

Internet Area WG
Internet-Draft
Intended status: Best Current Practice
Expires: December 12, 2018

R. Bonica
Juniper Networks
F. Baker
Unaffiliated
G. Huston
APNIC
R. Hinden
Check Point Software
O. Troan
Cisco
F. Gont
SI6 Networks
June 10, 2018

IP Fragmentation Considered Fragile
draft-bonica-intarea-frag-fragile-02

Abstract

This document provides an overview of IP fragmentation. It explains how IP fragmentation works and why it is required. As part of that explanation, this document also explains how IP fragmentation reduces the reliability of Internet communication.

This document also proposes alternatives to IP fragmentation. Finally, it provides recommendations for application developers and network operators.

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 December 12, 2018.

Copyright Notice

Copyright (c) 2018 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
2. IP Fragmentation	3
2.1. Links, Paths, MTU and PMTU	3
2.2. Upper-layer Protocols	5
3. Requirements Language	7
4. IP Fragmentation Reduces Reliability	7
4.1. Middle Box Failures	7
4.2. Partial Filtering	8
4.3. Suboptimal Load Balancing	8
4.4. Security Vulnerabilities	9
4.5. Blackholing Due to ICMP Loss	11
4.5.1. Transient Loss	12
4.5.2. Incorrect Implementation of Security Policy	12
4.5.3. Persistent Loss Caused By Anycast	13
4.6. Blackholing Due To Filtering	13
5. Alternatives to IP Fragmentation	14
5.1. Transport Layer Solutions	14
5.2. Application Layer Solutions	15
6. Applications That Rely on IPv6 Fragmentation	16
6.1. DNS	16
6.2. OSPFv3	17
6.3. Packet-in-Packet Encapsulations	17
7. Recommendations	17
7.1. For Application Developers	17
7.2. For Network Operators	17
8. IANA Considerations	18
9. Security Considerations	18
10. Acknowledgements	18
11. References	18
11.1. Normative References	18
11.2. Informative References	19

Appendix A. Contributors' Address	22
Authors' Addresses	22

1. Introduction

Operational experience [RFC7872] [Huston] reveals that IP fragmentation reduces the reliability of Internet communication. This document provides an overview of IP fragmentation. It explains how IP fragmentation works and why it is required. As part of that explanation, this document also explains how IP fragmentation reduces the reliability of Internet communication.

This document also proposes alternatives to IP fragmentation. Finally, it provides recommendations for application developers and network operators.

2. IP Fragmentation

2.1. Links, Paths, MTU and PMTU

An Internet path connects a source node to a destination node. A path can contain links and intermediate systems. If a path contains more than one link, the links are connected in series and an intermediate system connects each link to the next. An intermediate system can be a router or a middle box.

Internet paths are dynamic. Assume that the path from one node to another contains a set of links and intermediate systems. If the network topology changes, that path can also change so that it includes a different set of links and intermediate systems.

Each link is constrained by the number of bytes that it can convey in a single IP packet. This constraint is called the link Maximum Transmission Unit (MTU). IPv4 [RFC0791] requires every link to have an MTU of 68 bytes or greater. IPv6 [RFC8200] requires every link to have an MTU of 1280 bytes or greater. These are called the IPv4 and IPv6 minimum link MTU's.

Each Internet path is constrained by the number of bytes that it can convey in a IP single packet. This constraint is called the Path MTU (PMTU). For any given path, the PMTU is equal to the smallest of its link MTU's. Because Internet paths are dynamic, PMTU is also dynamic.

For reasons described below, source nodes estimate the PMTU between themselves and destination nodes. A source node can produce extremely conservative PMTU estimates in which:

- o The estimate for each IPv4 path is equal to the IPv4 minimum link MTU.
- o The estimate for each IPv6 path is equal to the IPv6 minimum link MTU.

While these conservative estimates are guaranteed to be less than or equal to the actual MTU, they are likely to be much less than the actual PMTU. This may adversely affect upper-layer protocol performance.

By executing Path MTU Discovery (PMTUD) [RFC1191] [RFC8201] procedures, a source node can maintain a less conservative, running estimate of the PMTU between itself and a destination node. According to these procedures, the source node produces an initial PMTU estimate. This initial estimate is equal to the MTU of the first link along the path to the destination node. It can be greater than the actual PMTU.

Having produced an initial PMTU estimate, the source node sends non-fragmentable IP packets to the destination node. If one of these packets is larger than the actual PMTU, a downstream router will not be able to forward the packet through the next link along the path. Therefore, the downstream router drops the packet and sends an Internet Control Message Protocol (ICMP) [RFC0792] [RFC4443] Packet Too Big (PTB) message to the source node. The ICMP PTB message indicates the MTU of the link through which the packet could not be forwarded. The source node uses this information to refine its PMTU estimate.

PMTUD produces a running estimate of the PMTU between a source node and a destination node. Because PMTU is dynamic, at any given time, the PMTU estimate can differ from the actual PMTU. In order to detect PMTU increases, PMTUD occasionally resets the PMTU estimate to the MTU of the first link along path to the destination node. It then repeats the procedure described above.

PMTUD has the following characteristics:

- o It relies on the network's ability to deliver ICMP PTB messages to the source node.
- o It is susceptible to attack because ICMP messages are easily forged [RFC5927].

FOOTNOTE: According to RFC 0791, every IPv4 host must be capable of receiving a packet whose length is equal to 576 bytes. However, the

IPv4 minimum link MTU is not 576. Section 3.2 of RFC 0791 explicitly states that the IPv4 minimum link MTU is 68 bytes.

FOOTNOTE: In the paragraphs above, the term "non-fragmentable packet" is introduced. A non-fragmentable packet can be fragmented at its source. However, it cannot be fragmented by a downstream node. An IPv4 packet whose DF-bit is set to zero is fragmentable. An IPv4 packet whose DF-bit is set to one is non-fragmentable. All IPv6 packets are also non-fragmentable.

FOOTNOTE: In the paragraphs above, the term "ICMP PTB message" is introduced. The ICMP PTB message has two instantiations. In ICMPv4 [RFC0792], the ICMP PTB message is Destination Unreachable message with Code equal to (4) fragmentation needed and DF set. This message was augmented by [RFC1191] to indicate the MTU of the link through which the packet could not be forwarded. In ICMPv6 [RFC4443], the ICMP PTB message is a Packet Too Big Message with Code equal to (0). This message also indicates the MTU of the link through which the packet could not be forwarded.

2.2. Upper-layer Protocols

When an upper-layer protocol submits data to the underlying IP module, and the resulting IP packet's length is greater than the PMTU, IP fragmentation may be required. IP fragmentation divides a packet into fragments. Each fragment includes an IP header and a portion of the original packet.

[RFC0791] describes IPv4 fragmentation procedures. IPv4 packets whose DF-bit is set to one cannot be fragmented. IPv4 packets whose DF-bit is set to zero can be fragmented at the source node or by any downstream router. [RFC8200] describes IPv6 fragmentation procedures. IPv6 packets can be fragmented at the source node only.

IPv4 fragmentation differs slightly from IPv6 fragmentation. However, in both IP versions, the upper-layer header appears in the first fragment only. It does not appear in subsequent fragments.

Upper-layer protocols can operate in the following modes:

- o Do not rely on IP fragmentation.
- o Rely on IP source fragmentation only (i.e., fragmentation at the source node).
- o Rely on IP source fragmentation and downstream fragmentation (i.e., fragmentation at any node along the path).

Upper-layer protocols running over IPv4 can operate in all of the above-mentioned modes. Upper-layer protocols running over IPv6 can operate in the first and second modes only.

Upper-layer protocols that operate in the first two modes (above) require access to the PMTU estimate. In order to fulfil this requirement, they can

- o Estimate the PMTU to be equal to the IPv4 or IPv6 minimum link MTU.
- o Access the estimate that PMTUD produced.
- o Execute PMTUD procedures themselves.
- o Execute Packetization Layer PMTUD (PLPMTUD) [RFC4821] [I-D.fairhurst-tsvwg-datagram-plpmtud] procedures.

According to PLPMTUD procedures, the upper-layer protocol maintains a running PMTU estimate. It does so by sending probe packets of various sizes to its peer and receiving acknowledgements. This strategy differs from PMTUD in that it relies on acknowledgement of received messages, as opposed to ICMP PTB messages concerning dropped messages. Therefore, PLPMTUD does not rely on the network's ability to deliver ICMP PTB messages to the source.

An upper-layer protocol that does not rely on IP fragmentation never causes the underlying IP module to emit

- o A fragmentable IP packet (i.e., an IPv4 packet with the DF-bit set to zero).
- o An IP fragment.
- o A packet whose length is greater than the PMTU estimate.

However, when the PMTU estimate is greater than the actual PMTU, the upper-layer protocol can cause the underlying IP module to emit a packet whose length is greater than the actual PMTU. When this occurs, a downstream router drops the packet and the source node refines its PMTU estimate, employing either PMTUD or PLPMTUD procedures.

When an upper-layer protocol that relies on IP source fragmentation only submits data to the underlying IP module, and the resulting packet is larger than the PMTU estimate, the underlying IP module fragments the packet and emits the fragments. However, the upper-layer protocol never causes the underlying IP module to emit

- o A fragmentable IP packet.
- o A packet whose length is greater than the PMTU estimate.

When the PMTU estimate is greater than the actual PMTU, the upper-layer protocol can cause the underlying IP module to emit a packet whose length is greater than the actual PMTU. When this occurs, a downstream router drops the packet and the source node refines its PMTU estimate, employing either PMTUD or PLPMTUD procedures.

An upper-layer protocol that relies on IP source fragmentation and downstream fragmentation can cause the underlying IP module to emit

- o A fragmentable IP packet.
- o An IP fragment.
- o A packet whose length is greater than the PMTU estimate.

A protocol that relies on IP source fragmentation and downstream fragmentation does not require access to the PMTU estimate. For these protocols, the underlying IP module:

- o Fragments all packets whose length exceeds the MTU of the first link along the path to the destination.
- o Sets the DF-bit to zero, so that downstream nodes can fragment the packet.

3. 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.

4. IP Fragmentation Reduces Reliability

This section explains how IP fragmentation reduces the reliability of Internet communication.

4.1. Middle Box Failures

Many middle boxes require access to the transport-layer header. However, when a packet is divided into fragments, the transport-layer header appears in the first fragment only. It does not appear in

subsequent fragments. This omission can prevent middle boxes from delivering their intended services.

For example, assume that a router diverts selected packets from their normal path towards network appliances that support deep packet inspection and lawful intercept. The router selects packets for diversion based upon the following 5-tuple:

- o IP Source Address.
- o IP Destination Address.
- o IPv4 Protocol or IPv6 Next Header.
- o transport-layer source port.
- o transport-layer destination port.

IP fragmentation causes this selection algorithm to behave suboptimally, because the transport-layer header appears only in the first fragment of each packet.

In another example, a middle box remarks a packet's Differentiated Services Code Point [RFC2474] based upon the above-mentioned 5-tuple. IP fragmentation causes this process to behave suboptimally, because the transport-layer header appears only in the first fragment of each packet.

In all of the above-mentioned examples, the middle box cannot deliver its intended service without reassembling fragmented packets.

4.2. Partial Filtering

IP fragments cause problems for firewalls whose filter rules include decision making based on TCP and UDP ports. As the port information is not in the trailing fragments the firewall may elect to accept all trailing fragments, which may admit certain classes of attack, or may elect to block all trailing fragments, which may block otherwise legitimate traffic, or may elect to reassemble all fragmented packets, which may be inefficient and negatively affect performance.

4.3. Suboptimal Load Balancing

Many stateless load-balancers require access to the transport-layer header. Assume that a load-balancer distributes flows among parallel links. In order to optimize load balancing, the load-balancer sends every packet or packet fragment belonging to a flow through the same link.

In order to assign a packet or packet fragment to a link, the load-balancer executes an algorithm. If the packet or packet fragment contains a transport-layer header, the load balancing algorithm accepts the following 5-tuple as input:

- o IP Source Address.
- o IP Destination Address.
- o IPv4 Protocol or IPv6 Next Header.
- o transport-layer source port.
- o transport-layer destination port.

However, if the packet or packet fragment does not contain a transport-layer header, the load balancing algorithm accepts only the following 3-tuple as input:

- o IP Source Address.
- o IP Destination Address.
- o IPv4 Protocol or IPv6 Next Header.

Therefore, non-fragmented packets belonging to a flow can be assigned to one link while fragmented packets belonging to the same flow can be divided between that link and another. This can cause suboptimal load balancing.

4.4. Security Vulnerabilities

Security researchers have documented several attacks that rely on IP fragmentation. The following are examples:

- o Overlapping fragment attack [RFC1858][RFC3128] [RFC5722]
- o Resource exhaustion attacks (such as the Rose Attack)
- o Attacks based on predictable fragment identification values [RFC7739]
- o Attacks based on bugs in the implementation of the fragment reassembly algorithm
- o Evasion of Network Intrusion Detection Systems (NIDS) [Ptacek1998]

In the overlapping fragment attack, an attacker constructs a series of packet fragments. The first fragment contains an IP header, a transport-layer header, and some transport-layer payload. This fragment complies with local security policy and is allowed to pass through a stateless firewall. A second fragment, having a non-zero offset, overlaps with the first fragment. The second fragment also passes through the stateless firewall. When the packet is reassembled, the transport layer header from the first fragment is overwritten by data from the second fragment. The reassembled packet does not comply with local security policy. Had it traversed the firewall in one piece, the firewall would have rejected it.

A stateless firewall cannot protect against the overlapping fragment attack. However, destination nodes can protect against the overlapping fragment attack by implementing the reassembly procedures described in RFC 1858, RFC 3128 and RFC 8200. These reassembly procedures detect the overlap and discard the packet.

The fragment reassembly algorithm is a stateful procedure for an otherwise stateless protocol. As such, it can be exploited for resource exhaustion attacks. An attacker can construct a series of fragmented packets, with one fragment missing from each packet so that the reassembly process cannot complete. Thus, this attack causes resource exhaustion on the destination node, possibly denying reassembly services to other flows. This type of attack can be mitigated by flushing fragment reassembly buffers when necessary, at the expense of possibly dropping legitimate fragments.

An IP fragment contains an "Identification" field that, together with the IP Source Address and Destination Address of a packet, identifies fragments that correspond to the same original datagram, so that they can be reassembled together by the receiving host. Many implementations have employed predictable values for the Identification field, thus making it easy for an attacker to forge malicious IP fragments that would cause the reassembly procedure for legitimate packets to fail.

Over the years multiple IPv4 and IPv6 implementations have been found to have flaws in their implementation of the IP fragment reassembly algorithm, typically resulting in buffer overflows. These buffer overflows have been exploitable for denial of service and remote code execution attacks.

NIDS aims at identifying malicious activity by analyzing network traffic. Ambiguity in the possible result of the fragment reassembly process may allow an attacker to evade these systems. Many of these systems try to mitigate some of these evasion techniques by e.g.

computing all possible outcomes of the fragment reassembly process, at the expense of increased processing requirements.

4.5. Blackholing Due to ICMP Loss

As stated above, an upper-layer protocol requires access the PMTU estimate if it:

- o Does not rely on IP fragmentation.
- o Relies on IP source fragmentation only (i.e., fragmentation at the source node).

In order to satisfy this requirement, the upper-layer protocol can:

- o Estimate the PMTU to be equal to the IPv4 or IPv6 minimum link MTU.
- o Access the estimate that PMTUD produced.
- o Execute PMTUD procedures itself.
- o Execute PLPMTUD procedures.

PMTUD relies upon the network's ability to deliver ICMP PTB messages to the source node. Therefore, if an upper-layer protocol relies on PMTUD, it also relies on the network's ability to deliver ICMP PTB messages to the source node.

According to [RFC4890], ICMP PTB messages must not be filtered. However, ICMP PTB delivery is not reliable. It is subject to both transient and persistent loss.

Transient loss of ICMP PTB messages causes PMTUD to perform less efficiently, but does not cause it to fail completely. When the conditions contributing to transient loss abate, the network regains its ability to deliver ICMP PTB messages and PMTUD regains its ability to function. Section 4.5.1 of this document describes conditions that lead to transient loss of ICMP PTB messages.

However, persistent loss of ICMP PTB messages causes PMTUD to fail completely. Section 4.5.2 and Section 4.5.3 of this document describe conditions that lead to persistent loss of ICMP PTB messages.

The problem described in this section is specific to PMTUD. It does not occur when the upper-layer protocol obtains its PMTU estimate from PLPMTUD or any other source.

4.5.1. Transient Loss

The following factors can contribute to transient loss of ICMP PTB messages:

- o Network congestion.
- o Packet corruption.
- o Transient routing loops.
- o ICMP rate limiting.

The effect of rate limiting may be severe, as RFC 4443 recommends strict rate limiting of IPv6 traffic.

4.5.2. Incorrect Implementation of Security Policy

Incorrect implementation of security policy can cause persistent loss of ICMP PTB messages.

Assume that a Customer Premise Equipment (CPE) router implements the following zone-based security policy:

- o Allow any traffic to flow from the inside zone to the outside zone.
- o Do not allow any traffic to flow from the outside zone to the inside zone unless it is part of an existing flow (i.e., it was elicited by an outbound packet).

When a correct implementation of the above-mentioned security policy receives an ICMP PTB message, it examines the ICMP PTB payload in order to determine the original packet (i.e., the packet that elicited the ICMP PTB message) belonged to an existing flow. If the original packet belonged to an existing flow, the implementation allows the ICMP PTB to flow from the outside zone to the inside zone. If not, the implementation discards the ICMP PTB message.

When a incorrect implementation of the above-mentioned security policy receives an ICMP PTB message, it discards the packet because its source address is not associated with an existing flow.

The security policy described above is implemented incorrectly on many consumer CPE routers.

4.5.3. Persistent Loss Caused By Anycast

Anycast can cause persistent loss of ICMP PTB messages. Consider the example below:

A DNS client sends a request to an anycast address. The network routes that DNS request to the nearest instance of that anycast address (i.e., a DNS Server). The DNS server generates a response and sends it back to the DNS client. While the response does not exceed the DNS server's PMTU estimate, it does exceed the actual PMTU.

A downstream router drops the packet and sends an ICMP PTB message the packet's source (i.e., the anycast address). The network routes the ICMP PTB message to the anycast instance closest to the downstream router. Sadly, that anycast instance may not be the DNS server that originated the DNS response. It may be another DNS server with the same anycast address. The DNS server that originated the response may never receive the ICMP PTB message and may never update its PMTU estimate.

4.6. Blackholing Due To Filtering

In RFC 7872, researchers sampled Internet paths to determine whether they would convey packets that contain IPv6 extension headers. Sampled paths terminated at popular Internet sites (e.g., popular web, mail and DNS servers).

The study revealed that at least 28% of the sampled paths did not convey packets containing the IPv6 Fragment extension header. In most cases, fragments were dropped in the destination autonomous system. In other cases, the fragments were dropped in transit autonomous systems.

Another recent study [Huston] confirmed this finding. It reported that 37% of sampled endpoints used IPv6-capable DNS resolvers that were incapable of receiving a fragmented IPv6 response.

It is difficult to determine why network operators drop fragments. Possible causes follow:

- o Hardware inability to process fragmented packets.
- o Failure to change a vendor defaults.
- o Unintentional misconfiguration.

- o Intentional configuration (e.g., network operators consciously chooses to drop IPv6 fragments in order to address the issues raised in Section 4.1 through Section 4.5, above.)

5. Alternatives to IP Fragmentation

5.1. Transport Layer Solutions

The Transport Control Protocol (TCP) [RFC0793]) can be operated in a mode that does not require IP fragmentation.

Applications submit a stream of data to TCP. TCP divides that stream of data into segments, with no segment exceeding the TCP Maximum Segment Size (MSS). Each segment is encapsulated in a TCP header and submitted to the underlying IP module. The underlying IP module prepends an IP header and forwards the resulting packet.

If the TCP MSS is sufficiently small, the underlying IP module never produces a packet whose length is greater than the actual PMTU. Therefore, IP fragmentation is not required.

TCP offers the following mechanisms for MSS management:

- o Manual configuration
- o PMTUD
- o PLPMTUD

For IPv6 nodes, manual configuration is always applicable. If the MSS is manually configured to 1220 bytes and the packet does not contain extension headers, the IP layer will never produce a packet whose length is greater than the IPv6 minimum link MTU (1280 bytes). However, manual configuration prevents TCP from taking advantage of larger link MTU's.

RFC 8200 strongly recommends that IPv6 nodes implement PMTUD, in order to discover and take advantage of path MTUs greater than 1280 bytes. However, as mentioned in Section 2.1, PMTUD relies upon the network's ability to deliver ICMP PTB messages. Therefore, PMTUD is applicable only in environments where the risk of ICMP PTB loss is acceptable.

By contrast, PLPMTUD does not rely upon the network's ability to deliver ICMP PTB messages. However, in many loss-based TCP congestion control algorithms, the dropping of a packet may cause the TCP control algorithm to drop the congestion control window, or even re-start with the entire slow start process. For high capacity, long

round-trip time, large volume TCP streams, the deliberate probing with large packets and the consequent packet drop may impose too harsh a penalty on total TCP throughput for it to be a viable approach. [RFC4821] defines PLPMTUD procedures for TCP.

