-D.ietf-drip-rid]
Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID)", Work in Progress, Internet-Draft, draft-ietf-drip-rid-24, 24 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-drip-rid-24.txt>>.
- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-10, 12 September 2019, <<https://www.ietf.org/archive/id/draft-ietf-intarea-tunnels-10.txt>>.

[I-D.ietf-ipwave-vehicular-networking]

Jeong, J. (., "IPv6 Wireless Access in Vehicular Environments (IPWAVE): Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-ietf-ipwave-vehicular-networking-28, 30 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-ipwave-vehicular-networking-28.txt>>.

[I-D.templin-6man-aero]

Templin, F. L., "Automatic Extended Route Optimization (AERO)", Work in Progress, Internet-Draft, draft-templin-6man-aero-45, 22 April 2022, <<https://www.ietf.org/archive/id/draft-templin-6man-aero-45.txt>>.

[I-D.templin-6man-fragrep]

Templin, F. L., "IPv6 Fragment Retransmission and Path MTU Discovery Soft Errors", Work in Progress, Internet-Draft, draft-templin-6man-fragrep-07, 29 March 2022, <<https://www.ietf.org/archive/id/draft-templin-6man-fragrep-07.txt>>.

[I-D.templin-6man-lla-type]

Templin, F. L., "The IPv6 Link-Local Address Type Field", Work in Progress, Internet-Draft, draft-templin-6man-lla-type-02, 23 November 2020, <<https://www.ietf.org/archive/id/draft-templin-6man-lla-type-02.txt>>.

[I-D.templin-intarea-parcels]

Templin, F. L., "IP Parcels", Work in Progress, Internet-Draft, draft-templin-intarea-parcels-10, 29 March 2022, <<https://www.ietf.org/archive/id/draft-templin-intarea-parcels-10.txt>>.

[IPV4-GUA] Postel, J., "IPv4 Address Space Registry,

<https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml>", 14 December 2020.

[IPV6-GUA] Postel, J., "IPv6 Global Unicast Address Assignments,

<https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml>", 14 December 2020.

[RFC1035]

Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.

- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [RFC1146] Zweig, J. and C. Partridge, "TCP alternate checksum options", RFC 1146, DOI 10.17487/RFC1146, March 1990, <<https://www.rfc-editor.org/info/rfc1146>>.
- [RFC1149] Waitzman, D., "Standard for the transmission of IP datagrams on avian carriers", RFC 1149, DOI 10.17487/RFC1149, April 1990, <<https://www.rfc-editor.org/info/rfc1149>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC1256] Deering, S., Ed., "ICMP Router Discovery Messages", RFC 1256, DOI 10.17487/RFC1256, September 1991, <<https://www.rfc-editor.org/info/rfc1256>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<https://www.rfc-editor.org/info/rfc2464>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC2492] Armitage, G., Schuler, P., and M. Jork, "IPv6 over ATM Networks", RFC 2492, DOI 10.17487/RFC2492, January 1999, <<https://www.rfc-editor.org/info/rfc2492>>.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <<https://www.rfc-editor.org/info/rfc2675>>.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <<https://www.rfc-editor.org/info/rfc2863>>.

- [RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <<https://www.rfc-editor.org/info/rfc2923>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.
- [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, DOI 10.17487/RFC3056, February 2001, <<https://www.rfc-editor.org/info/rfc3056>>.
- [RFC3068] Huitema, C., "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, DOI 10.17487/RFC3068, June 2001, <<https://www.rfc-editor.org/info/rfc3068>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <<https://www.rfc-editor.org/info/rfc3330>>.
- [RFC3366] Fairhurst, G. and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", BCP 62, RFC 3366, DOI 10.17487/RFC3366, August 2002, <<https://www.rfc-editor.org/info/rfc3366>>.
- [RFC3684] Ogier, R., Templin, F., and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", RFC 3684, DOI 10.17487/RFC3684, February 2004, <<https://www.rfc-editor.org/info/rfc3684>>.
- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <<https://www.rfc-editor.org/info/rfc3692>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.

- [RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <<https://www.rfc-editor.org/info/rfc3819>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <<https://www.rfc-editor.org/info/rfc4380>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<https://www.rfc-editor.org/info/rfc4429>>.
- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <<https://www.rfc-editor.org/info/rfc4605>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <<https://www.rfc-editor.org/info/rfc4963>>.