While TCP will never cause the underlying IP module to emit a packet that is larger than the PMTU estimate, it can cause the underlying IP module to emit a packet that is larger than the actual PMTU. If this occurs, the packet is dropped, the PMTU estimate is updated, the segment is divided into smaller segments and each smaller segment is submitted to the underlying IP module.

The Datagram Congestion Control Protocol (DCCP) [RFC4340] and the Stream Control Protocol (SCP) [RFC4960] also can be operated in a mode that does not require IP fragmentation. They both accept data from an application and divide that data into segments, with no segment exceeding a maximum size. Both DCCP and SCP offer manual configuration, PMTUD and PLPMTUD as mechanisms for managing that maximum size. [I-D.fairhurst-tsvwg-datagram-plpmtud] proposes PLPMTUD procedures for DCCP and SCP.

Currently, User Data Protocol (UDP) [RFC0768] lacks a fragmentation mechanism of its own and relies on IP fragmentation. However, [I-D.ietf-tsvwg-udp-options] proposes a fragmentation mechanism for UDP.

5.2. Application Layer Solutions

[RFC8085] recognizes that IP fragmentation reduces the reliability of Internet communication. It also recognizes that UDP lacks a fragmentation mechanism of its own and relies on IP fragmentation. Therefore, [RFC8085] offers the following advice regarding applications the run over the UDP.

"An application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself to determine whether the path to a destination will support its desired message size without fragmentation."

RFC 8085 continues:

"Applications that do not follow the recommendation to do PMTU/PLPMTUD discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending

messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU. For IPv6, EMTU_S is 1280 bytes. The effective PMTU for a directly connected destination (with no routers on the path) is the configured interface MTU, which could be less than the maximum link payload size. Transmission of minimum-sized UDP datagrams is inefficient over paths that support a larger PMTU, which is a second reason to implement PMTU discovery."

RFC 8085 assumes that for IPv4, an EMTU_S of 576 is sufficiently small, even though the IPv4 minimum link MTU is 68 bytes.

This advice applies equally to application that run directly over IP.

6. Applications That Rely on IPv6 Fragmentation

The following applications rely on IPv6 fragmentation:

- o DNS [RFC1035]
- o OSPFv3 [RFC5340]
- o Packet-in-packet encapsulations

Each of these applications relies on IPv6 fragmentation to a varying degree. In some cases, that reliance is essential, and cannot be broken without fundamentally changing the protocol. In other cases, that reliance is incidental, and most implementations already take appropriate steps to avoid fragmentation.

This list is not comprehensive, and other protocols that rely on IPv6 fragmentation may exist. They are not specifically considered in the context of this document.

6.1. DNS

DNS relies on UDP for efficiency, and the consequence is the use of IP fragmentation for large responses, as permitted by the DNS EDNS(0) options in the query. It is possible to mitigate the issue of fragmentation-based packet loss by having queries use smaller EDNS(0) UDP buffer sizes, but then the operational issue of the partial level of support for DNS over TCP over IPv6 becomes a limiting factor of the efficacy of this approach in an IPv6 context [Damas].

Larger DNS responses can normally be avoided by aggressively pruning the Additional section of DNS responses. One scenario where such pruning is ineffective is in the use of DNSSEC, where large key sizes act to increase the response size to certain DNS queries. There is

no effective response to this situation within the DNS other than using smaller cryptographic keys and adoption of DNSSEC administrative practices that attempt to keep DNS response as short as possible.

6.2. OSPFv3

OSPFv3 implementations can emit messages large enough to cause IPv6 fragmentation. However, in keeping with the recommendations of RFC8200, and in order to optimize performance, most OSPFv3 implementations restrict their maximum message size to the IPv6 minimum link MTU.

6.3. Packet-in-Packet Encapsulations

In this document, packet-in-packet encapsulations include IP-in-IP [RFC2003], Generic Routing Encapsulation (GRE) [RFC2784], GRE-in-UDP [RFC8086] and Generic Packet Tunneling in IPv6 [RFC2473]. [RFC4459] describes fragmentation issues associated with all of the above-mentioned encapsulations.

The fragmentation strategy described for GRE in [RFC7588] has been deployed for all of the above-mentioned encapsulations. This strategy does not rely on IPv6 fragmentation except in one corner case. (see Section 3.3.2.2 of RFC 7588 and Section 7.1 of RFC 2473). Section 3.3 of [RFC7676] further describes this corner case.

7. Recommendations

7.1. For Application Developers

Application developers SHOULD NOT develop applications that rely on IPv6 fragmentation.

Application-layer protocols then depend upon IPv6 fragmentation SHOULD be updated to break that dependency.

7.2. For Network Operators

As per RFC 4890, network operators MUST NOT filter ICMPv6 PTB messages unless they are known to be forged or otherwise illegitimate. As stated in Section 4.5, filtering ICMPv6 PTB packets causes PMTUD to fail. Operators MUST ensure proper PMTUD operation in their network, including making sure the network generates PTB packets when dropping packets too large compared to outgoing interface MTU.

Many upper-layer protocols rely on PMTUD.

8. IANA Considerations

This document makes no request of IANA.

9. Security Considerations

This document mitigates some of the security considerations associated with IP fragmentation by discouraging the use of IP fragmentation. It does not introduce any new security vulnerabilities, because it does not introduce any new alternatives to IP fragmentation. Instead, it recommends well-understood alternatives.

10. Acknowledgements

Thanks to Mikael Abrahamsson, Mike Heard, Tom Herbert, Tatuya Jinmei, Eric Nygren, and Joe Touch for their comments.

11. References

11.1. 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>>.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [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>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.

- [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>>.
- [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>>.
- [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>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [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>>.

11.2. Informative References

- [Damas] Damas, J. and G. Huston, "Measuring ATR", April 2018, <<http://www.potaroo.net/ispcol/2018-04/atr.html>>.
- [Huston] Huston, G., "IPv6, Large UDP Packets and the DNS (<http://www.potaroo.net/ispcol/2017-08/xtn-hdrs.html>)", August 2017.
- [I-D.fairhurst-tsvwg-datagram-plpmtud] Fairhurst, G., Jones, T., Tuexen, M., and I. Ruengeler, "Packetization Layer Path MTU Discovery for Datagram Transports", draft-fairhurst-tsvwg-datagram-plpmtud-02 (work in progress), December 2017.

- [I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-02 (work in progress), January 2018.
- [Ptacek1998]
Ptacek, T. and T. Newsham, "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection", 1998, <<http://www.aciri.org/vern/Ptacek-Newsham-Evasion-98.ps>>.
- [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>>.
- [RFC1858] Ziemba, G., Reed, D., and P. Traina, "Security Considerations for IP Fragment Filtering", RFC 1858, DOI 10.17487/RFC1858, October 1995, <<https://www.rfc-editor.org/info/rfc1858>>.
- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.
- [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>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [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>>.
- [RFC3128] Miller, I., "Protection Against a Variant of the Tiny Fragment Attack (RFC 1858)", RFC 3128, DOI 10.17487/RFC3128, June 2001, <<https://www.rfc-editor.org/info/rfc3128>>.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <<https://www.rfc-editor.org/info/rfc4340>>.

- [RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, DOI 10.17487/RFC4459, April 2006, <<https://www.rfc-editor.org/info/rfc4459>>.
- [RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <<https://www.rfc-editor.org/info/rfc4890>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <<https://www.rfc-editor.org/info/rfc5722>>.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <<https://www.rfc-editor.org/info/rfc5927>>.
- [RFC7588] Bonica, R., Pignataro, C., and J. Touch, "A Widely Deployed Solution to the Generic Routing Encapsulation (GRE) Fragmentation Problem", RFC 7588, DOI 10.17487/RFC7588, July 2015, <<https://www.rfc-editor.org/info/rfc7588>>.
- [RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC 7676, DOI 10.17487/RFC7676, October 2015, <<https://www.rfc-editor.org/info/rfc7676>>.
- [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>>.
- [RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <<https://www.rfc-editor.org/info/rfc7872>>.

[RFC8086] Yong, L., Ed., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", RFC 8086, DOI 10.17487/RFC8086, March 2017, <<https://www.rfc-editor.org/info/rfc8086>>.

Appendix A. Contributors' Address

Authors' Addresses

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Fred Baker
Unaffiliated
Santa Barbara, California 93117
USA

Email: FredBaker.IETF@gmail.com

Geoff Huston
APNIC
6 Cordelia St
Brisbane, 4101 QLD
Australia

Email: gih@apnic.net

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, California 94070
USA

Email: bob.hinden@gmail.com

Ole Troan
Cisco
Philip Pedersens vei 1
N-1366 Lysaker
Norway

Email: ot@cisco.com

Fernando Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires
Argentina

Email: fgont@si6networks.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 2, 2019

B. Carpenter
Univ. of Auckland
S. Jiang
Huawei Technologies Co., Ltd
July 1, 2018

Limited Domains and Internet Protocols
draft-carpenter-limited-domains-01

Abstract

There is a noticeable trend towards network requirements, behaviours and semantics that are specific to a limited region of the Internet and a particular set of requirements. Policies, default parameters, the options supported, the style of network management and security requirements may vary. This document reviews examples of such limited domains and emerging solutions. It shows the needs for a precise definition of a limited domain boundary and for a corresponding protocol to allow nodes to discover where such a boundary exists.

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 January 2, 2019.

Copyright Notice

Copyright (c) 2018 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
2. Examples of Limited Domain Requirements	3
3. Examples of Limited Domain Solutions	5
4. Common Aspects of Limited Domains	8
5. The Need to Define a Limited Domain Boundary	8
6. Defining Protocol Scope	8
7. Security Considerations	8
8. IANA Considerations	8
9. Acknowledgements	8
10. Informative References	9
Appendix A. Change log [RFC Editor: Please remove]	12
Authors' Addresses	12

1. Introduction

As the Internet continues to grow and diversify, with a realistic prospect of tens of billions of nodes being connected directly and indirectly, there is a noticeable trend towards local requirements, behaviours and semantics. The word "local" should be understood in a special sense, however. In some cases it may refer to geographical and physical locality - all the nodes in a single building, on a single campus, or in a given vehicle. In other cases it may refer to a defined set of users or nodes distributed over a much wider area, but drawn together by a single virtual network over the Internet, or a single physical network running partially in parallel with the Internet. We expand on these possibilities below. To capture the topic, this document refers to such networks as "limited domains".

Some people have concerns about splintering of the Internet along political or linguistic boundaries by mechanisms that block the free flow of information across the network. That is not the topic of this document, which does not discuss filtering mechanisms and does not apply to protocols that are designed for use across the whole Internet. It is only concerned with domains that have specific technical requirements.

The word "domain" in this document does not refer to naming domains in the DNS, although in some cases a limited domain might incidentally be congruent with a DNS domain.

The requirements of limited domains will be different in different scenarios. Policies, default parameters, and the options supported may vary. Also, the style of network management may vary, between a completely unmanaged network, one with fully autonomic management, one with traditional central management, and mixtures of the above. Finally, the requirements and solutions for security and privacy may vary.

This documents analyses and discusses some of the consequences of this trend, and how it impacts the idea of universal interoperability in the Internet. In particular, we challenge the notion that all Internet standards must be universal in scope and applicability. To the contrary, we assert that some standards need to be specifically limited in their applicability. This requires that the concepts of a limited domain, and of its boundary, need to be formalised.

NOTE: This document is incomplete. Comments on the following two sections are invited before we complete the later sections.

2. Examples of Limited Domain Requirements

This section describes various examples where limited domain requirements can be identified. It is of course not a complete list.

NOTE: The authors welcome more suggestions and references for this list.

1. A home network. It will be unmanaged, constructed by a non-specialist, and will possibly include wiring errors such as physical loops. It must work with devices "out of the box" as shipped by their manufacturers and must create adequate security by default. Remote access may be required. The requirements and applicable principles are summarised in [RFC7368].
2. A small office network. This is very similar to a home network, since whoever is in charge will probably have little or no specialist knowledge, but may have differing security and privacy requirements. Remote access may be required.
3. A vehicle network. This will be designed by the vehicle manufacturer but may include devices added by the vehicle's owner or operator. Parts of the network will have demanding performance and reliability requirements with implications for human safety. Remote access may be required to certain functions, but absolutely forbidden for others. Communication with other vehicles, roadside infrastructure, and external data sources will be required. See

[I-D.ietf-ipwave-vehicular-networking] for a survey of use cases.

4. A building services network. This will be designed specifically for a particular building, but using standard components. Additional devices may need to be added at any time. Parts of the network may have demanding reliability requirements with implications for human safety. Remote access may be required to certain functions, but absolutely forbidden for others. [I-D.martocci-6lowapp-building-applications] (need current reference!)
5. Supervisory Control And Data Acquisition (SCADA) networks in general, which will exhibit widely differing requirements, including tough real-time performance targets, of which building networks are a simple example. See for example [I-D.ietf-detnet-use-cases]
6. The three preceding cases will all include sensors, but some networks may be specifically limited to sensors and the collection and processing of sensor data. They may be in remote or technically challenging locations and installed by non-specialists.
7. "Traditional" enterprise and campus networks, which may be spread over many kilometres and over multiple separate sites.
8. Data centres and hosting centres, or distributed services acting as such centres. These will have high performance, security and privacy requirements and will typically include large numbers of independent "tenant" networks overlaid on shared infrastructure.
9. Content Delivery Networks, comprising distributed data centres and the paths between them, spanning thousands of kilometres.
10. Internet of Things (IoT) networks. While this term is very flexible and covers many innovative types of network, it seems reasonable to assert that many IoT edge networks will in fact have special requirements and protocols that are useful only within a specific domain, and that these protocols cannot, and for security reasons should not, run over the Internet as a whole.

Two other concepts, while not tied to specific network types, also strongly depend on the concept of limited domains:

1. Intent Based Networking. In this concept, a network domain is configured and managed in accordance with an abstract policy

known as "Intent", to ensure that the network performs as required [I-D.moulchan-nmrg-network-intent-concepts]. Whatever technologies are used to support this, they will be applied within the domain boundary.

2. Network Slicing. A network slice is a virtual network that consists of a managed set of resources carved off from a larger network [I-D.geng-netslices-architecture]. Whatever technologies are used to support slicing, they will require a clear definition of the boundary of a given slice.

While it is clearly desirable to use common solutions, and therefore common standards, wherever possible, it is increasingly difficult to do so while satisfying the widely varying requirements outlined above. However, there is a tendency when new protocols and protocol extensions are proposed to always ask the question "How will this work across the open Internet?" This document suggests that this is not always the right question. There are protocols and extensions that are not intended to work across the open Internet. On the contrary, their requirements and semantics are specifically limited (in the sense defined above).

A common argument is that if a protocol is intended for limited use, the chances are very high that it will in fact be used (or misused) in other scenarios including the so-called open Internet. This is undoubtedly true and means that limited use is not an excuse for bad design or poor security. In fact, a limited use requirement potentially adds complexity to both the protocol and its security design, as discussed later.

Nevertheless, because of the diversity of limited environments with specific requirements that is now emerging, specific standards will necessarily emerge. There will be attempts to capture each market sector, but the market will demand standardised limited solutions. However, the "open Internet" must remain as the universal method of interconnection. Reconciling these two aspects is a major challenge.

3. Examples of Limited Domain Solutions

This section lists various examples of specific limited domain solutions that have been proposed or defined. It intentionally does not include Layer 2 technology solutions, which are by definition defined for limited domains.

NOTE: Please suggest additional items for this list.

1. Differentiated Services. This mechanism [RFC2474] allows a network to assign locally significant values to the 6-bit

Differentiated Services Code Point field in any IP packet. Although there are some recommended codepoint values for specific per-hop queue management behaviours, these are specifically intended to be domain-specific codepoints with traffic being classified, conditioned and re-marked at domain boundaries (unless there is an inter-domain agreement that makes re-marking unnecessary).

2. Network function virtualisation. As described in [I-D.irtf-nfvrg-gaps-network-virtualization], this general concept is an open research topic, in which virtual network functions are orchestrated as part of a distributed system. Inevitably such orchestration applies to an administrative domain of some kind, even though cross-domain orchestration is also a research area.
3. Service Function Chaining (SFC). This technique [RFC7665] assumes that services within a network are constructed as sequences of individual functions within a specific SFC-enabled domain. As that RFC states: "Specific features may need to be enforced at the boundaries of an SFC-enabled domain, for example to avoid leaking SFC information". A Network Service Header (NSH) [RFC8300] is used to encapsulate packets flowing through the service function chain: "The intended scope of the NSH is for use within a single provider's operational domain."
4. Data Centre Network Virtualization Overlays. A common requirement in data centres that host many tenants (clients) is to provide each one with a secure private network, all running over the same physical infrastructure. [RFC8151] describes various use cases for this, and specifications are under development. These include use cases in which the tenant network is physically split over several data centres, but which must appear to the user as a single secure domain.
5. Segment Routing. This is a technique which "steers a packet through an ordered list of instructions, called segments" [I-D.ietf-spring-segment-routing]. The semantics of these instructions are explicitly local to a segment routing domain or even to a single node. Technically, these segments or instructions are represented as an MPLS label or an IPv6 address, which clearly adds a semantic interpretation to them within the domain.
6. Autonomic Networking. As explained in [I-D.ietf-anima-reference-model], an autonomic network is also a security domain within which an autonomic control plane [I-D.ietf-anima-autonomic-control-plane] is used by service

agents. These service agents manage technical objectives, which may be locally defined, subject to domain-wide policy. Thus the domain boundary is important for both security and protocol purposes.

7. Homenet. As shown in [RFC7368], a home networking domain has specific protocol needs that differ from those in an enterprise network or the Internet as a whole. These include the Home Network Control Protocol (HNCP) [RFC7788] and a naming and discovery solution [I-D.ietf-homenet-simple-naming].
8. Creative uses of IPv6 features. As IPv6 enters more general use, engineers notice that it has much more flexibility than IPv4. Innovative suggestions have been made for:
 - * The flow label, e.g. [RFC6294], [I-D.fioccola-v6ops-ipv6-alt-mark].
 - * Extension headers, e.g. for segment routing [I-D.ietf-6man-segment-routing-header].
 - * Meaningful address bits, e.g. [I-D.jiang-semantic-prefix]. Also, segment routing uses IPv6 addresses as segment identifiers with specific local meanings [I-D.ietf-spring-segment-routing].

All of these suggestions are only viable within a specified domain. The case of the extension header is particularly interesting, since its existence has been a major "selling point" for IPv6, but it is notorious that new extension headers are virtually impossible to deploy across the whole Internet [RFC7045], [RFC7872]. It is worth noting that extension header filtering is considered as an important security issue [I-D.ietf-opsec-ipv6-eh-filtering]. There is considerable appetite among vendors or operators to have flexibility in defining extension headers for use in limited or specialised domains, e.g. [I-D.voyer-6man-extension-header-insertion] and [BIGIP].

9. Deterministic Networking (DetNet). The Deterministic Networking Architecture [I-D.ietf-detnet-architecture] and encapsulation [I-D.ietf-detnet-dp-sol] aim to support flows with extremely low data loss rates and bounded latency, but only within a part of the network that is "DetNet aware". Thus, as for differentiated services above, the concept of a domain is fundamental.

4. Common Aspects of Limited Domains

This section derives common aspects of limited domains from the examples above.

TBD

5. The Need to Define a Limited Domain Boundary

This section justifies the need for a precise definition of a limited domain boundary and for a corresponding protocol to allow nodes to discover where such a boundary exists.

TBD

6. Defining Protocol Scope

This section suggests that protocols or protocol extensions should, when appropriate, be standardised to interoperate only within a Limited Domain Boundary. Such protocols are not required to operate across the Internet as a whole.

TBD

7. Security Considerations

Clearly, the boundary of a limited domain will almost always also act as a security boundary. In particular, it will serve as a trust boundary, and as a boundary of authority for defining capabilities. Within the boundary, limited-domain protocols or protocol features will be useful, but they will be meaningless if they enter or leave the domain.

The security model for a limited-scope protocol must allow for the boundary, and in particular for a trust model that changes at the boundary. Typically, credentials will need to be signed by a domain-specific authority.

8. IANA Considerations

This document makes no request of the IANA.

9. Acknowledgements

Useful comments were received from John Klensin, Niels ten Oever, and others.

10. Informative References

- [BIGIP] Li, R., "HUAWEI - Big IP Initiative.", 2018, <<https://www.iaria.org/announcements/HuaweiBigIP.pdf>>.
- [I-D.fioccola-v6ops-ipv6-alt-mark]
Fioccola, G., Velde, G., Cociglio, M., and P. Muley, "IPv6 Performance Measurement with Alternate Marking Method", draft-fioccola-v6ops-ipv6-alt-mark-01 (work in progress), June 2018.
- [I-D.geng-netslices-architecture]
67, 4., Dong, J., Bryant, S., kiran.makhijani@huawei.com, k., Galis, A., Foy, X., and S. Kuklinski, "Network Slicing Architecture", draft-geng-netslices-architecture-02 (work in progress), July 2017.
- [I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), June 2018.
- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-16 (work in progress), June 2018.
- [I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-06 (work in progress), February 2018.
- [I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-06 (work in progress), June 2018.
- [I-D.ietf-detnet-dp-sol]
Korhonen, J., Andersson, L., Jiang, Y., Finn, N., Varga, B., Farkas, J., Bernardos, C., Mizrahi, T., and L. Berger, "DetNet Data Plane Encapsulation", draft-ietf-detnet-dp-sol-04 (work in progress), March 2018.
- [I-D.ietf-detnet-use-cases]
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-17 (work in progress), June 2018.

- [I-D.ietf-homenet-simple-naming]
Lemon, T., Migault, D., and S. Cheshire, "Simple Homenet Naming and Service Discovery Architecture", draft-ietf-homenet-simple-naming-01 (work in progress), March 2018.
- [I-D.ietf-ipwave-vehicular-networking]
Jeong, J., "IP-based Vehicular Networking: Use Cases, Survey and Problem Statement", draft-ietf-ipwave-vehicular-networking-02 (work in progress), March 2018.
- [I-D.ietf-opsec-ipv6-eh-filtering]
Gont, F. and W. LIU, "Recommendations on the Filtering of IPv6 Packets Containing IPv6 Extension Headers", draft-ietf-opsec-ipv6-eh-filtering-05 (work in progress), March 2018.
- [I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.
- [I-D.irtf-nfvrg-gaps-network-virtualization]
Bernardos, C., Rahman, A., Zuniga, J., Contreras, L., Aranda, P., and P. Lynch, "Network Virtualization Research Challenges", draft-irtf-nfvrg-gaps-network-virtualization-09 (work in progress), February 2018.
- [I-D.jiang-semantic-prefix]
Jiang, S., Qiong, Q., Farrer, I., Bo, Y., and T. Yang, "Analysis of Semantic Embedded IPv6 Address Schemas", draft-jiang-semantic-prefix-06 (work in progress), July 2013.
- [I-D.martocci-6lowapp-building-applications]
Martocci, J., Schoofs, A., and P. Stok, "Commercial Building Applications Requirements", draft-martocci-6lowapp-building-applications-01 (work in progress), July 2010.
- [I-D.moulchan-nmrg-network-intent-concepts]
Sivakumar, K. and M. Chandramouli, "Concepts of Network Intent", draft-moulchan-nmrg-network-intent-concepts-00 (work in progress), October 2017.

- [I-D.voyer-6man-extension-header-insertion]
daniel.voyer@bell.ca, d., Leddy, J., Filsfils, C., Dukes, D., Previdi, S., and S. Matsushima, "Insertion of IPv6 Segment Routing Headers in a Controlled Domain", draft-voyer-6man-extension-header-insertion-04 (work in progress), June 2018.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC6294] Hu, Q. and B. Carpenter, "Survey of Proposed Use Cases for the IPv6 Flow Label", RFC 6294, DOI 10.17487/RFC6294, June 2011, <<https://www.rfc-editor.org/info/rfc6294>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<https://www.rfc-editor.org/info/rfc7045>>.
- [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <<https://www.rfc-editor.org/info/rfc7368>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <<https://www.rfc-editor.org/info/rfc7788>>.
- [RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <<https://www.rfc-editor.org/info/rfc7872>>.
- [RFC8151] Yong, L., Dunbar, L., Toy, M., Isaac, A., and V. Manral, "Use Cases for Data Center Network Virtualization Overlay Networks", RFC 8151, DOI 10.17487/RFC8151, May 2017, <<https://www.rfc-editor.org/info/rfc8151>>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed.,
"Network Service Header (NSH)", RFC 8300,
DOI 10.17487/RFC8300, January 2018,
<<https://www.rfc-editor.org/info/rfc8300>>.

Appendix A. Change log [RFC Editor: Please remove]

draft-carpenter-limited-domains-00, 2018-06-11:

Initial version

draft-carpenter-limited-domains-01, 2018-07-01:

Minor terminology clarifications

Authors' Addresses

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

intarea
Internet-Draft
Intended status: Standards Track
Expires: December 6, 2018

P. Pfister
E. Vyncke, Ed.
Cisco
T. Pauly
D. Schinazi
Apple
W. Shao
Telecom-ParisTech
June 4, 2018

Discovering Provisioning Domain Names and Data
draft-ietf-intarea-provisioning-domains-02

Abstract

An increasing number of hosts access the Internet via multiple interfaces or, in IPv6 multi-homed networks, via multiple IPv6 prefix configurations context.

This document describes a way for hosts to identify such contexts, called Provisioning Domains (PvDs), where Fully Qualified Domain Names (FQDNs) act as PvD identifiers. Those identifiers are advertised in a new Router Advertisement (RA) option and, when present, are associated with the set of information included within the RA.

Based on this FQDN, hosts can retrieve additional information about their network access characteristics via an HTTP over TLS query. This allows applications to select which Provisioning Domains to use as well as to provide configuration parameters to the transport layer and above.

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 December 6, 2018.

Copyright Notice

Copyright (c) 2018 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
2. Terminology	3
3. Provisioning Domain Identification using Router Advertisements	4
3.1. PvD ID Option for Router Advertisements	4
3.2. Router Behavior	7
3.3. Non-PvD-aware Host Behavior	8
3.4. PvD-aware Host Behavior	8
3.4.1. DHCPv6 configuration association	9
3.4.2. DHCPv4 configuration association	9
3.4.3. Connection Sharing by the Host	9
4. Provisioning Domain Additional Information	10
4.1. Retrieving the PvD Additional Information	10
4.2. Operational Consideration to Providing the PvD Additional Information	12
4.3. PvD Additional Information Format	12
4.3.1. Private Extensions	13
4.3.2. Example	13
4.4. Detecting misconfiguration and misuse	14
5. Operational Considerations	14
6. Security Considerations	16
7. Privacy Considerations	16
8. IANA Considerations	17
9. Acknowledgements	17
10. References	18
10.1. Normative references	18
10.2. Informative references	19
Appendix A. Changelog	20
A.1. Version 00	20

A.2.	Version 01	20
A.3.	Version 02	21
A.4.	WG Document version 00	22
A.5.	WG Document version 01	22
A.6.	WG Document version 02	23
Authors' Addresses			23

1. Introduction

It has become very common in modern networks for hosts to access the internet through different network interfaces, tunnels, or next-hop routers. To describe the set of network configurations associated with each access method, the concept of Provisioning Domain (PvD) was defined in [RFC7556].

This document specifies a way to identify PvDs with Fully Qualified Domain Names (FQDN), called PvD IDs. Those identifiers are advertised in a new Router Advertisement (RA) [RFC4861] option called the PvD ID Router Advertisement option which, when present, associates the PvD ID with all the information present in the Router Advertisement as well as any configuration object, such as addresses, deriving from it. The PVD ID Router Advertisement option may also contain a set of other RA options. Since such options are only considered by hosts implementing this specification, network operators may configure hosts that are 'PvD-aware' with PvDs that are ignored by other hosts.

Since PvD IDs are used to identify different ways to access the internet, multiple PvDs (with different PvD IDs) could be provisioned on a single host interface. Similarly, the same PvD ID could be used on different interfaces of a host in order to inform that those PvDs ultimately provide identical services.

This document also introduces a way for hosts to retrieve additional information related to a specific PvD by means of an HTTP over TLS query using an URI derived from the PvD ID. The retrieved JSON object contains additional information that would typically be considered unfit, or too large, to be directly included in the Router Advertisement, but might be considered useful to the applications, or even sometimes users, when choosing which PvD should be used.

2. Terminology

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 [RFC2119].

In addition, this document uses the following terminology:

Provisioning Domain (PvD): A set of network configuration information; for more information, see [RFC7556].

PvD ID: A Fully Qualified Domain Name (FQDN) used to identify a PvD.

Explicit PvD: A PvD uniquely identified with a PvD ID. For more information, see [RFC7556].

Implicit PvD: A PvD that, in the absence of a PvD ID, is identified by the host interface to which it is attached and the address of the advertising router. See also [RFC7556].

PvD-aware host A host that supports the association of network configuration information into PvDs and the use of these PvDs. Also named PvD-aware node in [RFC7556].

3. Provisioning Domain Identification using Router Advertisements

Explicit PvDs are identified by a PvD ID. The PvD ID is a Fully Qualified Domain Name (FQDN) which MUST belong to the network operator in order to avoid naming collisions. The same PvD ID MAY be used in several access networks when they ultimately provide identical services (e.g., in all home networks subscribed to the same service); else, the PvD ID MUST be different to follow section 2.4 of [RFC7556].

3.1. PvD ID Option for Router Advertisements

This document introduces a Router Advertisement (RA) option called PvD option. It is used to convey the FQDN identifying a given PvD (see Figure 1), bind the PvD ID with configuration information received over DHCPv4 (see Section 3.4.2), enable the use of HTTP over TLS to retrieve the PvD Additional Information JSON object (see Section 4), as well as contain any other RA options which would otherwise be valid in the RA.

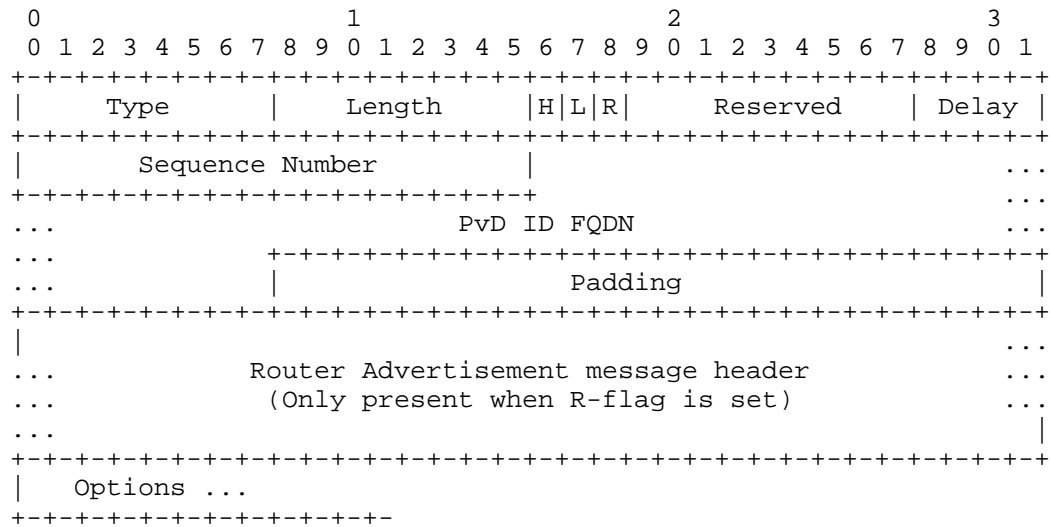


Figure 1: PvD ID Router Advertisements Option format

Type : (8 bits) Set to 21.

Length : (8 bits) The length of the option in units of 8 octets, including the Type and Length fields, the Router Advertisement message header, if any, as well as the RA options that are included within the PvD Option.

H-flag : (1 bit) 'HTTP' flag stating whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 4.

L-flag : (1 bit) 'Legacy' flag stating whether the router is also providing IPv4 information using DHCPv4 (see Section 3.4.2).

R-flag : (1 bit) 'Router Advertisement' flag stating whether the PvD Option is followed (right after padding to the next 64 bits boundary) by a Router Advertisement message header (See section 4.2 of [RFC4861]).

Delay : (4 bits) Unsigned integer used to delay HTTP GET queries from hosts by a randomized backoff (see Section 4.1).

Reserved : (13 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

Sequence Number: (16 bits) Sequence number for the PvD Additional Information, as described in Section 4.

PvD ID FQDN : The FQDN used as PvD ID encoded in DNS format, as described in Section 3.1 of [RFC1035]. Domain names compression described in Section 4.1.4 of [RFC1035] MUST NOT be used.

Padding : Zero or more padding octets to the next 8 octets boundary. It MUST be set to zero by the sender, and ignored by the receiver.

RA message header : (16 octets) When the R-flag is set, a full Router Advertisement message header as specified in [RFC4861]. The 'Type', 'Code' and 'Checksum' fields (i.e. the first 32 bits), MUST be set to zero by the sender and ignored by the receiver. The other fields are to be set and parsed as specified in [RFC4861] or any updating documents.

Options : Zero or more RA options that would otherwise be valid as part of the Router Advertisement main body, but are instead included in the PvD Option such as to be ignored by hosts that are not 'PvD-aware'.

Here is an example of a PvD option with example.org as the PvD ID FQDN and including a RDNSS and prefix information options (it also have the sequence number 123, presence of additional information to be fetched with a delay indicated as 5):

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																		
Type: 21										Length: 12										1 0 0										Reserved										Delay:5									
Seq number: 123										7										e																													
x										a										m										p																			
l										e										3										o																			
r										g										0										0 (padding)																			
0 (padding)										0 (padding)										0 (padding)										0 (padding)																			
RDNSS option (RFC 6106) length: 5																														...																			
...																														...																			
...																																																	
Prefix Information Option (RFC 4861) length: 4																														...																			
...																																																	
...																																																	

Figure 2

3.2. Router Behavior

A router MAY send RAs containing one PvD option, but MUST NOT include more than one PvD option in each RA. In particular, the PvD option MUST NOT contain further PvD options.

The PvD Option MAY contain zero, one, or more RA options which would otherwise be valid as part of the same RA. Such options are processed by PvD-aware hosts, while ignored by others.

In order to provide multiple different PvDs, a router MUST send multiple RAs. Different explicit PvDs MAY be advertised with RAs using the same IPv6 source address; but different implicit PvDs, advertised by different RAs, MUST use different link-local addresses because these implicit PvDs are identified by the source addresses of the RAs.

Whenever an RA, for a single PvD, would need to be sent via multiple packets, the PvD option header (i.e., all fields except the 'Options' field) MUST be repeated in all the transmitted RAs. But the options

within the 'Options' field, MAY be transmitted only once, included in one of the transmitted PvD options.

3.3. Non-PvD-aware Host Behavior

As the PvD Option has a new option code, non-PvD-aware hosts will simply ignore the PvD Option and all the options it contains. This ensures the backward compatibility required in section 3.3 of [RFC7556]. This behavior allows for a mixed-mode network with a mix of PvD-aware and non-PvD-aware hosts coexist.

3.4. PvD-aware Host Behavior

Hosts MUST associate received RAs and included configuration information (e.g., Router Valid Lifetime, Prefix Information [RFC4861], Recursive DNS Server [RFC8106], Routing Information [RFC4191] options) with the explicit PvD identified by the first PvD Option present in the received RA, if any, or with the implicit PvD identified by the host interface and the source address of the received RA otherwise.

In case multiple PvD options are found in a given RA, hosts MUST ignore all but the first PvD option.

Similarly, hosts MUST associate all network configuration objects (e.g., default routers, addresses, more specific routes, DNS Recursive Resolvers) with the PvD associated with the RA which last updated the object. For example, addresses that are generated using a received Prefix Information option (PIO) are associated with the PvD of the last received RA which included the given PIO.

PvD IDs MUST be compared in a case-insensitive manner (i.e., A=a), assuming ASCII with zero parity while non-alphabetic codes must match exactly (see also Section 3.1 of [RFC1035]). For example, "pvd.example.com." or "PvD.Example.coM." would refer to the same PvD.

While resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm when forwarding packets ([RFC2461], [RFC4191] and [RFC8028]), hosts MAY consider only the configuration associated with an arbitrary set of PvDs.

For example, a host MAY associate a given process with a specific PvD, or a specific set of PvDs, while associating another process with another PvD. A PvD-aware application might also be able to select, on a per-connection basis, which PvDs should be used. In particular, constrained devices such as small battery operated devices (e.g. IoT), or devices with limited CPU or memory resources

may purposefully use a single PvD while ignoring some received RAs containing different PvD IDs.

The way an application expresses its desire to use a given PvD, or a set of PvDs, or the way this selection is enforced, is out of the scope of this document. Useful insights about these considerations can be found in [I-D.kline-mif-mpvd-api-reqs].

3.4.1. DHCPv6 configuration association

When a host retrieves configuration elements using DHCPv6 (e.g., addresses or DNS recursive resolvers), they MUST be associated with the explicit or implicit PvD of the RA received on the same interface, sent from the same LLA, and with the O-flag or M-flag set [RFC4861]. If no such PvD is found, or whenever multiple different PvDs are found, the host behavior is unspecified.

This process requires hosts to keep track of received RAs, associated PvD IDs, and routers LLA; it also assumes that the router either acts as a DHCPv6 server or relay and uses the same LLA for DHCPv6 and RA traffic (which may not be the case when the router uses VRRP to send its RA).

3.4.2. DHCPv4 configuration association

When a host retrieves configuration elements from DHCPv4, they MUST be associated with the explicit PvD received on the same interface, whose PVD Options L-flag is set and, in the case of a non point-to-point link, using the same datalink address. If no such PvD is found, or whenever multiple different PvDs are found, the configuration elements coming from DHCPv4 MUST be associated with the implicit PvD identified by the interface on which the DHCPv4 transaction happened. The case of multiple explicit PvD for an IPv4 interface is undefined.

3.4.3. Connection Sharing by the Host

The situation when a node receives an RA on one interface (e.g. cellular) and shares this connectivity by also acting as a router by transmitting RA on another interface (e.g. WiFi) is known as 'tethering'. It can be done as ND proxy. The exact behavior is out of scope of this document but it is expected that the one or several PvD associated to the shared interface (e.g. cellular) will also be advertised to the clients on the other interface (e.g. WiFi).

4. Provisioning Domain Additional Information

Additional information about the network characteristics can be retrieved based on the PvD ID. This set of information is called PvD Additional Information, and is encoded as a JSON object [RFC7159].

The purpose of this additional set of information is to securely provide additional information to applications about the connectivity that is provided using a given interface and source address pair. It typically includes data that would be considered too large, or not critical enough, to be provided within an RA option. The information contained in this object MAY be used by the operating system, network libraries, applications, or users, in order to decide which set of PvDs should be used for which connection, as described in Section 3.4.

4.1. Retrieving the PvD Additional Information

When the H-flag of the PvD Option is set, hosts MAY attempt to retrieve the PvD Additional Information associated with a given PvD by performing an HTTP over TLS [RFC2818] GET query to `https://<PvD-ID>/.well-known/pvd` [RFC5785]. Inversely, hosts MUST NOT do so whenever the H-flag is not set.

Note that the DNS name resolution of the PvD ID, the PKI checks as well as the actual query MUST be performed using the considered PvD. In other words, the name resolution, PKI checks, source address selection, as well as the next-hop router selection MUST be performed while using exclusively the set of configuration information attached with the PvD, as defined in Section 3.4. In some cases, it may therefore be necessary to wait for an address to be available for use (e.g., once the Duplicate Address Detection or DHCPv6 processes are complete) before initiating the HTTP over TLS query. If the host has a temporary address per [RFC4941] in this PvD, then hosts SHOULD use a temporary address to fetch the PvD Additional Information and SHOULD deprecate the used temporary address and generate a new temporary address afterward.

If the HTTP status of the answer is greater than or equal to 400 the host MUST abandon and consider that there is no additional PvD information. If the HTTP status of the answer is between 300 and 399, inclusive, it MUST follow the redirection(s). If the HTTP status of the answer is between 200 and 299, inclusive, the host MAY get a file containing a single JSON object. When a JSON object could not be retrieved, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

After retrieval of the PvD Additional Information, hosts MUST keep track of the Sequence Number value received in subsequent RAs including the same PvD ID. In case the new value is greater than the value that was observed when the PvD Additional Information object was retrieved (using serial number arithmetic comparisons [RFC1982]), or whenever the validity time included in the PVD Additional Information JSON object is expired, hosts MUST either perform a new query and retrieve a new version of the object, or, failing that, deprecate the object and stop using the additional information provided in the JSON object.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields specified in Section 4.3. A retrieved object including an expiration time that is already past or missing a mandatory element MUST be ignored.

In order to avoid synchronized queries toward the server hosting the PvD Additional Information when an object expires, object updates are delayed by a randomized backoff time.

When a host performs an object update after it detected a change in the PvD Option Sequence number, it MUST delay the query by a random time between zero and $2^{**}(\text{Delay} * 2)$ milliseconds, where 'Delay' corresponds to the 4 bits long unsigned integer in the last received PvD Option.

When a host last retrieved an object at time A including a validity time B, and is configured to keep the object up to date, it MUST perform the update at a uniformly random time in the interval $[(B-A)/2, B]$.

In the example Figure 2, the delay field value is 5, this means that host MUST delay the query by a random number between 0 and $2^{**}(5 * 2)$ milliseconds, i.e., between 0 and 1024 milliseconds.

Since the 'Delay' value is directly within the PvD Option rather than the object itself, an operator may perform a push-based update by incrementing the Sequence value while changing the Delay value depending on the criticality of the update and its PvD Additional Information servers capacity.

The PvD Additional Information object includes a set of IPv6 prefixes (under the key "prefixes") which MUST be checked against all the Prefix Information Options advertised in the RA. If any of the prefixes included in the PIO is not covered by at least one of the listed prefixes, the PvD associated with the tested prefix MUST be considered unsafe and MUST NOT be used. While this does not prevent

a malicious network provider, it does complicate some attack scenarios, and may help detecting misconfiguration.