- [RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <<https://www.rfc-editor.org/info/rfc5213>>.
- [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <<https://www.rfc-editor.org/info/rfc5214>>.
- [RFC5237] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, <<https://www.rfc-editor.org/info/rfc5237>>.
- [RFC5558] Templin, F., Ed., "Virtual Enterprise Traversal (VET)", RFC 5558, DOI 10.17487/RFC5558, February 2010, <<https://www.rfc-editor.org/info/rfc5558>>.
- [RFC5614] Ogier, R. and P. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding", RFC 5614, DOI 10.17487/RFC5614, August 2009, <<https://www.rfc-editor.org/info/rfc5614>>.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<https://www.rfc-editor.org/info/rfc5798>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5889] Baccelli, E., Ed. and M. Townsley, Ed., "IP Addressing Model in Ad Hoc Networks", RFC 5889, DOI 10.17487/RFC5889, September 2010, <<https://www.rfc-editor.org/info/rfc5889>>.
- [RFC5942] Singh, H., Beebee, W., and E. Nordmark, "IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes", RFC 5942, DOI 10.17487/RFC5942, July 2010, <<https://www.rfc-editor.org/info/rfc5942>>.
- [RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, DOI 10.17487/RFC6081, January 2011, <<https://www.rfc-editor.org/info/rfc6081>>.

- [RFC6214] Carpenter, B. and R. Hinden, "Adaptation of RFC 1149 for IPv6", RFC 6214, DOI 10.17487/RFC6214, April 2011, <<https://www.rfc-editor.org/info/rfc6214>>.
- [RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <<https://www.rfc-editor.org/info/rfc6221>>.
- [RFC6247] Eggert, L., "Moving the Undeployed TCP Extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to Historic Status", RFC 6247, DOI 10.17487/RFC6247, May 2011, <<https://www.rfc-editor.org/info/rfc6247>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC6543] Gundavelli, S., "Reserved IPv6 Interface Identifier for Proxy Mobile IPv6", RFC 6543, DOI 10.17487/RFC6543, May 2012, <<https://www.rfc-editor.org/info/rfc6543>>.
- [RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <<https://www.rfc-editor.org/info/rfc6706>>.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <<https://www.rfc-editor.org/info/rfc6935>>.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <<https://www.rfc-editor.org/info/rfc6980>>.
- [RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <<https://www.rfc-editor.org/info/rfc7042>>.

- [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <<https://www.rfc-editor.org/info/rfc7094>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <<https://www.rfc-editor.org/info/rfc7401>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<https://www.rfc-editor.org/info/rfc7421>>.
- [RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <<https://www.rfc-editor.org/info/rfc7542>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<https://www.rfc-editor.org/info/rfc7739>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC7847] Melia, T., Ed. and S. Gundavelli, Ed., "Logical-Interface Support for IP Hosts with Multi-Access Support", RFC 7847, DOI 10.17487/RFC7847, May 2016, <<https://www.rfc-editor.org/info/rfc7847>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8726] Farrel, A., "How Requests for IANA Action Will Be Handled on the Independent Stream", RFC 8726, DOI 10.17487/RFC8726, November 2020, <<https://www.rfc-editor.org/info/rfc8726>>.
- [RFC8892] Thaler, D. and D. Romascanu, "Guidelines and Registration Procedures for Interface Types and Tunnel Types", RFC 8892, DOI 10.17487/RFC8892, August 2020, <<https://www.rfc-editor.org/info/rfc8892>>.
- [RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <<https://www.rfc-editor.org/info/rfc8899>>.
- [RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.
- [RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <<https://www.rfc-editor.org/info/rfc8981>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <<https://www.rfc-editor.org/info/rfc9001>>.
- [RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <<https://www.rfc-editor.org/info/rfc9002>>.
- [WG] WireGuard, W., "WireGuard, Fast, Modern, Secure VPN Tunnel, <https://wireguard.com/>", 7 March 2022.

Appendix A. OAL Checksum Algorithm

The OAL Checksum Algorithm adopts the 8-bit Fletcher algorithm specified in Appendix I of [RFC1146] as also analyzed in [CKSUM]. [RFC6247] declared [RFC1146] historic for the reason that the algorithms had never seen widespread use with TCP, however this document adopts the 8-bit Fletcher algorithm for a different purpose. Quoting from Appendix I of [RFC1146], the OAL Checksum Algorithm proceeds as follows:

"The 8-bit Fletcher Checksum Algorithm is calculated over a sequence of data octets (call them $D[1]$ through $D[N]$) by maintaining 2 unsigned 1's-complement 8-bit accumulators A and B whose contents are initially zero, and performing the following loop where i ranges from 1 to N:

$$A := A + D[i]$$
$$B := B + A$$

It can be shown that at the end of the loop A will contain the 8-bit 1's complement sum of all octets in the datagram, and that B will contain $(N)D[1] + (N-1)D[2] + \dots + D[N]$."

To calculate the OAL checksum, the above algorithm is applied over the N-octet concatenation of the OAL pseudo-header and the encapsulated IP packet or packets. Specifically, the algorithm is first applied over the 40 octets of the OAL pseudo-header as data octets $D[1]$ through $D[40]$, then continues over the entire length of the original IP packet(s) as data octets $D[41]$ through $D[N]$.