4.2. Operational Consideration to Providing the PVD Additional Information

Whenever the H-flag is set in the PVD Option, a valid PVD Additional Information object MUST be made available to all hosts receiving the RA by the network operator. In particular, when a captive portal is present, hosts MUST still be allowed to perform DNS, PKI and HTTP over TLS operations related to the retrieval of the object, even before logging into the captive portal.

Routers MAY increment the PVD Option Sequence number in order to inform host that a new PVD Additional Information object is available and should be retrieved.

The server providing the JSON files SHOULD also check whether the client address is part of the prefixes listed into the additional information and SHOULD return a 403 responsecode if there is no match.

4.3. PVD Additional Information Format

The PVD Additional Information is a JSON object.

The following table presents the mandatory keys which MUST be included in the object:

JSON key	Description	Type	Example
name	Human-readable service name	UTF-8 string [RFC3629]	"Awesome Wifi"
expires	Date after which this object is not valid	[RFC3339]	"2017-07-23T06:00:00Z"
prefixes	Array of IPv6 prefixes valid for this PVD	Array of strings	["2001:db8:1::/48", "2001:db8:4::/48"]

A retrieved object which does not include a valid string associated with the "name" key at the root of the object, or a valid date associated with the "expires" key, also at the root of the object, MUST be ignored. In such cases, an error message SHOULD be logged

and/or displayed in a rate-limited fashion. If the PIO of the received RA is not covered by at least one of the "prefixes" key, the retrieved object SHOULD be ignored.

The following table presents some optional keys which MAY be included in the object.

JSON key	Description	Type	Example
localizedName	Localized user-visible service name, language can be selected based on the HTTP Accept-Language header in the request.	UTF-8 string	"Wifi Genial"
dnsZones	DNS zones searchable and accessible	array of DNS zones	["example.com", "sub.example.org"]
noInternet	No Internet, set when the PvD only provides restricted access to a set of services	boolean	true

It is worth noting that the JSON format allows for extensions. Whenever an unknown key is encountered, it MUST be ignored along with its associated elements.

4.3.1. Private Extensions

JSON keys starting with "x-" are reserved for private use and can be utilized to provide information that is specific to vendor, user or enterprise. It is RECOMMENDED to use one of the patterns "x-FQDN-KEY" or "x-PEN-KEY" where FQDN is a fully qualified domain name or PEN is a private enterprise number [PEN] under control of the author of the extension to avoid collisions.

4.3.2. Example

Here are two examples based on the keys defined in this section.


```
{
  "name": "Foo Wireless",
  "localizedName": "Foo-France Wifi",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes" : ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "name": "Bar 4G",
  "localizedName": "Bar US 4G",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
}
```

4.4. Detecting misconfiguration and misuse

When a host retrieves the PvD Additional Information, it MUST verify that the TLS server certificate is valid for the performed request (e.g., that the Subject Name is equal to the PvD ID expressed as an FQDN). This authentication creates a secure binding between the information provided by the trusted Router Advertisement, and the HTTPS server. But this does not mean the Advertising Router and the PvD server belong to the same entity.

Hosts MUST verify that all prefixes in the RA PIO are covered by a prefix from the PvD Additional Information. An adversarial router willing to fake the use of a given explicit PvD, without any access to the actual PvD Additional Information, would need to perform NAT66 in order to circumvent this check.

It is also RECOMMENDED that the HTTPS server checks the source addresses of incoming connections (see Section 4.1). This check give reasonable assurance that neither NPTv6 [RFC6296] nor NAT66 were used and restricts the information to the valid network users.

5. Operational Considerations

This section describes some use cases of PvD. For the sake of simplicity, the RA messages will not be described in the usual ASCII art but rather in an indented list. For example, a RA message containing some options and a PvD option that also contains other options will be described as:

- o RA Header: router lifetime = 6000
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64

- o PvD Option header: length = 3+ 5 +4 , PvD ID FQDN = example.org., R-flag = 0 (actual length of the header with padding 24 bytes = 3 * 8 bytes)
- * Recursive DNS Server: length = 5, addresses= [2001:db8:cafe::53, 2001:db8:f00d::53]
- * Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64

It is expected that for some years, networks will have a mixed environment of PvD-aware hosts and non-PvD-aware hosts. If there is a need to give specific information to PvD-aware hosts only, then it is recommended to send TWO RA messages: one for each class of hosts. For example, here is the RA for non-PvD-aware hosts:

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses= [2001:db8:cafe::53]
- o PvD Option header: length = 3+ 2, PvD ID FQDN = foo.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 * 8 bytes)
- * RA Header: router lifetime = 0 (PvD-aware hosts will not use this router as a default router), implicit length = 2

And here is a RA example for PvD-aware hosts:

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3+ 2 + 4 + 3, PvD ID FQDN = example.org., R-flag = 1 (actual length of the header 24 bytes = 3 * 8 bytes)
- * RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
- * Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
- * Recursive DNS Server Option: length = 3, addresses= [2001:db8:f00d::53]

In the above example, non-PvD-aware hosts will only use the first RA sent from their default router and using the 2001:db8:cafe::/64 prefix. PvD-aware hosts will autonomously configure addresses from both PIOs, but will only use the source address in 2001:db8:f00d::/64 to communicate past the first hop router since only the router sending the second RA will be used as default router; similarly, they will use the DNS server 2001:db8:f00d::53 when communicating with this address.

6. Security Considerations

Although some solutions such as IPsec or SeND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, in practice actual deployments largely rely on link layer or physical layer security mechanisms (e.g. 802.1x [IEEE8021X]) in conjunction with RA Guard [RFC6105].

This specification does not improve the Neighbor Discovery Protocol security model, but extends the purely link-local trust relationship between the host and the default routers with HTTP over TLS communications which servers are authenticated as rightful owners of the FQDN received within the trusted PvD ID RA option.

It must be noted that Section 4.4 of this document only provides reasonable assurance against misconfiguration but does not prevent an hostile network access provider to advertize wrong information that could lead applications or hosts to select an hostile PvD. Users should always apply caution when connecting to an unknown network.

7. Privacy Considerations

Retrieval of the PvD Additional Information over HTTPS requires early communications between the connecting host and a server which may be located further than the first hop router. Although this server is likely to be located within the same administrative domain as the default router, this property can't be ensured. Therefore, hosts willing to retrieve the PvD Additional Information before using it without leaking identity information, SHOULD make use of an IPv6 Privacy Address and SHOULD NOT include any privacy sensitive data, such as User Agent header or HTTP cookie, while performing the HTTP over TLS query.

From a privacy perspective, retrieving the PvD Additional Information is not different from establishing a first connection to a remote server, or even performing a single DNS lookup. For example, most operating systems already perform early queries to well known web sites, such as <http://captive.example.com/hotspot-detect.html>, in order to detect the presence of a captive portal.

There may be some cases where hosts, for privacy reasons, should refrain from accessing servers that are located outside a certain network boundary. In practice, this could be implemented as a whitelist of 'trusted' FQDNs and/or IP prefixes that the host is allowed to communicate with. In such scenarios, the host SHOULD check that the provided PvD ID, as well as the IP address that it resolves into, are part of the allowed whitelist.

8. IANA Considerations

Upon publication of this document, IANA is asked to remove the 'reclaimable' tag off the value 21 for the PvD option (from the IPv6 Neighbor Discovery Option Formats registry).

IANA is asked to assign the value "pvd" from the Well-Known URIs registry.

IANA is asked to create and maintain a new registry entitled "Additional Information PvD Keys" containing ASCII strings. The initial content of this registry are given in Section 4.3; future assignments are to be made through Expert Review [BCP36].

Finally, IANA is asked to create and maintain a new registry entitled "PvD option Flags" reserving bit positions from 0 to 15 to be used in the PvD option bitmask. Bit position 0, 1 and 2 are reserved by this document (as specified in Figure 1). Future assignments require a Standard Track RFC document.

9. Acknowledgements

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns], as well as to Basile Bruneau who was author of an early version of this document.

Thanks also to Marcus Keane, Mikael Abrahamson, Ray Bellis, Zhen Cao, Tim Chow, Lorenzo Colitti, Ian Farrer, Bob Hinden, Tatuya Jinmei, Erik Kline, Ted Lemon, Jen Lenkova, Veronika McKillop, Mark Townsley and James Woodyatt for useful and interesting discussions.

Finally, special thanks to Thierry Danis and Wenqin Shao for their valuable inputs and implementation efforts ([github]), Tom Jones for his integration effort into the NEAT project and Rigil Salim for his implementation work.

10. References

10.1. Normative references

- [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>>.
- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [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>>.
- [RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, DOI 10.17487/RFC2461, December 1998, <<https://www.rfc-editor.org/info/rfc2461>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [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>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [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>>.

10.2. Informative references

- [github] Cisco, "IPv6-mPvD github repository",
<<https://github.com/IPv6-mPvD>>.
- [I-D.kline-mif-mpvd-api-reqs]
Kline, E., "Multiple Provisioning Domains API
Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in
progress), November 2015.
- [I-D.stenberg-mif-mpvd-dns]
Stenberg, M. and S. Barth, "Multiple Provisioning Domains
using Domain Name System", draft-stenberg-mif-mpvd-dns-00
(work in progress), October 2015.
- [IEEE8021X]
IEEE, "IEEE Standards for Local and Metropolitan Area
Networks: Port based Network Access Control, IEEE Std".
- [PEN] IANA, "Private Enterprise Numbers",
<<https://www.iana.org/assignments/enterprise-numbers>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet:
Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002,
<<https://www.rfc-editor.org/info/rfc3339>>.
- [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>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy
Extensions for Stateless Address Autoconfiguration in
IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007,
<<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
Uniform Resource Identifiers (URIs)", RFC 5785,
DOI 10.17487/RFC5785, April 2010,
<<https://www.rfc-editor.org/info/rfc5785>>.

- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6731] Savolainen, T., Kato, J., and T. Lemon, "Improved Recursive DNS Server Selection for Multi-Interfaced Nodes", RFC 6731, DOI 10.17487/RFC6731, December 2012, <<https://www.rfc-editor.org/info/rfc6731>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [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>>.
- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.

Appendix A. Changelog

Note to RFC Editors: Remove this section before publication.

A.1. Version 00

Initial version of the draft. Edited by Basile Bruneau + Eric Vyncke and based on Basile's work.

A.2. Version 01

Major rewrite intended to focus on the the retained solution based on corridors, online, and WG discussions. Edited by Pierre Pfister. The following list only includes major changes.

PvD ID is an FQDN retrieved using a single RA option. This option contains a sequence number for push-based updates, a new H-flag, and a L-flag in order to link the PvD with the IPv4 DHCP server.

A lifetime is included in the PvD ID option.

Detailed Hosts and Routers specifications.

Additional Information is retrieved using HTTP-over-TLS when the PvD ID Option H-flag is set. Retrieving the object is optional.

The PvD Additional Information object includes a validity date.

DNS-based approach is removed as well as the DNS-based encoding of the PvD Additional Information.

Major cut in the list of proposed JSON keys. This document may be extended later if need be.

Monetary discussion is moved to the appendix.

Clarification about the 'prefixes' contained in the additional information.

Clarification about the processing of DHCPv6.

A.3. Version 02

The FQDN is now encoded with ASCII format (instead of DNS binary) in the RA option.

The PvD ID option lifetime is removed from the object.

Use well known URI "https://<PvD-ID>/.well-known/pvd"

Reference RFC3339 for JSON timestamp format.

The PvD ID Sequence field has been extended to 16 bits.

Modified host behavior for DHCPv4 and DHCPv6.

Removed IKEv2 section.

Removed mention of RFC7710 Captive Portal option. A new I.D. will be proposed to address the captive portal use case.

A.4. WG Document version 00

Document has been accepted as INTAREA working group document

IANA considerations follow RFC8126 [RFC8126]

PvD ID FQDN is encoded as per RFC 1035 [RFC1035]

PvD ID FQDN is prepended by a one-byte length field

Marcus Keane added as co-author

dnsZones key is added back

draft of a privacy consideration section and added that a temporary address should be used to retrieve the PvD additional information

per Bob Hinden's request: the document is now aiming at standard track and security considerations have been moved to the main section

A.5. WG Document version 01

Removing references to 'metered' and 'characteristics' keys. Those may be in scope of the PvD work, but this document will focus on essential parts only.

Removing appendix section regarding link quality and billing information.

The PvD RA Option may now contain other RA options such that PvD-aware hosts may receive configuration information otherwise invisible to non-PvD-aware hosts.

Clarify that the additional PvD Additional Information is not intended to modify host's networking stack behavior, but rather provide information to the Application, used to select which PvDs must be used and provide configuration parameters to the transport layer.

The RA option padding is used to increase the option size to the next 64 (was 32) bits boundary.

Better detail the Security model and Privacy considerations.

A.6. WG Document version 02

Use the IANA value of 21 in the text and update the IANA considerations section accordingly

add the Delay field to avoid the thundering herd effect

add Wenqin Shao as author

keep the 1 PvD per RA model

changed the intro (per Zhen Cao) "when choosing which PvD and transport should be used" => "when choosing which PvD should be used"

rename A-flag in R-flag to avoid A-flag of PIO

use the wording "PvD Option", removing the ID token as it is now a container with more than just an ID, removing 'RA' in the option name to be consistent with other IANA NDP option

use "non-PvD-aware" rather than "PvD-ignorant"

added more reference to RFC 7556 (notably for PvD being globally unique, introducing PvD-aware host vs. PvD-aware node)

Section 3.4.3 renamed from "interconnection shared by node" to 'connection shared by node"

Section 3.4 renamed into "PvD-aware Host Behavior"

Added a section "Non-PvD-aware Host Behavior"

Authors' Addresses

Pierre Pfister
Cisco
11 Rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: ppfister@cisco.com

Eric Vyncke (editor)
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

Tommy Pauly
Apple

Email: tpauly@apple.com

David Schinazi
Apple

Email: dschinazi@apple.com

Wenqin Shao
Telecom-ParisTech
France

Email: wenqin.shao@telecom-paristech.fr

INTAREA
Internet-Draft
Intended status: Informational
Expires: December 2, 2019

S. Kanugovi
Nokia
F. Baboescu
Broadcom
J. Zhu
Intel
J. Mueller
AT&T
S. Seo
Korea Telecom
May 31, 2019

Multiple Access Management Services
draft-kanugovi-intarea-mams-framework-04

Abstract

In multiconnectivity scenarios, the end-user devices can simultaneously connect to multiple networks based on different access technologies and network architectures like WiFi, LTE, DSL. Both the quality of experience of the users and the overall network utilization and efficiency may be improved through the smart selection and combination of access and core network paths that can dynamically adapt to changing network conditions. This document presents a unified problem statement and introduces a solution for managing multiconnectivity. The solution has been developed by the authors based on their experiences in multiple standards bodies including the IETF and 3GPP, but is not an Internet Standard and does not represent the consensus opinion of the IETF. This document describes the requirements, solution principles, and an architectural framework that aims to provide best performance while being easy to implement in a wide variety of multiconnectivity deployments. It specifies the protocol multi-access management to: 1) flexibly select the best combination of access and core network paths for uplink and downlink; as well as 2) determine the user plane treatment and traffic distribution over the selected links ensuring network efficiency and application performance.

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 December 2, 2019.

Copyright Notice

Copyright (c) 2019 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	6
2. Terminology	7
3. Problem Statement	8
4. Requirements	9
4.1. Access Technology Agnostic Interworking	9
4.2. Support Common Transport Deployments	9
4.3. Independent Access Path Selection for Uplink and Downlink	9
4.4. Core Selection Independent of Uplink and Downlink Access	9
4.5. Adaptive Access Network Path Selection	9
4.6. Multipath Support and Aggregation of Access Link Capacities	10
4.7. Scalable Mechanism based on User Plane Interworking	10
4.8. Separate Control and Data Plane functions	10
4.9. Lossless Path (Connection) Switching	10
4.10. Concatenation and Fragmentation for adaptation to MTU Differences	11
4.11. Configuring Network Middleboxes based on Negotiated Protocols	11
4.12. Policy based Optimal Path Selection	11
4.13. Access Technology Agnostic Control Signaling	11
4.14. Service Discovery and Reachability	11
5. Solution Principles	12
6. MAMS Reference Architecture	12
7. MAMS Protocol Architecture	15

7.1.	MAMS Control-Plane Protocol	15
7.2.	MAMS User Plane Protocol	16
8.	MAMS Control Plane Procedures	18
8.1.	Overview	18
8.2.	Common fields in MAMS Control Messages	20
8.3.	Common Procedures for MAMS Control Messages	20
8.3.1.	Message Timeout	20
8.3.2.	Keep Alive Procedure	20
8.4.	Discovery & Capability Exchange	21
8.5.	User Plane Configuration	25
8.6.	MAMS Path Quality Estimation	29
8.6.1.	MX Control PDU definition	31
8.6.2.	Keep-Alive Message	32
8.6.3.	Probe REQ/ACK Message	32
8.7.	MAMS Traffic Steering	33
8.8.	MAMS Application MADP Association	34
8.9.	MAMS Network ID Indication	35
8.10.	MAMS Client Measurement Configuration and Reporting	36
8.11.	MAMS Session Termination Procedure	38
8.12.	MAMS Network Analytics Request Procedure	39
9.	Generic MAMS Signaling Flow	41
10.	Relation to IETF Technologies	43
11.	Applying MAMS Control Procedures with MPTCP Proxy as User Plane	43
12.	Applying MAMS Control Procedures for Network Assisted Traffic Steering when there is No Convergence Layer	49
13.	Co-existence of MX Adaptation and MX Convergence Layers	51
14.	Security Considerations	51
14.1.	MAMS Control Plane Security	51
14.2.	MAMS User Plane Security	52
15.	Implementation Considerations	52
16.	Applicability to Multi Access Edge Computing	52
17.	Related work in other Industry and Standards Forums	53
18.	Contributing Authors	53
19.	Acknowledgments	54
20.	IANA Considerations	54
21.	References	54
21.1.	Normative References	54
21.2.	Informative References	54
Appendix A.	MAMS Control Plane Optimization over Secure Connections	56
Appendix B.	MAMS Application Interface	57
B.1.	Overall Design	57
B.2.	Notation	57
B.3.	Error Indication	57
B.4.	CCM APIs	57
B.4.1.	Get Capabilities	57
B.4.2.	Post App Requirements	58

B.4.3. Get Predictive Link Parameters	59
Appendix C. JSON Specification for MAMS Control Plane	60
C.1. Protocol Specification: General Processing	60
C.1.1. Notation	60
C.1.2. Discovery Procedure	61
C.1.3. System Information Procedure	61
C.1.4. Capability Exchange Procedure	62
C.1.5. User Plane Configuration Procedure	63
C.1.6. Reconfiguration Procedure	65
C.1.7. Path Estimation Procedure	66
C.1.8. Traffic Steering Procedure	67
C.1.9. MAMS Application MADP Association	68
C.1.10. SSID Indication	69
C.1.11. Measurements	70
C.1.12. Keep Alive	71
C.1.13. Session Termination Procedure	72
C.1.14. Network Analytics	73
C.2. Protocol Specification: Data Types	74
C.2.1. MXBase	74
C.2.2. Unique Session Id	75
C.2.3. NCM Connections	76
C.2.4. Connection Information	76
C.2.5. Features Activation Status	77
C.2.6. Anchor Connections	77
C.2.7. Delivery Connections	78
C.2.8. Method Support	78
C.2.9. Convergence Methods	78
C.2.10. Adaptation Methods	79
C.2.11. Setup of Anchor Connections	79
C.2.12. Init Probe Results	81
C.2.13. Active Probe Results	82
C.2.14. Downlink Delivery	82
C.2.15. Uplink Delivery	82
C.2.16. Traffic Flow Template	83
C.2.17. Measurement Report Configuration	83
C.2.18. Measurement Report	84
C.3. Schemas in JSON	85
C.3.1. MX Base Schema	85
C.3.2. MX Definitions	86
C.3.3. MX Discover	93
C.3.4. MX System Update	93
C.3.5. MX Capability Request	94
C.3.6. MX Capability Response	95
C.3.7. MX Capability Ack	96
C.3.8. MX Reconfiguration Request	97
C.3.9. MX Reconfiguration Response	98
C.3.10. MX UP Setup Configuration	99
C.3.11. MX UP Setup Confirmation	100

C.3.12.	MX Traffic Steering Request	101
C.3.13.	MX Traffic Steering Response	101
C.3.14.	MX Application MADP Association Request	102
C.3.15.	MX Application MADP Association Response	103
C.3.16.	MX Path Estimation Request	103
C.3.17.	MX Path Estimation Report	104
C.3.18.	MX SSID Indication	105
C.3.19.	MX Measurements Configuration	106
C.3.20.	MX Measurements Report	107
C.3.21.	MX Keep Alive Request	109
C.3.22.	MX Keep Alive Response	109
C.3.23.	MX Session Termination Request	109
C.3.24.	MX Session Termination Response	110
C.3.25.	MX Network Analytics Request	110
C.3.26.	MX Network Analytics Response	111
C.4.	Examples in JSON	112
C.4.1.	MX Discover	112
C.4.2.	MX System Update	112
C.4.3.	MX Capability Request	113
C.4.4.	MX Capability Response	115
C.4.5.	MX Capability Ack	116
C.4.6.	MX Reconfiguration Request	116
C.4.7.	MX Reconfiguration Response	117
C.4.8.	MX UP Setup Configuration Request	117
C.4.9.	MX UP Setup Confirmation	119
C.4.10.	MX Traffic Steering Request	119
C.4.11.	MX Traffic Steering Response	121
C.4.12.	MX Application MADP Association Request	121
C.4.13.	MX Application MADP Association Response	122
C.4.14.	MX Path Estimation Request	122
C.4.15.	MX Path Estimation Results	123
C.4.16.	MX SSID Indication	123
C.4.17.	MX Measurements Configuration	124
C.4.18.	MX Measurements Report	125
C.4.19.	MX Keep Alive Request	127
C.4.20.	MX Keep Alive Response	127
C.4.21.	MX Session Termination Request	127
C.4.22.	MX Session Termination Response	127
C.4.23.	MX Network Analytics Request	128
C.4.24.	MX Network Analytics Response	128
Appendix D.	Definition of APIs provided by CCM to the Applications at the Client	129
Appendix E.	Implementation Example using Python for MAMS Client and Server	137
E.1.	Client Side Implementation	137
E.2.	Server Side Implementation	139
Authors' Addresses	141

1. Introduction

Multi Access Management Services (MAMS) is a programmable framework that provides mechanisms for flexible selection of network paths in a multi-access communication environment, based on application needs. It leverages network intelligence and policies to dynamically adapt traffic distribution across selected paths and user plane treatment to changing network/link conditions. The network path selection and configuration messages are carried as user plane data between the functional elements in the network and the end-user device, and thus without any impact to the control plane signaling schemes of the underlying access network(s). For example, in a multi-access network with LTE and WiFi technologies, existing LTE and existing WiFi signaling procedures will be used to setup the LTE and WiFi connections, respectively, and MAMS specific control plane messages are carried as LTE or WiFi user plane data. The MAMS framework defined in this document provides the capabilities of smart selection and flexible combination of access paths and core network paths, as well as the user plane treatment when the traffic is distributed across the selected paths. Thus, it is a broad programmable framework providing functions beyond simple sharing of network policies such as provided by Access Network Discovery and Selection Function (ANDSF) [ANDSF] that offers policies and rules for assisting 3GPP devices to discover and select available access networks. Further, it allows the choice and configuration of user plane treatment for the traffic over the multiple paths, depending on the needs of the application.