Appendix B. IPv6 ND Message Authentication and Integrity

OMNI interface IPv6 ND messages are subject to authentication and integrity checks at multiple levels. When an OMNI interface sends an IPv6 ND message over an INET interface, it includes an authentication sub-option with a valid signature but does not include an IPv6 ND message checksum. The OMNI interface that receives the message verifies the OAL checksum as a first-level integrity check, then verifies the authentication signature (while ignoring the IPv6 ND message checksum) to ensure IPv6 ND message authentication and integrity.

When an OMNI interface sends an IPv6 ND message over an underlay interface connected to a secured network, it omits the authentication sub-option but instead calculates/includes an IPv6 ND message checksum. The OMNI interface that receives the message applies any lower-layer authentication and integrity checks, then verifies both

the OAL checksum and the IPv6 ND message checksum. (Note that optimized implementations can verify both the OAL and IPv6 ND message checksums in a single pass over the data.) When an OMNI interface sends IPv6 ND messages to a synchronized neighbor, it includes an authentication sub-option only if authentication is necessary; otherwise, it calculates/includes the IPv6 ND message checksum.

When the OMNI interface calculates the authentication signature or IPv6 ND message checksum, it performs the calculation beginning with a pseudo-header of the IPv6 ND message header and extends over all following OAL packet data. In particular, for OAL super-packets any additional original IP packets included beyond the end of the IPv6 ND message are simply considered as extensions of the IPv6 ND message for the purpose of the calculation.

OAL destinations discard carrier packets with unacceptable Identifications and submit the encapsulated fragments in all others for reassembly. The reassembly algorithm rejects any fragments with unacceptable sizes, offsets, etc. and reassembles all others. Following reassembly, the OAL checksum algorithm provides an integrity assurance layer that compliments any integrity checks already applied by lower layers as well as a first-pass filter for any checks that will be applied later by upper layers.

Appendix C. VDL Mode 2 Considerations

ICAO Doc 9776 is the "Technical Manual for VHF Data Link Mode 2" (VDLM2) that specifies an essential radio frequency data link service for aircraft and ground stations in worldwide civil aviation air traffic management. The VDLM2 link type is "multicast capable" [RFC4861], but with considerable differences from common multicast links such as Ethernet and IEEE 802.11.

First, the VDLM2 link data rate is only 31.5Kbps - multiple orders of magnitude less than most modern wireless networking gear. Second, due to the low available link bandwidth only VDLM2 ground stations (i.e., and not aircraft) are permitted to send broadcasts, and even so only as compact layer 2 "beacons". Third, aircraft employ the services of ground stations by performing unicast RS/RA exchanges upon receipt of beacons instead of listening for multicast RA messages and/or sending multicast RS messages.

This beacon-oriented unicast RS/RA approach is necessary to conserve the already-scarce available link bandwidth. Moreover, since the numbers of beaconing ground stations operating within a given spatial range must be kept as sparse as possible, it would not be feasible to have different classes of ground stations within the same region observing different protocols. It is therefore highly desirable that all ground stations observe a common language of RS/RA as specified in this document.

Note that links of this nature may benefit from compression techniques that reduce the bandwidth necessary for conveying the same amount of data. The IETF lpwan working group is considering possible alternatives: [<https://datatracker.ietf.org/wg/lpwan/documents>].

Appendix D. Client-Proxy/Server Isolation Through Link-Layer Address Mapping

Per [RFC4861], IPv6 ND messages may be sent to either a multicast or unicast link-scoped IPv6 destination address. However, IPv6 ND messaging should be coordinated between the Client and Proxy/Server only without invoking other nodes on the underlay network. This implies that Client-Proxy/Server control messaging should be isolated and not overheard by other nodes on the link.

To support Client-Proxy/Server isolation on some links, Proxy/Servers can maintain an OMNI-specific unicast link-layer address ("MSADDR"). For Ethernet-compatible links, this specification reserves one Ethernet unicast address TBD5 (see: IANA Considerations). For non-Ethernet statically-addressed links MSADDR is reserved per the assigned numbers authority for the link-layer addressing space. For still other links, MSADDR may be dynamically discovered through other means, e.g., link-layer beacons.

Clients map the L3 addresses of all IPv6 ND messages they send (i.e., both multicast and unicast) to MSADDR instead of to an ordinary unicast or multicast link-layer address. In this way, all of the Client's IPv6 ND messages will be received by Proxy/Servers that are configured to accept packets destined to MSADDR. Note that multiple Proxy/Servers on the link could be configured to accept packets destined to MSADDR, e.g., as a basis for supporting redundancy.

Therefore, Proxy/Servers must accept and process packets destined to MSADDR, while all other devices must not process packets destined to MSADDR. This model has well-established operational experience in Proxy Mobile IPv6 (PMIP) [RFC5213][RFC6543].

Appendix E. Change Log

<< RFC Editor - remove prior to publication >>

Differences from earlier versions:

- * Submit for RFC publication.

Author's Address

Fred L. Templin (editor)
The Boeing Company
P.O. Box 3707
Seattle, WA 98124
United States of America
Email: fltemplin@acm.org