MAMS mechanisms are not dependent on any specific access network type or user plane protocols like TCP, UDP, GRE, MPTCP etc. It co-exists and complements the existing protocols by providing a way to negotiate and configure these protocols based on client and network capabilities per access basis to match their use for a given multi-access scenario. Further, it allows load balancing of the traffic flows across the selected multiple accesses and exchange of network state information to be used for network intelligence to optimize the performance of such protocols.

The document presents the requirements, solution principles, functional architecture, and protocols for realizing the MAMS framework. An important goal for MAMS is to ensure that it either requires minimum dependency or (better) no dependency on the actual access technologies of the participating links, beyond the fact that MAMS functional elements form an IP-overlay across the multiple paths. This allows the scheme to be future proof by allowing independent technology evolution of the existing access and core networks as well as, seamless integration of new access technologies.

The solution described in this document has been developed by the authors based on their experiences in multiple standards bodies including the IETF and 3GPP, but is not an Internet Standard and does not represent the consensus opinion of the IETF.

2. Terminology

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.

"Client": The end-user device supporting connections with multiple access nodes, possibly over different access technologies.

"Multiconnectivity Client": A client with multiple network connections.

"Access network": The segment in the network that delivers user data packets to the client via an access link like WiFi airlink, LTE airlink, or DSL.

"Core": The functional element that anchors the client IP address used for communication with applications via the network.

"Network Connection manager" (NCM): A functional entity in the network that handles MAMS control messages from the client and configures distribution of data packets over the multiple available access and core network paths, and user plane treatment of the traffic flows.

"Client Connection Manager" (CCM): A functional entity in the client that exchanges MAMS Signaling with the Network Connection Manager and configures the multiple network paths at the client for transport of user data.

"Network Multi Access Data Proxy" (N-MADP): This functional entity in the network handles the user data traffic forwarding across multiple network paths. N-MADP is responsible for MAMS related user-plane functionalities in the network.

"Client Multi Access Data Proxy" (C-MADP): This functional entity in the client handles the user data traffic forwarding across multiple network paths. C-MADP is responsible for MAMS related user-plane functionalities in the client.

"Anchor Connection": Refers to the network path from the N-MADP to the user plane gateway (IP anchor) that has assigned an IP address to the client.

"Delivery Connection": Refers to the network path from the N-MADP to the client.

3. Problem Statement

Typically, an end-user device has access to multiple communication networks based on different technologies, say LTE, WiFi, DSL, MuLTEfire, for accessing application services. Different technologies exhibit benefits and limitations in different scenarios. For example, WiFi provides high throughput for end users when under good coverage, but the throughput degrades significantly as the user moves closer to the edge of WiFi coverage (typically in the range of few tens of meters) or with large user population (due to contention based WiFi access scheme). In LTE networks, the capacity is often constrained by the limited availability of licensed spectrum. However, the quality of the service is predictable even in multi-user scenarios due to controlled scheduling and licensed spectrum usage.

Additionally, the use of a particular access network path is often coupled with the use of its associated core network and the services that are offered by it. For example, in an enterprise that has deployed both WiFi and LTE networks, the enterprise services, like printers, Corporate Audio and Video conferencing, are accessible only via WiFi access connected to the enterprise hosted (WiFi) core, whereas the LTE access can be used to get operator core anchored services including access to public Internet.

Thus, application performance in different scenarios becomes dependent on the choice of the access networks (e.g. WiFi, LTE, etc.) and the used network and transport protocols (e.g. VPN, MPTCP, GRE etc.). Therefore, to achieve the best possible application performance in a wide range of scenarios, a framework is needed that allows the selection and flexible combination of access and core network paths and used protocols for uplink and downlink data delivery.

For example, to ensure best performance for enterprise applications at all times, in uncongested scenarios, when the user is under good WiFi coverage, it would be beneficial to use WiFi access in both uplink and downlink for connecting to enterprise applications. However, in congested scenarios or when the user is getting close to the edge of its WiFi coverage, the use of WiFi in uplink by multiple users can lead to degraded capacity and increased delays due to contention. In this case, it would be beneficial to at least use the

LTE access for increased uplink coverage while WiFi may still continue to be used for downlink.

4. Requirements

The requirements set out in this section are for the definition of behavior of the MAMS mechanism and the related functional elements.

4.1. Access Technology Agnostic Interworking

The access nodes may use different technology types like LTE, WiFi, etc. The framework, however, MUST be agnostic to the type of underlying technology used at the access network.

4.2. Support Common Transport Deployments

The network path selection and user data distribution MUST work transparently across various transport deployments that include end-to-end IPsec, VPNs, and middleboxes like NATs and proxies.

4.3. Independent Access Path Selection for Uplink and Downlink

A Client SHOULD be able to transmit on the uplink and, receive on the downlink, using one or more accesses. The selection of the access paths for uplink and downlink SHOULD happen independent of each other.

4.4. Core Selection Independent of Uplink and Downlink Access

A client SHOULD flexibly select the Core, independent of the access paths used to reach the Core, depending on the application needs, local policies and the result of MAMS control plane negotiation.

4.5. Adaptive Access Network Path Selection

The framework MUST have the ability to determine the quality of each of the network paths, e.g. access link delay and capacity. The network path quality information needs to be considered in the logic for selection of the combination of network paths to be used for transporting user data. The path selection algorithm can use network path quality information, in addition to other considerations like network policies, for optimizing network usage and enhancing QoE delivered to the user.

4.6. Multipath Support and Aggregation of Access Link Capacities

The framework MUST support distribution and aggregation of user data across multiple network paths at the IP layer. The client SHOULD be able to leverage the combined capacity of the multiple network connections by enabling simultaneous transport of user data over multiple network paths. If required, packet re-ordering needs to be done at the receiver. The framework MUST allow flexibility to choose the flow steering and aggregation protocols based on capabilities supported by the client and the network data plane entities. The multi-connection aggregation solution MUST support existing transport and network layer protocols like TCP, UDP, GRE. The framework MUST allow use and configuration of existing aggregation protocols such as Multi-Path TCP (MPTCP) and SCTP.

4.7. Scalable Mechanism based on User Plane Interworking

The framework MUST leverage commonly available transport, routing and tunneling capabilities to provide user plane interworking functionality. The addition of functional elements in the user plane path between the client and the network MUST not impact the access technology specific procedures. This makes solution easy to deploy and scale when different networks are added and removed.

4.8. Separate Control and Data Plane functions

The client MUST use the control plane protocol to negotiate with the network, the choice of access and core network paths for both uplink and downlink, as well as the user plane protocol treatment. The control plane MUST configure the actual user plane data distribution function per this negotiation. A common control protocol SHOULD allow creation of multiple user plane function instance with potentially different user plane (e.g. tunneling) protocol types. This enables maintaining a clear separation between the control and data plane functions, allowing the framework to be scalable and extensible, e.g. using SDN based architecture and implementations.

4.9. Lossless Path (Connection) Switching

When switching data traffic from one path (connection) to another, packets may be lost or delivered out-of-order, which will have negative impacts on the performance of higher layer protocols, e.g. TCP. The framework SHOULD provide necessary mechanisms to ensure in-order delivery at the receiver, e.g. during path switching. The framework MUST not cause any packet loss beyond that of access network mobility functions may cause.

4.10. Concatenation and Fragmentation for adaptation to MTU Differences

Different network paths may have different security and middlebox (e.g NAT) configurations, which will lead to use of different tunneling protocols for transport of data between the network user plane function and the client. As a result, different effective payload sizes (e.g. due to variable encapsulation header overheads) per network path are possible. Hence, MAMS framework SHOULD support fragmentation of a single IP packet payload across MTU sized IP packets to avoid IP fragmentation when aggregating packets from different paths. Further, concatenation of multiple IP packets into a single IP packet to improve efficiency in packing the MTU size should also be supported.

4.11. Configuring Network Middleboxes based on Negotiated Protocols

The framework SHOULD enable identification of the optimal parameters that may be used for configuring the middle-boxes, like radio link dormancy timers, binding expiry times and supported MTUs, for efficient operation of the user plane protocols, based on parameters negotiated between the client and the network, e.g. Configuring longer binding expiry time in NATs when UDP transport is used in contrast to the scenario where TCP is configured at the transport layer.

4.12. Policy based Optimal Path Selection

The framework MUST support consideration of policies at the client, in addition to guidance from the network, for network path selection addressing different application requirements.

4.13. Access Technology Agnostic Control Signaling

The control plane signaling MUST NOT be dependent on the underlying access technology procedures, e.g. be carried transparently as user plane. It should support delivery of control plane signaling over the existing Internet protocols, e.g. TCP or UDP.

4.14. Service Discovery and Reachability

There can be multiple instances of the control and user plane functional elements of the framework, either collocated or hosted on separate network elements, and reachable via any of the available user plane paths. The client MUST have flexibility to choose the appropriate control plane instance in the network and use the control plane signaling to choose the desired user plane functional element instances. The choice can be based on considerations like, but not

limited to, quality of link through which the network function is reachable, client preferences, pre-configuration etc.

5. Solution Principles

This document proposes the Multiple Access Management Services (MAMS) framework for dynamic selection and flexible combination of access and core network paths independently for the uplink and downlink, as well as the user plane treatment for the traffic spread across the selected links. MAMS framework consists of clearly separated control and user plane functions in the network and the client. The control plane protocol allows configuration of the user plane protocols and desired network paths for transport of application traffic. The control plane messages are carried as user plane data over any of the available network paths between the peer control plane functional elements in the client and the network. Multiple user plane paths are dynamically distributed across multiple access networks and aggregated inside the common core network. The access network diversity is not exposed to the application servers but kept within the scope of the elements defined in this framework. This offloads the application servers from reacting to access link changes caused to mobility events or changing of link characteristics. The selection of paths and user plane treatment of the traffic, is based on negotiation of capabilities (of device and network) and probing of network link quality between the user plane functional elements at the end-user device/client and the network. The framework enables leveraging network intelligence to setup and dynamically configure the best access network path combination based on device and network capabilities, application needs and knowledge of the network state.

6. MAMS Reference Architecture

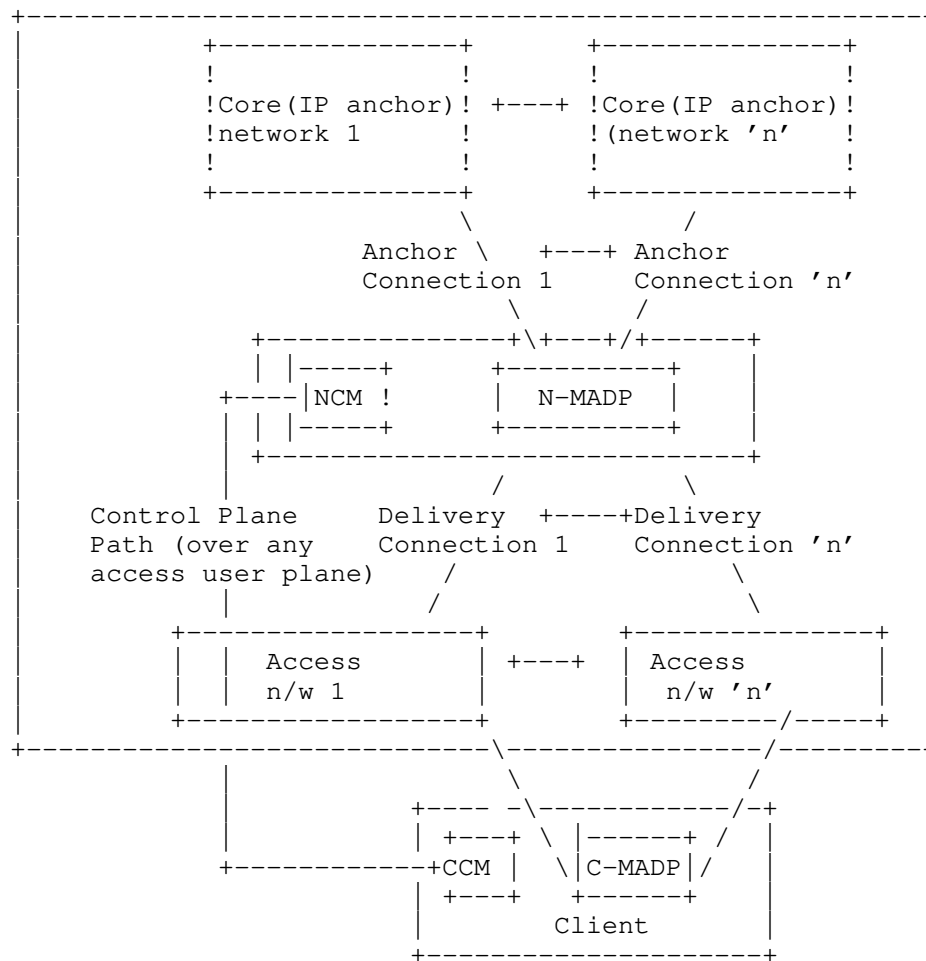


Figure 1: MAMS Reference Architecture

Figure 1 illustrates MAMS architecture for the scenario of a client served by multiple (n) networks. It introduces the following functional elements,

- o Network Connection Manager (NCM) and Client Connection Manager (CCM) in the control plane, and
- o Network Multi Access Data Proxy (N-MADP) and Client Multi Access Data Proxy (C-MADP) handling the user plane.

NCM: It is the functional element in the network that handles the MAMS control plane procedures. It configures the network (N-MADP)

and client (C-MADP) user plane functions like negotiating the client on the use of available access network paths, protocols and rules for processing the user plane traffic, as well as link monitoring procedures. The control plane messages between the NCM and CCM are transported as an overlay, without any impact to the underlying access networks.

CCM: It is the peer functional element in the client for handling MAMS control plane procedures. It manages multiple network connections at the client. It is responsible for exchange of MAMS signaling messages with the NCM for supporting functions like UL and DL user network path configuration for transporting user data packets, link probing and reporting to support adaptive network path selection by NCM. In the downlink, for the user data received by the client, it configures C-MADP such that application data packet received over any of the accesses to reach the appropriate application on the client. In the uplink, for the data transmitted by the client, it configures the C-MADP to determine the best access links to be used for uplink data based on a combination of local policy and network policy delivered by the NCM.

N-MADP: It is the functional element in the network that handles the user data traffic forwarding across multiple network paths, as well as other user-plane functionalities like encapsulation, fragmentation, concatenation, reordering, retransmission, etc. It is the distribution node that routes the uplink user plane traffic to the appropriate anchor connection towards the core network, and the downlink user traffic to the client over the appropriate delivery connection(s). In the downlink, the NCM configures the use of delivery connections, and user plane protocols at the N-MADP for transporting user data traffic. The N-MADP should implement ECMP support for the down link traffic. Or alternatively, it may be connected to a router with ECMP functionality. The load balancing algorithm at the N-MADP is configured by the NCM, based on static and/or dynamic network policies like assigning access and core paths for specific user data traffic type, data volume based percentage distribution, and link availability and feedback information from exchange of MAMS signaling with the CCM at the Client.. N-MADP can be configured with appropriate user plane protocols to support both per-flow and per-packet traffic distribution across the delivery connections. In the uplink, N-MADP selects the appropriate anchor connection over which to forward the user data traffic, received from the client (via the delivery connections). The forwarding rules in the uplink at the N-MADP are configured by the NCM based on application requirements, e.g. Enterprise hosted Application flows via Wi-Fi Anchor, Mobile Operator hosted applications via the Cellular Core.

C-MADP: It is the functional element in the client that handles the MAMS user plane data procedures. C-MADP is configured by CCM based on signaling exchange with NCM and local policies at the client. The CCM configures the selection of delivery connections and the user plane protocols to be used for uplink user data traffic based on the signaling exchanged with NCM. The C-MADP entity handles user plane data forwarding across multiple delivery connections and associated user-plane functions like encapsulation, fragmentation, concatenation, reordering, retransmissions, etc.

The NCM and N-MADP can be either collocated or instantiated on different network nodes. NCM can setup multiple N-MADP instances in the network. NCM controls the selection of N-MADP instance by the client and the rules for distribution of user traffic across the N-MADP instances., This is beneficial in multiple deployment scenarios, like the following examples.

- o Different N-MADP instances to handle different sets of clients for load balancing across clients
- o Address deployment topologies e.g. N-MADP hosted at the user plane node at the access edge or in the core network, while the NCM hosted at the access edge node)
- o Address access network technology architecture. For example, N-MADP instance at core network node to manage traffic distribution across LTE and DSL networks, and N-MADP instance at access network node to manage traffic distribution across LTE and Wi-Fi traffic.
- o A single client can be configured to use multiple N-MADP instances. This is beneficial in addressing different application requirements. For example, separate N-MADP instances to handle TCP and UDP transport based traffic.

Thus, MAMS architecture flexibly addresses multiple network deployments.

7. MAMS Protocol Architecture

This section describes the protocol structure for the MAMS User and Control plane functional elements.

7.1. MAMS Control-Plane Protocol

Figure 2 shows the default MAMS control plane protocol stack. WebSocket is used for transporting management and control messages between NCM and CCM.

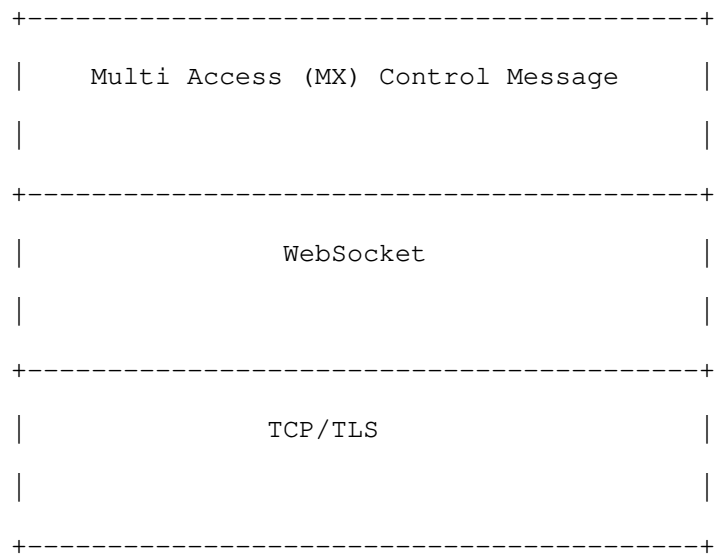


Figure 2: TCP-based MAMS Control Plane Protocol Stack

7.2. MAMS User Plane Protocol

Figure 3 shows the MAMS user plane protocol stack.

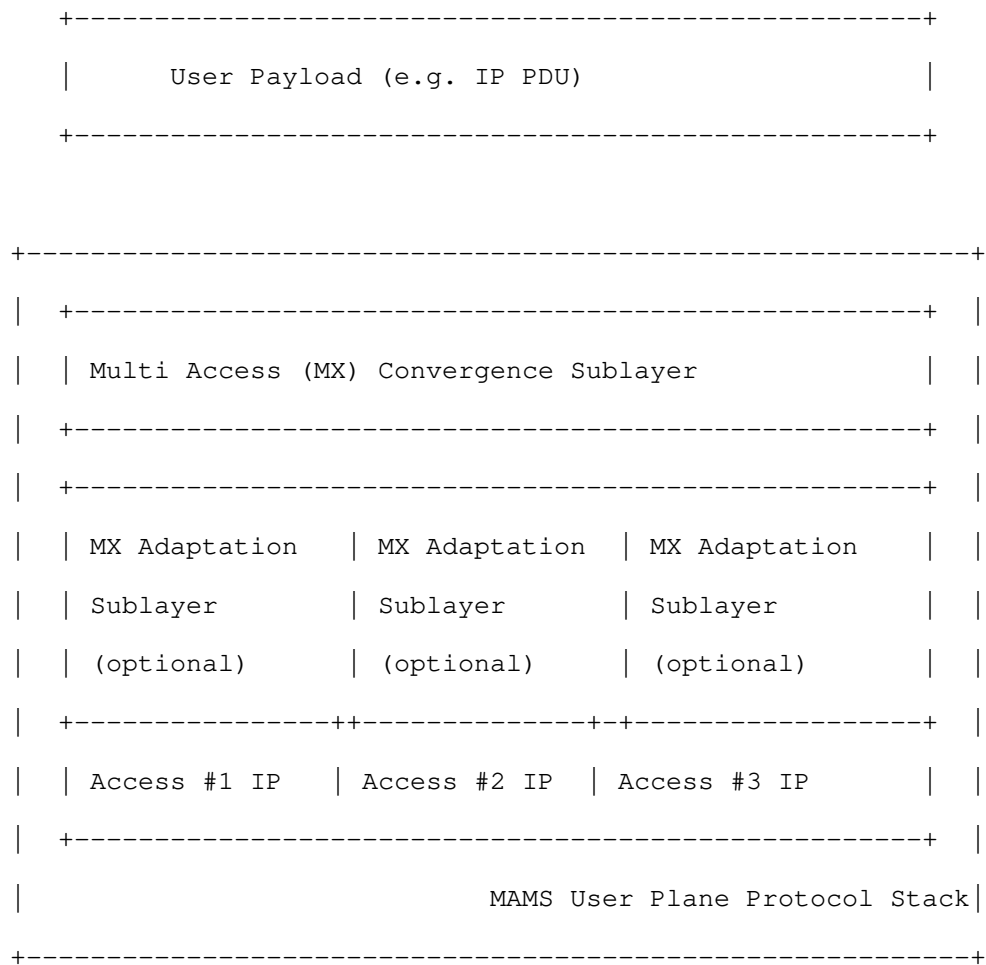


Figure 3: MAMS User Plane Protocol Stack

It consists of the following two Sublayers:

- o Multi-Access (MX) Convergence Sublayer: The MAMS framework configures the Convergence sublayer to perform multi-access specific tasks in the user plane. This layer performs functions

like access (path) selection, multi-link (path) aggregation, splitting/reordering, lossless switching, fragmentation, concatenation, etc. MX Convergence layer can be implemented using existing user plane protocols like Multipath TCP (MPTCP [RFC 6824]), Multi Path QUIC (MPQUIC [I-D.deconinck-multipath-quic]) or by adapting encapsulating header/trailer schemes like Generic Routing and Encapsulation (GRE [RFC 2784], [RFC 2890]), Generic Multi Access (GMA [I-D.zhu-intarea-gma]).

- o Multi-Access (MX) Adaptation Sublayer: The MAMS framework configures the Adaptation Sublayer to address transport network related aspects like reachability and security in the user plane. This layer performs functions to handle tunnelling, network layer security, and NAT. MX Adaptation can be implemented using IPsec, DTLS or Client NAT (Source NAT at Client with inverse mapping at N-MADP [I-D.zhu-intarea-mams-user-protocol]). The MX Adaptation Layer is optional and can be independently configured for each of the Access Links. E.g. In a deployment with LTE (assumed secure) and Wi-Fi (assumed not secure), the MX Adaptation Sublayer can be omitted for the LTE link but MX Adaptation Sublayer is configured as IPsec for securing the Wi-Fi link. Further details on the MAMS user plane are described in [I-D.zhu-intarea-mams-user-protocol].

8. MAMS Control Plane Procedures

8.1. Overview

CCM and NCM exchange signaling messages to configure the user plane functions, C-MADP and N-MADP, at the client and network respectively. The means for CCM to obtain the NCM credentials (FQDN or IP Address) for sending the initial discovery messages are out of the scope of MAMS document. As an example, the client can obtain the NCM credentials using methods like provisioning, DNS query. Once the discovery process is successful, the (initial) NCM can update and assign additional NCM addresses, e.g. based on MCC/MNC tuple information received in the MX Discovery Message, for sending subsequent control plane messages.

CCM discovers and exchanges capabilities with the NCM. NCM provides the credentials of the N-MADP end-point and negotiates the parameters for user plane with the CCM. CCM configures C-MADP to setup the user plane path (e.g. MPTCP/UDP Proxy Connection) with the N-MADP based on the credentials (e.g. (MPTCP/UDP) Proxy IP address and port, Associated Core Network Path), and the parameters exchanged with the NCM. Further, NCM and CCM exchange link status information to adapt traffic steering and user plane treatment with dynamic network conditions. The key procedures are described in details in the following sub-sections.

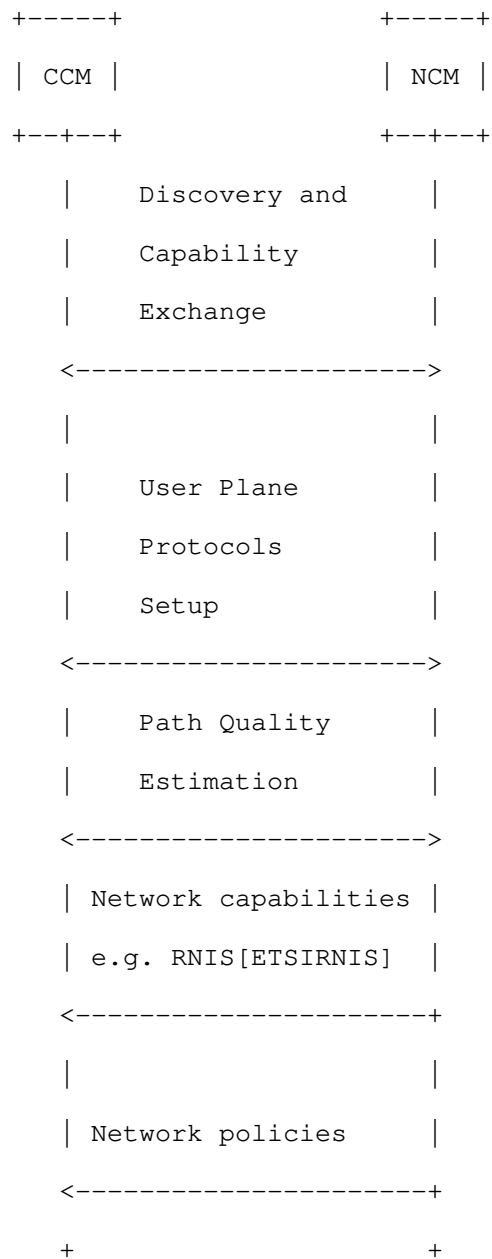


Figure 4: MAMS Control Plane Procedures

8.2. Common fields in MAMS Control Messages

Each MAMS control message consists of the following common fields:

- o Version: indicates the version of MAMS control protocol.
- o Message Type: indicates the type of the message, e.g. MX Discovery, MX Capability REQ/RSP etc.
- o Sequence Number: auto-incremented integer to uniquely identify a transaction of message exchange, e.g. MX Capability REQ/RSP.

8.3. Common Procedures for MAMS Control Messages

This section describes the common procedures for MAMS Control Messages.

8.3.1. Message Timeout

MAMS Control plane peer (NCM or CCM) waits for a duration of MAMS_TIMEOUT ms, after sending a MAMS control message, before timing out when expecting a response. The sender of the message will retransmit the message for MAMS_RETRY times before declaring failure. A failure implies that the MAMS peer is dead, and the sender reverts back to native non-multi access/single path mode. CCM may initiate the MAMS discovery procedure for re-establishment of the MAMS session.

8.3.2. Keep Alive Procedure

MAMS Control plane peers execute the keep alive procedures to ensure that peers are reachable and to recover from dead-peer scenarios. Each MAMS control plane end-point maintains a MAMS_KEEP_ALIVE timer that is set for duration MAMS_KEEP_ALIVE_TIMEOUT. MAMS_KEEP_ALIVE timer is reset whenever the peer receives a MAMS Control message. When MAMS_KEEP_ALIVE timer expires, MAMS KEEP ALIVE REQ message is sent. On reception of a MAMS KEEP ALIVE REQ message, the receiver responds with a MAMS KEEP ALIVE RSP message. If the sender does not receive a MAMS Control message in response to MAMS_RETRY number of retries of MAMS KEEP ALIVE REQ message, the MAMS peer declares that the peer is dead. CCM may initiate MAMS Discovery procedure for re-establishment of the MAMS session.

CCM shall additionally send MX KEEP ALIVE REQ message immediately to NCM whenever it detects a handover from one base station/access point to another. During this time the user equipment shall stop using MAMS user plane functionality in uplink direction till it receives a MX KEEP ALIVE RSP from NCM.

MX KEEP ALIVE REQ includes following information:

- o Reason: Can be 'Timeout' or 'Handover'. Reason 'Handover' shall be used by CCM only on detection of handover.
- o Unique Session Identifier: As defined in Section 8.4.
- o Connection Id: This field shall be mandatorily be included if the reason is 'Handover'.
- o Delivery Node Identity (ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi). This field shall be mandatorily be included if the reason is 'Handover'.

8.4. Discovery & Capability Exchange

Figure 5 shows the MAMS discovery and capability exchange procedure consisting of the following key steps:

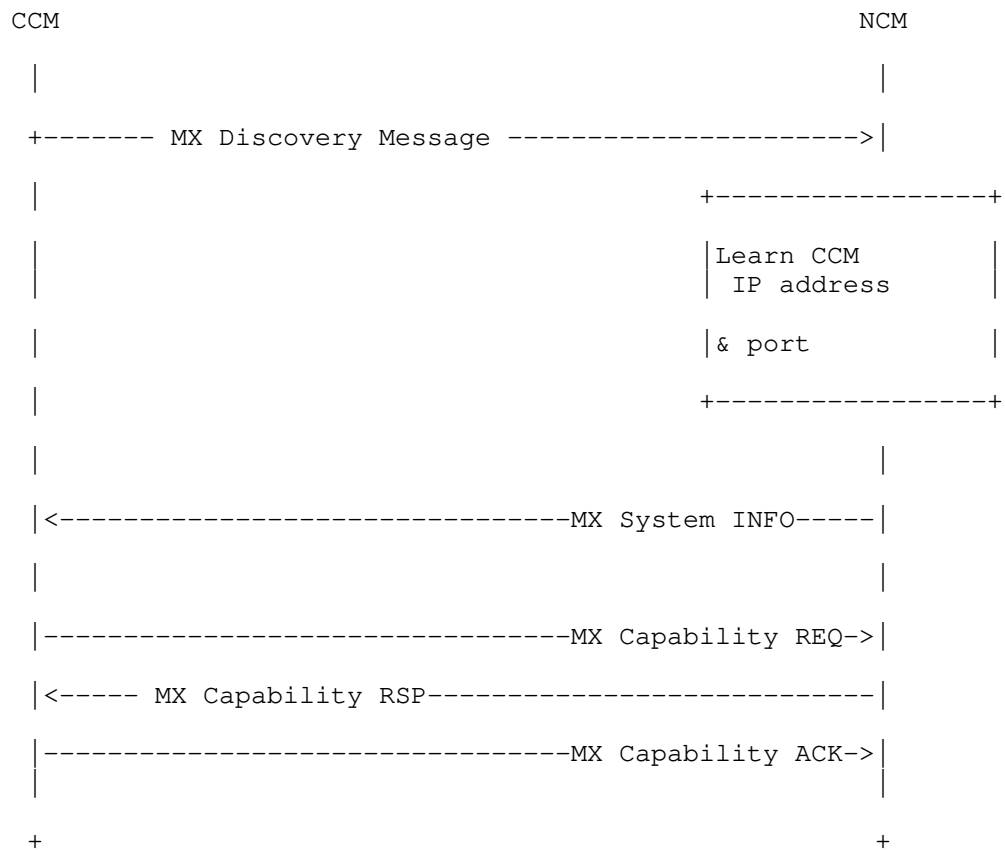


Figure 5: MAMS Control Procedure for Discovery & Capability Exchange

Step 1 (Discovery): CCM periodically sends out the MX Discovery Message to a pre-defined (NCM) IP Address/port until MX System INFO message is received in acknowledgement.

MX Discovery Message includes the following information:

- o MAMS Version
- o MCC/MNC Tuple: Optional Parameter to Identify the Operator Network to which the client is subscribed, in conformance with format specified in [E212]

MX System INFO includes the following information:

- o Number of Anchor Connections

For each Anchor Connection, it includes the following parameters:

- * Connection ID: Unique identifier for the Anchor Connection
- * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- * NCM Endpoint Address (For Control Plane Messages over this connection)
 - + IP Address or FQDN (Fully Qualified Domain Name)
 - + Port Number

Step 2 (Capability Exchange): On receiving MX System Info message CCM learns the IP Address and port to start the step 2 of the control plane connection, and sends out the MX Capability REQ message, including the following Parameters:

- o MX Feature Activation List: Indicates if the corresponding feature is supported or not, e.g. lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, Measurement, probing, etc.
- o Number of Anchor Connections (Core Networks)

For each Anchor Connection, it includes the following parameters:

- * Connection ID
- * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- o Number of Delivery Connections (Access Links)

For each Delivery Connection, it includes the following parameters:

- * Connection ID
- * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- o MX Convergence Method Support List
 - * GMA
 - * MPTCP Proxy
 - * GRE Aggregation Proxy
 - * MPQUIC
- o MX Adaptation Method Support List
 - * UDP Tunnel without DTLS
 - * UDP Tunnel with DTLS
 - * IPsec Tunnel [RFC3948]

- * Client NAT

In response, NCM creates a unique identity for the CCM session, and sends out the MX Capability RSP message, including the following information:

- o MX Feature Activation List: Indicates if the corresponding feature is enabled or not, e.g. lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, Measurement, probing, etc.
- o Number of Anchor Connections (Core Networks)

For each Anchor Connection, it includes the following parameters:

- * Connection ID
- * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- o Number of Delivery Connections (Access Links)

For each Delivery Connection, it includes the following parameters:

- * Connection ID
- * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: Multi-Fire; 3: LTE)
- o MX Convergence Method Support List
 - * GMA
 - * MPTCP Proxy
 - * GRE Aggregation Proxy
 - * MPQUIC
- o MX Adaptation Method Support List
 - * UDP Tunnel without DTLS
 - * UDP Tunnel with DTLS
 - * IPsec Tunnel [RFC3948]
 - * Client NAT

Unique Session Identifier: Unique session identifier for the CCM which has setup the connection. In case the session for the UE already exists then the existing unique session identifier is sent back.

- o NCM Id: Unique Identity of the NCM in the operator network.
- o Session Id: Unique identity assigned to the CCM instance by this NCM instance.

In response to MX Capability RSP message, the CCM sends confirmation (or reject) in the MX Capability ACK message. MX Capability ACK includes the following parameters

- o Unique Session Identifier: Same identifier as provided in MX Capability RSP.
- o Acknowledgement: An indication if the client has accepted or rejected the capability phase.
 - * MX ACCEPT: CCM Accepts the Capability set proposed by the NCM.
 - * MX REJECT: CCM Rejects the Capability set proposed by the NCM.

If MX_REJECT is received by the NCM, the current MAMS session will be terminated.

If CCM can no longer continue with the current capabilities, it should send an MX SESSION TERMINATE message to terminate the MAMS session. In response, the NCM should send a MX SESSION TERMINATE ACK to confirm the termination.

8.5. User Plane Configuration

Figure 6 shows the user plane configuration procedure consisting of the following key steps:

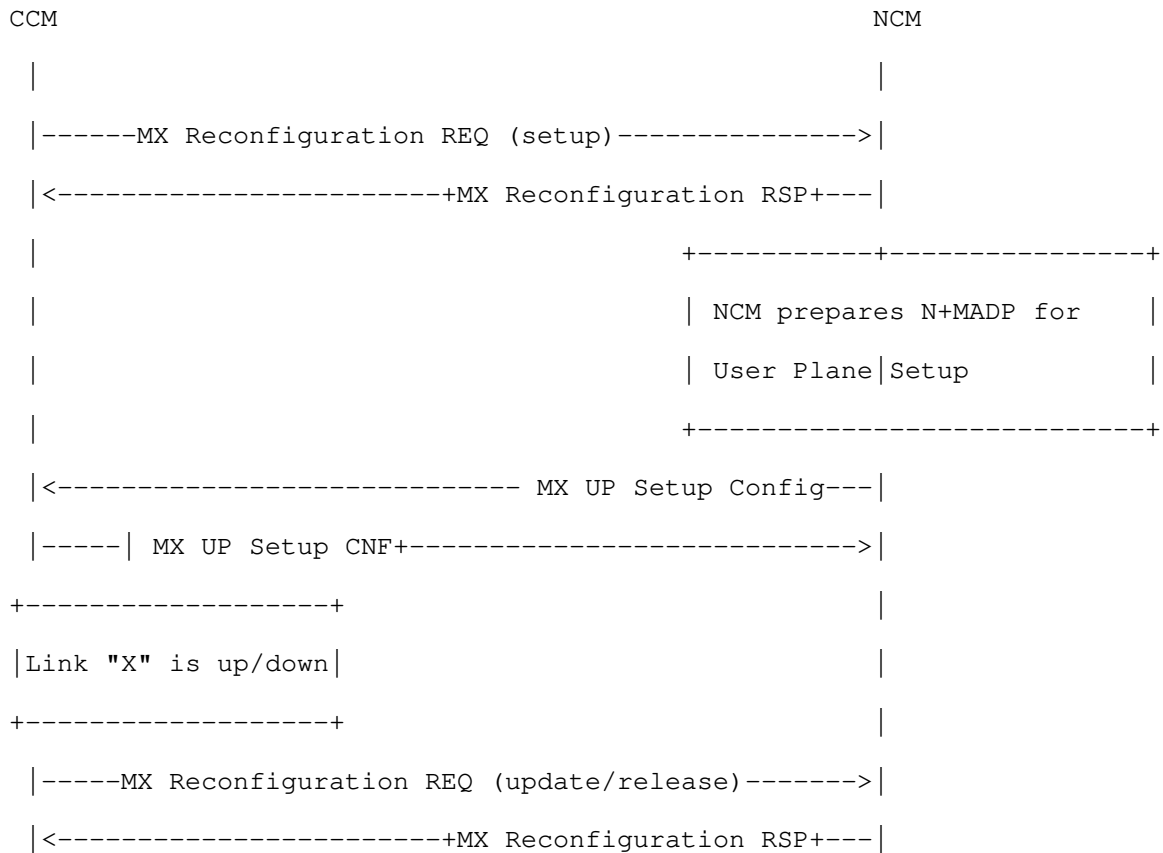


Figure 6: MAMS Control Procedure for User Plane Configuration

Reconfiguration: when the client detects that the link is up/down or the IP address changes (e.g. via APIs provided by the client OS), CCM sends out a MX Reconfiguration REQ Message to setup / release / update the connection, and the message SHOULD include the following information

- o Unique Session Identifier: Identity of the CCM identity at NCM, created by NCM during the capability exchange phase.

- o Reconfiguration Action: indicate the reconfiguration action (0:release; 1: setup; 2: update).
- o Connection ID: identify the connection for reconfiguration

If (Reconfiguration Action is setup or update), then include the following parameters

- o IP address of the connection
- o SSID (if Connection Type = WiFi)
- o MTU of the connection: MTU of the delivery path that is calculated at the UE for use by NCM to configure fragmentation and concatenation procedures[I-D.zhu-intarea-mams-user-protocol] at N-MADP.
- o Delivery Node Identity: Identity of the node to which the client is attached. ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi.

At the beginning of a connection setup, CCM informs the NCM of the connection status using the MX Reconfiguration REQ message with Reconfiguration Action type set to "setup". NCM acknowledges the connection setup status and exchanges parameters with the CCM for user plane setup, described as follows.

User Plane Protocols Setup: Based on the negotiated capabilities, NCM sets up the user plane (Adaptation Layer and Convergence Layer) protocols at the N-MADP, and informs the CCM of the user plane protocols to setup at the client (C-MADP) and the parameters for C-MADP to connect to N-MADP.

The MX UP Setup Config is used to create (multiple) MADP instances with each Anchor Connection having one or more Configurations, namely MX Configurations. It consists of the following parameters:

- o Number of Anchor Connections (Core Networks)

For Each Anchor Connection, it includes the following parameters

- * Anchor Connection ID
- * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- * Number of Active MX Configurations (Included only if more than one MX configurations are active for the anchor connection)

For each active MX configuration, it includes the following parameters

- + MX Configuration ID (included if more than one MX Configuration is present)

- + MX Convergence Method, one of the following
 - GMA
 - MPTCP Proxy
 - GRE Aggregation Proxy
 - MPQUIC
- + MX Convergence Method Parameters
 - Convergence Proxy IP Address
 - Convergence Proxy Port
 - Client Key
- + MX Convergence Control Parameters (included if any MX Control PDU, e.g. Probe-REQ/ACK, is supported):
 - UDP Port Number for sending and receiving MX Control PDUs, e.g. Probe-REQ/ACK, Keep-Alive, etc.)
 - Convergence Proxy Port
- + Number of Delivery Connections

For each Delivery Connection, include the following:

- Delivery Connection ID
- Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- MX Adaptation Method, one of the following
 - o UDP Tunnel without DTLS
 - o UDP Tunnel with DTLS
 - o IPsec Tunnel
 - o Client NAT
- MX Adaptation Method Parameters
 - o Tunnel Endpoint IP Address
 - o Tunnel Endpoint Port
 - o Shared Secret
 - o Header Optimization (included only if MX Convergence Method is GMA)

e.g. When LTE and Wi-Fi are the two user plane accesses, NCM conveys to CCM that IPsec needs to be setup as the MX Adaptation Layer over the Wi-Fi Access, using the following parameters - IPsec end-point IP address, Pre-Shared Key. No Adaptation Layer is needed or Client NAT may be used over the LTE Access as it is considered secure with no NAT.

Similarly, as an example of the MX Convergence Method configuration is to indicate the convergence protocol as MPTCP Proxy along with

parameters for connection to the MPTCP Proxy, namely IP Address and Port of the MPTCP Proxy for TCP Applications.

Once the user plane protocols are configured, CCM informs the NCM of the status via the MX UP Setup CNF message. The MX UP Setup CNF consists of the following parameters:

- o Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- o MX Convergence Control Parameters (included if any MX Control PDU, e.g. Probe-REQ/ACK, Keep-alive, is supported):
 - * UDP Port Number for sending and receiving MX Control PDUs, e.g. Probe-REQ/ACK, Keep-Alive, etc.)
 - * MX Configuration ID (if MX Configuration ID is specified in MX UP Setup Config, indicate the MX Configuration that will be used for Probing)
- o Client Adaptation Layer Parameters:
 - * Number of Delivery Connections
 - * For each Delivery Connection, include the following:
 - + Delivery Connection ID
 - + UDP port number: If UDP based adaptation is in use, the UDP port at C-MADP side

8.6. MAMS Path Quality Estimation

Path quality estimations can be done either passively or actively. Traffic measurements in the network could be performed passively by comparing the real-time data throughput of the device with the capacity available in the network. In special deployments where the NCM has interfaces with access nodes, direct interfaces can be used to gather path quality information. For example, the utilization of a cell/eNB attached to a device could be used as an indicator for path quality estimations without creating an extra traffic overhead. Active measurements by the device are an alternative for estimating path quality.

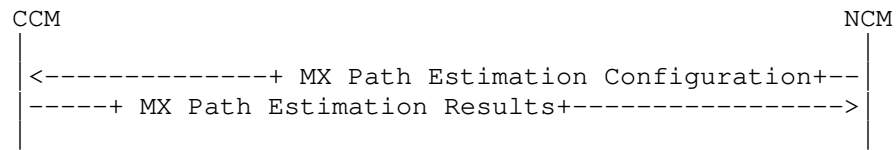


Figure 7: MAMS Control Plane Procedure for Path Quality Estimation

NCM sends following the configuration parameters in the MX Path Estimation Configuration message to the CCM

- o Connection ID (of Delivery Connection whose path quality needs to be estimated)
- o Init Probe Test Duration (ms)
- o Init Probe Test Rate (Mbps)
- o Init Probe Size (Bytes)
- o Init Probe Ack Required (0 -> No/1 -> Yes)
- o Active Probe Frequency (ms)
- o Active Probe Size (Bytes)
- o Active Probe Test Duration (ms)
- o Active Probe Ack Required (0 -> No/1 -> Yes)

CCM configures the C-MADP for probe reception based on these parameters and for collection of the statistics according to the following configuration.

- o Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- o Init Probe Results Configuration
 - * Lost Probes (%)
 - * Probe Receiving Rate (packets per second)
- o Active Probe Results Configuration
 - * Average Throughput in the last Probe Duration

The user plane probing is divided into two phases - Initialization phase and Active phase.

- o Initialization phase: A network path that is not included by N-MADP for transmission of user data is deemed to be in the Initialization phase. The user data may be transmitted over other available network paths.

- o Active phase: A network path that is included by N-MADP for transmission of user data is deemed to be in Active phase.

In Initialization phase, NCM configures N-MADP to send an MX Idle Probe REQ message. CCM collects the Idle probe statistics from C-MADP and sends the MX Path Estimation Results Message to NCM per the Initialization Probe Results configuration.

In Active phase, NCM configures N-MADP to send an MX Active Probe REQ message.. C-MADP calculates the metrics as specified by the Active Probe Results Configuration. CCM collects the Active probe statistics from C-MADP and sends the MX Path Estimation Results Message to NCM per the Active Probe Results configuration.

The following sub-sections define the control PDU encoding for Probe and Keep Alive messages to support path quality estimation.

8.6.1. MX Control PDU definition

Control PDUs are sent as UDP Messages between C-MADP and N-MADP to exchange control messages for keep-alive or path quality estimation. MX Probe Parameters are negotiated during the User Plane Setup phase (MX UP SETUP CFG and MX UP SETUP CNF). Figure 7 shows the MX control PDU format with the following fields:

- o Type (1 Byte): the type of the MX control message
 - * 0: Keep-Alive
 - * 1: Probe REQ/ACK
 - * Others: Reserved
- o CID (1 Byte): the connection ID of the delivery connection for sending out the MX control message
- o MX Control Message (variable): the payload of the MX control message
- o Figure 8 shows the MX Control PDU format. MX Control PDU is sent as a normal user plane packet over the desired delivery connection whose quality and reachability needs to be determined.

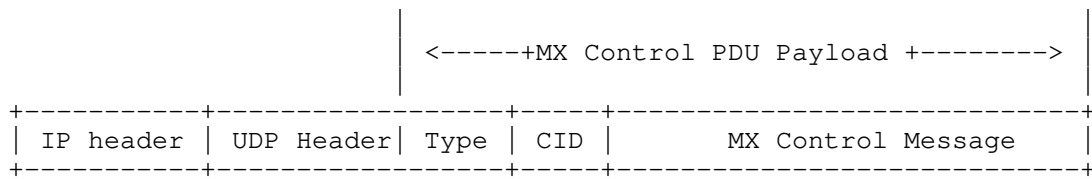


Figure 8: MX Control PDU Format

8.6.2. Keep-Alive Message

The "Type" field is set to "0" for Keep-Alive messages. C-MADP may send out Keep-Alive message periodically over one or multiple delivery connections, especially if UDP tunneling is used as the adaptation method for the delivery connection with a NAT function on the path.

A Keep-Alive message is 2 Bytes long, and consists of the following fields:

- o Keep-Alive Sequence Number (2 Bytes): the sequence number of the keep-alive message.

8.6.3. Probe REQ/ACK Message

The "Type" field is set to "1" for Probe REQ/ACK messages. N-MADP may send out the Probe REQ message for path quality estimation. In response, C-MADP may send back the Probe ACK message.

A Probe REQ message consists of the following fields:

- o Probing Sequence Number (2 Bytes): the sequence number of the Probe REQ message
- o Probing Flag (1 Byte):
 - * Bit #0: a Probe ACK flag to indicate if the Probe ACK message is expected (1) or not (0);
 - * Bit #1: a Probe Type flag to indicate if the Probe REQ/ACK message is sent during the initialization phase (0) when the network path is not included for transmission of user data or the active phase (1) when the network path is included for transmission of user data;
 - * Bit #2: a bit flag to indicate the presence of the Reverse Connection ID (R-CID) field.
 - * Bit #3~7: reserved

- o Reverse Connection ID (1 Byte): the connection ID of the delivery connection for sending out the Probe ACK message on the reverse path
- o Padding (variable)

The "R-CID" field is only present if both Bit #0 and Bit #2 of the "Probing Flag" field are set to "1". Moreover, Bit #2 of the "Probing Flag" field SHOULD be set to "0" if the Bit #0 is "0", indicating the Probe ACK message is not expected.

If the "R-CID" field is not present but the Bit #0 of the "Probing Flag" field is set to "1", the Probe ACK message SHOULD be sent over the same delivery connection as the Probe REQ message.

The "Padding" field is used to control the length of Probe REQ message.

C-MADP SHOULD send out the Probe ACK message in response to a Probe REQ message with the Probe ACK flag set to "1".

A Probe ACK message is 3 Bytes long, and consists of the following fields:

- o Probing Acknowledgement Number (2 Bytes): the sequence number of the corresponding Probe REQ message

8.7. MAMS Traffic Steering

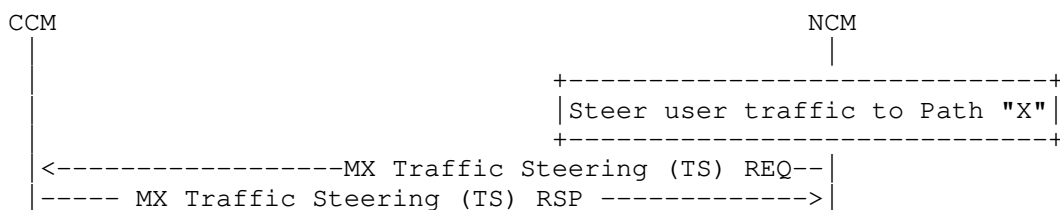


Figure 9: MAMS Traffic Steering Procedure

NCM sends out a MX Traffic Steering (TS) REQ message to steer data traffic. It is also possible to send data traffic over multiple connections simultaneously, i.e. aggregation. The message includes the following information:

- o Connection ID of the Anchor Connection
- o MX Configuration ID (if MX Configuration ID is specified in MX UP Setup Config)

- o Connection ID List of Delivery Connections for DL traffic
- o Connection ID of Default UL Delivery Connection
- o For the number of Specific UL traffic Templates, include the following
 - * Traffic Template for identifying the UL traffic
 - * Connection ID List of Delivery connections for UL traffic identified by the traffic template
- o MX Feature Activation List: each parameter indicates if the corresponding feature is enabled or not: lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, Measurement, probing

In response, CCM sends out a MX Traffic Steering (TS) RSP message, including the following information:

- o Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- o MX Feature Activation List: each parameter indicates if the corresponding feature is enabled or not: lossless switching, fragmentation, concatenation, Uplink aggregation, Downlink aggregation, probing

8.8. MAMS Application MADP Association

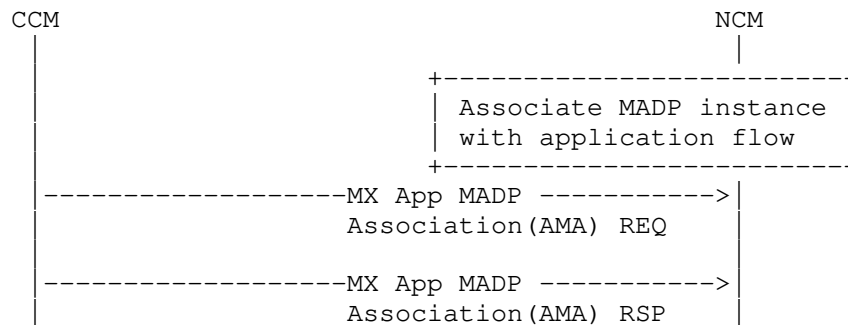


Figure 10: MAMS Application MADP Association Procedure

CCM sends out a MX App MADP Association (AMA) REQ message to request association of a specific Application flow with a specific MADP instance ID for the anchor connection with multiple active MX configurations. MADP Instance ID is a tuple (Anchor Connection ID, MX Configuration ID). This provides the capability for the client to indicate the user plane processing that needs to be associated with different application flows depending on their needs. The

application flow is identified by its associated traffic flow template.

The message includes the following information:

- o Number of Application Flows

For Each Application Flow, identified by the Traffic Flow Template(s),

- * Anchor Connection ID
- * MX Configuration ID (if more than one MX Configurations are associated with an Anchor Connection)
- * Traffic Template for identifying the UL traffic
- * Traffic Template for identifying the DL traffic

In response, NCM sends out a MX App MADP Association (AMA) RSP message, including the following information:

- o Number of Application Flows

For Each Application Flow, identified by the Traffic Flow Template(s),

- * Status (Success or Failure)

8.9. MAMS Network ID Indication

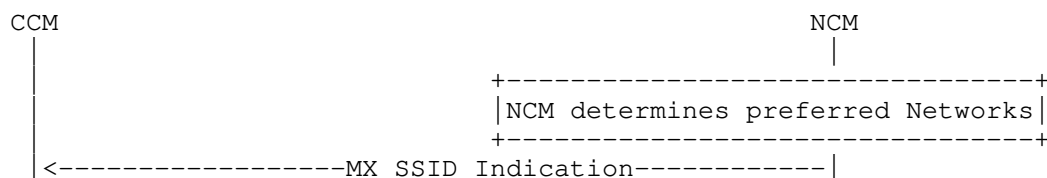


Figure 11: MAMS Network ID Indication Procedure

NCM indicates the preferred network list to the CCM to guide client on networks that it should connect to. To indicate preferred Wi-Fi Networks, the NCM sends the list of WLAN networks, represented by SSID/BSSID/HESSID, available in the MX SSID Indication.

8.10. MAMS Client Measurement Configuration and Reporting

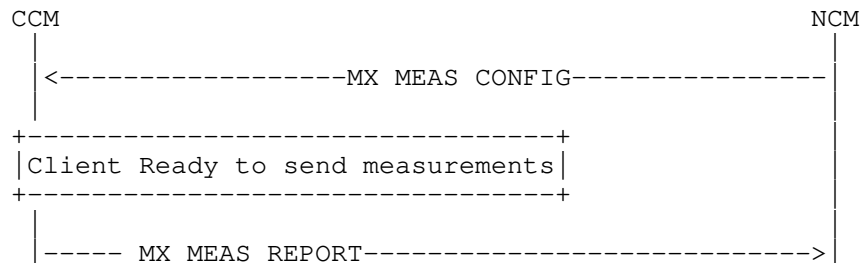


Figure 12: MAMS Client Measurement Configuration and Reporting Procedure

NCM configures the CCM with the different parameters (e.g. radio link information), with the associated thresholds to be reported by the client. The MX MEAS CONFIG message contains the following parameters. For each Delivery Connection, include the following:

- o Delivery Connection ID
- o Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
- o If Connection Type is Wi-Fi
 - * WLAN_RSSI_THRESH: High and Low Thresholds for sending Average RSSI of the Wi-Fi Link.
 - * WLAN_RSSI_PERIOD: Periodicity in ms for sending Average RSSI of the Wi-Fi Link.
 - * WLAN_LOAD_THRESH: High and Low Thresholds for sending Loading of the WLAN system.
 - * WLAN_LOAD_PERIOD: Periodicity in ms for sending Loading of the WLAN system.
 - * UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput on the Wi-Fi link.
 - * UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput on the Wi-Fi link.
 - * DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput on the Wi-Fi link.
 - * DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput on the Wi-Fi link.
 - * EST_UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput (EstimatedThroughputOutbound as defined in [IEEE]) on the Wi-Fi link.

- * EST_UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput (EstimatedThroughputOutbound as defined in [IEEE]) on the Wi-Fi link.
- * EST_DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput (EstimatedThroughputInbound as defined in [IEEE]) on the Wi-Fi link.
- * EST_DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput (EstimatedThroughputInbound as defined in [IEEE]) on the Wi-Fi link.
- o If Connection Type is LTE
 - * LTE_RSRP_THRESH: High and Low Thresholds for sending RSRP of Serving LTE link.
 - * LTE_RSRP_PERIOD: Periodicity in ms for sending RSRP of Serving LTE link.
 - * LTE_RSRQ_THRESH: High and Low Thresholds for sending RSRQ of the serving LTE link.
 - * LTE_RSRQ_PERIOD: Periodicity in ms for sending RSRP of Serving LTE link.
 - * UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput on the serving LTE link.
 - * UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput on the serving LTE link.
 - * DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput on the serving LTE link.
 - * DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput on the serving LTE link.
- o If Connection Type is 5G NR
 - * NR_RSRP_THRESH: High and Low Thresholds for sending RSRP of Serving NR link.
 - * NR_RSRP_PERIOD: Periodicity in ms for sending RSRP of Serving NR link.
 - * NR_RSRQ_THRESH: High and Low Thresholds for sending RSRQ of the serving NR link.
 - * NR_RSRQ_PERIOD: Periodicity in ms for sending RSRP of Serving NR link.
 - * UL_TPUT_THRESH: High and Low Thresholds for sending Reverse Link Throughput on the serving NR link.
 - * UL_TPUT_PERIOD: Periodicity in ms for sending Reverse Link Throughput on the serving NR link.
 - * DL_TPUT_THRESH: High and Low Thresholds for sending Forward Link Throughput on the serving NR link.
 - * DL_TPUT_PERIOD: Periodicity in ms for sending Forward Link Throughput on the serving NR link.

The MX MEAS REPORT message contains the following parameters

- o Unique Session Identifier: Session identifier provided to the client in MX Capability RSP.
- o For each Delivery Connection, include the following:
 - * Delivery Connection ID
 - * Connection Type (e.g., 0: Wi-Fi; 1: 5G NR; 2: MulteFire; 3: LTE)
 - * Delivery Node Identity (ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi)
 - * If Connection Type is Wi-Fi
 - + WLAN_RSSI: Average RSSI of the Wi-Fi Link.
 - + WLAN_LOAD: Loading of the WLAN system.
 - + UL_TPUT: Reverse Link Throughput on the Wi-Fi link.
 - + DL_TPUT: Forward Link Throughput on the Wi-Fi link.
 - + EST_UL_TPUT: Estimated Reverse Link Throughput on the Wi-Fi link (EstimatedThroughputOutbound as defined in [IEEE]).
 - + EST_DL_TPUT: Estimated Forward Link Throughput on the Wi-Fi link (EstimatedThroughputInbound as defined in [IEEE]).
 - * If Connection Type is LTE
 - + LTE_RSRP: RSRP of Serving LTE link.
 - + LTE_RSRQ: RSRQ of the serving LTE link.
 - + UL_TPUT: Reverse Link Throughput on the serving LTE link.
 - + DL_TPUT: Forward Link Throughput on the serving LTE link.
 - * If Connection Type is 5G NR
 - + NR_RSRP: RSRP of Serving NR link.
 - + NR_RSRQ: RSRQ of the serving NR link.
 - + UL_TPUT: Reverse Link Throughput on the serving NR link.
 - + DL_TPUT: Forward Link Throughput on the serving NR link.

8.11. MAMS Session Termination Procedure

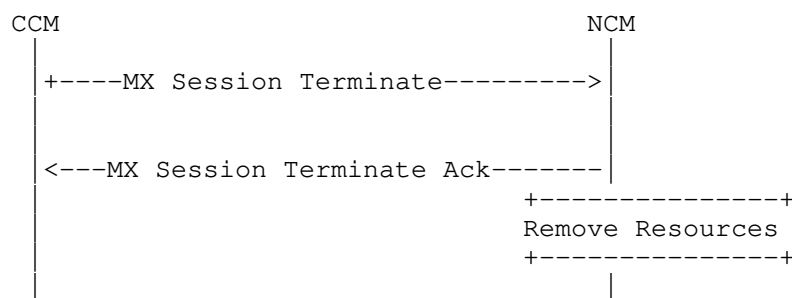


Figure 13: MAMS Session Termination Procedure - Client Initiated

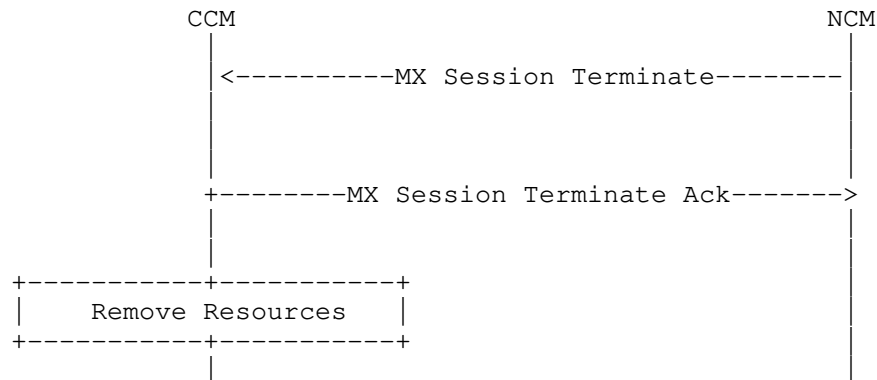


Figure 14: MAMS Session Termination Procedure - Network Initiated

At any point in MAMS functioning if CCM or NCM is unable to support the MAMS functions anymore, then either of them can initiate a termination procedure by sending MX Session Terminate to the peer, the peer shall acknowledge the termination by sending MX Session Terminate ACK message. After the session is disconnected the CCM shall start a new procedure with MX Discover Message. MX Session Terminate message shall contain Unique Session Identifier and reason for termination in Request. Possible reasons for termination can be:

- o Normal Release
- o No Response from Peer
- o Internal Error

8.12. MAMS Network Analytics Request Procedure

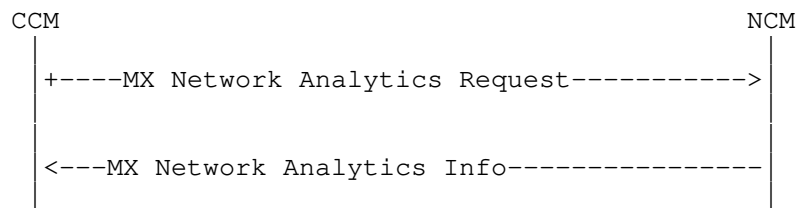


Figure 15: MAMS Network Analytics Request Procedure

CCM sends the MX Network Analytics Request informs the NCM to send information related to network parameters like bandwidth, latency, jitter, signal quality based on application of analytics at the

network utilizing the received path measurements and client measurement reporting.

The MX Network Analytics Request message consists of the following parameters.

- o Link Quality Indicators, one or more of the following:

- * Bandwidth
- * Jitter
- * Latency
- * Signal Quality

NCM sends the MX Network Analytics Info to convey the analytics info, predictive parameters with likelihoods, for the different parameters of interest for the CCM.

The MX Network Analytics Info messages consists of the following parameters.

- o Number of Delivery Connections For Each Delivery Connection,
 - * Access Link Identifier
 - + Connection Type
 - + Connection ID
 - * Link Quality Indicator
 - + Bandwidth
 - Predicted Value (in Mbps)
 - - Likelihood (in Percentage)
 - Prediction Validity (Validity Time in s)
 - + Jitter
 - Predicted Value (in s)
 - - Likelihood (in Percentage)
 - Prediction Validity (Validity Time in s)
 - + Latency
 - Predicted Value (in s)
 - - Likelihood (in Percentage)
 - Prediction Validity (Validity Time in s)
 - + Signal Quality
 - if Delivery Connection Type is LTE, LTE_RSRP Predicted Value (in dBm)

- if Delivery Connection Type is LTE, LTE_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRP Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is WiFi, WLAN_RSSI Predicted Value (in dBm)
- - Likelihood (in Percentage)
- Prediction Validity (Validity Time in s)

9. Generic MAMS Signaling Flow

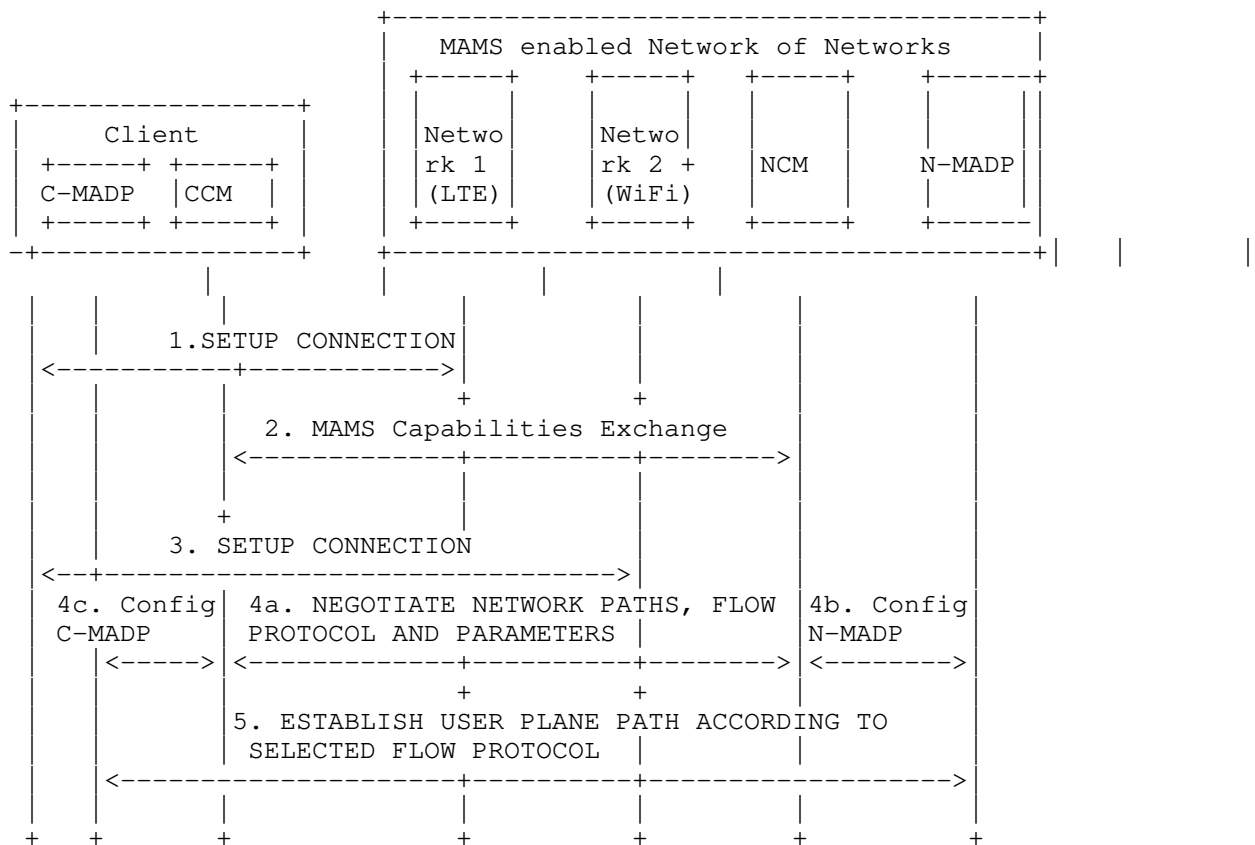


Figure 16: MAMS call flow

Figure 16 illustrates the MAMS signaling mechanism for negotiation of network paths and flow protocols between the client and the network. In this example scenario, the client is connected to two networks (say LTE and WiFi).

1. UE connects to network 1 and gets an IP address assigned by network 1.
2. CCM communicates with NCM functional element via the network 1 connection and exchanges capabilities and parameters for MAMS operation. Note: The NCM credentials (e.g. NCM IP Address) can be made known to the UE by pre-provisioning.
3. Client sets up connection with network 2 and gets an IP address assigned by network 2.
4. CCM and NCM negotiate capabilities and parameters for establishment of network paths, which are then used to configure user plane functions N-MADP at the network and C-MADP at the client.
 - 4a. CCM and NCM negotiate network paths, flow routing and aggregation protocols, and related parameters.
 - 4b. NCM communicates with the N-MADP to exchange and configure flow aggregation protocols, policies and parameters in alignment with those negotiated with the CCM.
 - 4c. CCM communicates with the C-MADP to exchange and configure flow aggregation protocols, policies and parameters in alignment with those negotiated with the NCM.
5. C-MADP and N-MADP establish the user plane paths, e.g. using IKE [RFC7296] signaling, based on the negotiated flow aggregation protocols and parameters specified by NCM.

CCM and NCM can further exchange messages containing access link measurements for link maintenance by the NCM. NCM evaluates the link conditions in the UL and DL across LTE and WiFi, based on link measurements reported by CCM and/or link probing techniques and determines the UL and DL user data distribution policy. NCM and CCM also negotiate application level policies for categorizing applications, e.g. based on DSCP, Destination IP address, and determining which of the available network paths, needs to be used for transporting data of that category of applications. NCM configures N-MADP, and CCM configures C-MADP, based on the negotiated application policies. CCM may apply local application policies, in addition to the application policy conveyed by the NCM.

10. Relation to IETF Technologies

MAMS leverages technologies developed in IETF like MPTCP, GRE and enables a control plane framework to negotiate the use of these protocols between the client and the network. It also addresses the limitations in the scope of other multihoming protocols. E.g. MOBIKE RFC 4555 (IKEv2 Mobility and Multihoming Protocol (MOBIKE)) scope indicates that it is limited to multihoming between IPsec clients (tunnel mode IPsec SAs) ONLY and does NOT support load balancing. MAMS addresses this limitation in handling multihoming scenario by supporting load balancing with simultaneous use of multiple access paths by negotiating use of protocols like MPTCP. Unlike MOBIKE, which only applies to end points connected with IPsec tunnel mode SA, MAMS allows flexibility to use a wide range of tunneling protocols to be used in the Adaptation layer.

11. Applying MAMS Control Procedures with MPTCP Proxy as User Plane

If NCM determines that N-MADP is to be instantiated with MPTCP as the MX Convergence Protocol, it exchanges the MPTCP capability support in discovery and capability exchange procedures. NCM then exchanges the credentials of the N-MADP instance, setup as MPTCP Proxy, along with related parameters to the CCM. CCM configures C-MADP with these parameters to connect with the N-MADP, MPTCP proxy (e.g. [I-D.ietf-tcpm-convergers]) instance, on the available network path (Access).

Figure 17 illustrates the user plane protocol layering when MPTCP is configured to be the "MX Convergence Sublayer" protocol. MPTCP manages traffic distribution and aggregation over multiple delivery connections.

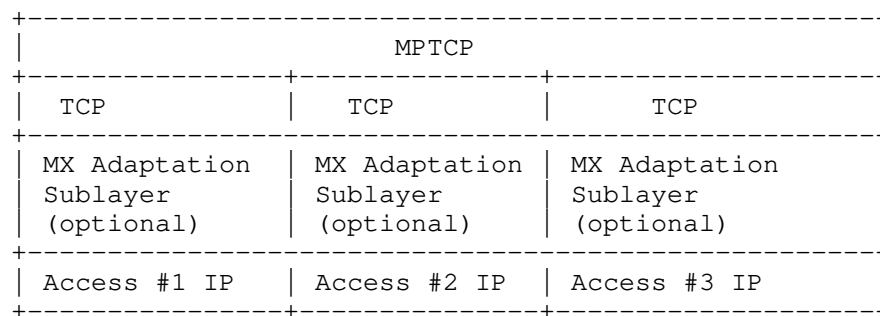


Figure 17: MAMS U-plane Protocol Stack with MPTCP as MX Convergence Layer

The Client (C-MADP) sets up an MPTCP connection with the N-MADP to begin with. MAMS control procedures are then applied to,

- o Connect to the appropriate MPTCP network endpoint, e.g. MPTCP Proxy (illustrated in Figure 18)
- o Control addition of second TCP subflow after WiFi connection is established and is deemed good, (illustrated in illustrated in Figure 19),
- o Control the MPTCP scheduler behavior like use of only LTE Subflow in UL and both LTE and WiFi subflows in DL (illustrated in illustrated in Figure 20),
- o Faster response to WiFi link degradation by proactive deletion of TCP subflow over WiFi when poor link conditions are reported, to maintain performance (illustrated in illustrated in Figure 21)

Figure 18 shows the call flow describing MAMS control procedures applied to configure user plane and dynamic optimal path selection in a scenario with MPTCP Proxy as the convergence protocol in the user plane.

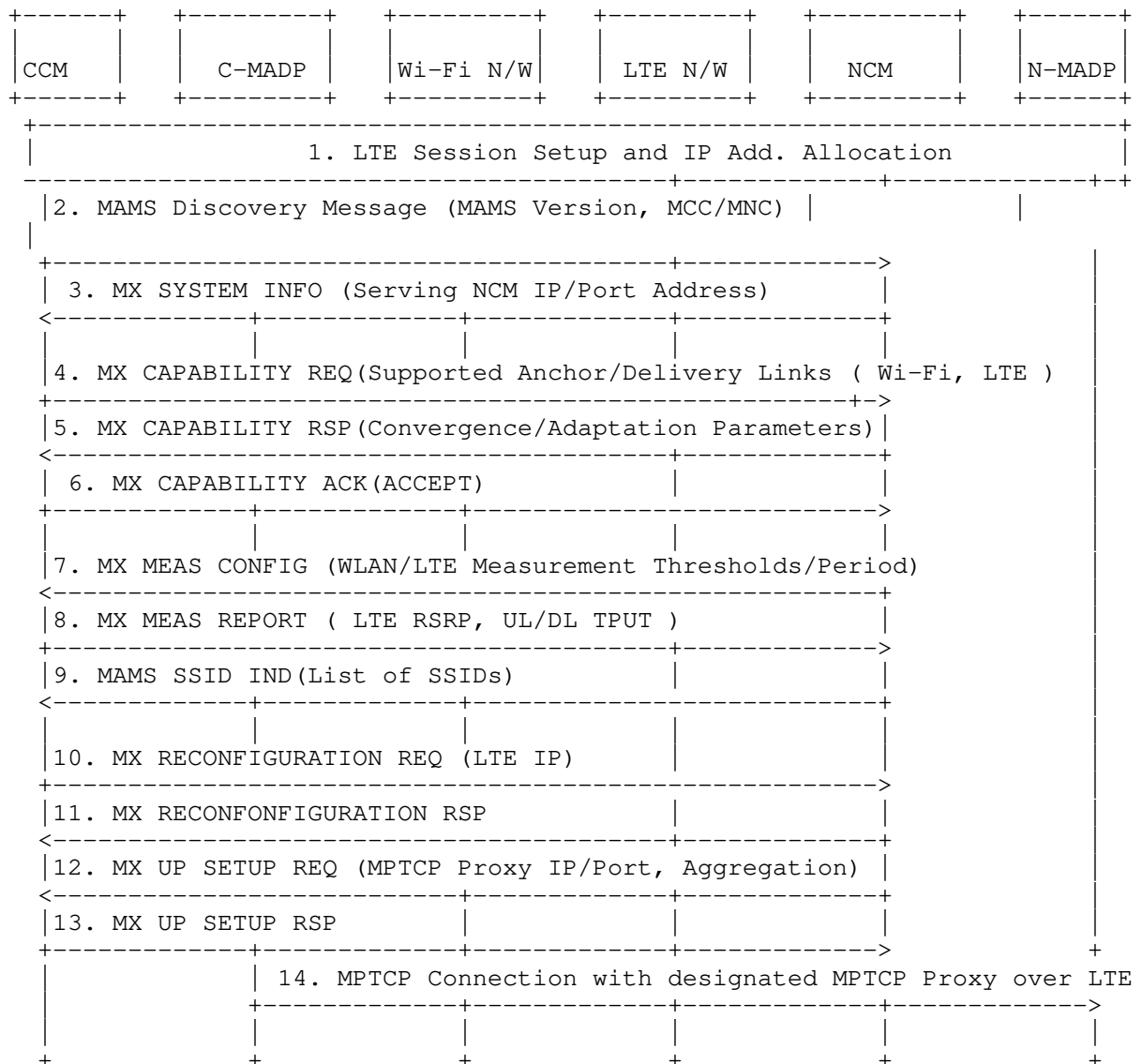


Figure 18: MAMS-assisted MPTCP Proxy as User Plane - Initial Setup with LTE leg

Following are the salient steps described in the call flow. The client connects to the LTE network and obtains an IP address (assume LTE is the first connection), and initiates the NCM discovery procedures and exchange capabilities, including the support for MPTCP as the convergence protocol at both the network and the client.

The CCM informs the LTE connection parameters to the NCM. NCM provides the parameters like MPTCP Proxy IP address/Port, MPTCP Client Key for configuring the convergence layer. This is useful if N-MADP is reachable via different IP address or/and port, from different access networks. The current MPTCP signaling can't identify or differentiate the MPTCP proxy IP address and port among multiple access networks. The client uses the MPTCP Client Key during the subflow creation, and this enables the N-MADP to uniquely identify the client, even if NAT is present. The N-MADP then can inform the NCM of the subflow creation and parameters related to creating additional subflows. Since LTE is the only connection, the user plane traffic, flows over the single TCP subflow over the LTE connection. Optionally, NCM provides assistance information to the device on the neighboring/preferred Wi-Fi networks that it can associate with.

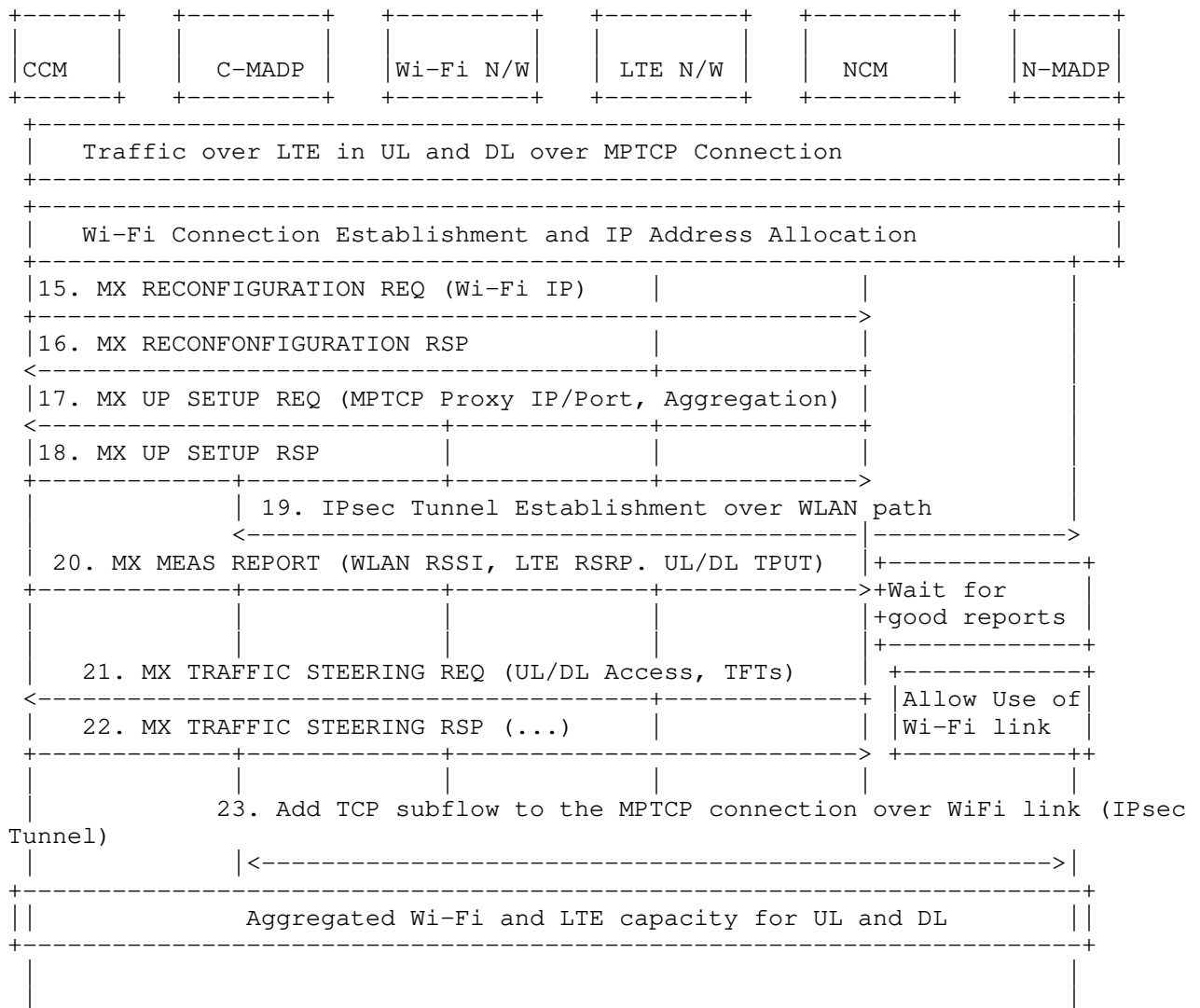


Figure 19: MAMS-assisted MPTCP Proxy as User Plane - Add Wi-Fi leg

Figure 19 describes the steps, when the client establishes a Wi-Fi connection. CCM informs the NCM of the Wi-Fi connection along with parameters like the Wi-Fi IP address, SSID. NCM determines that the Wi-Fi connection needs to be secured and configures the Adaptation Layer to be IPsec and provides the required parameters to the CCM. In addition, NCM provides the information to configure the convergence layer, (e.g. MPTCP Proxy IP Address), and provides the Traffic Steering Request to indicate that client should use only the

LTE access. NCM may do this, for example, on determination from the measurements that the Wi-Fi link is not consistently good enough. As the Wi-Fi link conditions improve, NCM sends a Traffic Steering Request to use Wi-Fi access as well. This triggers the client to establish the TCP subflow over the Wi-Fi link with the MPTCP proxy

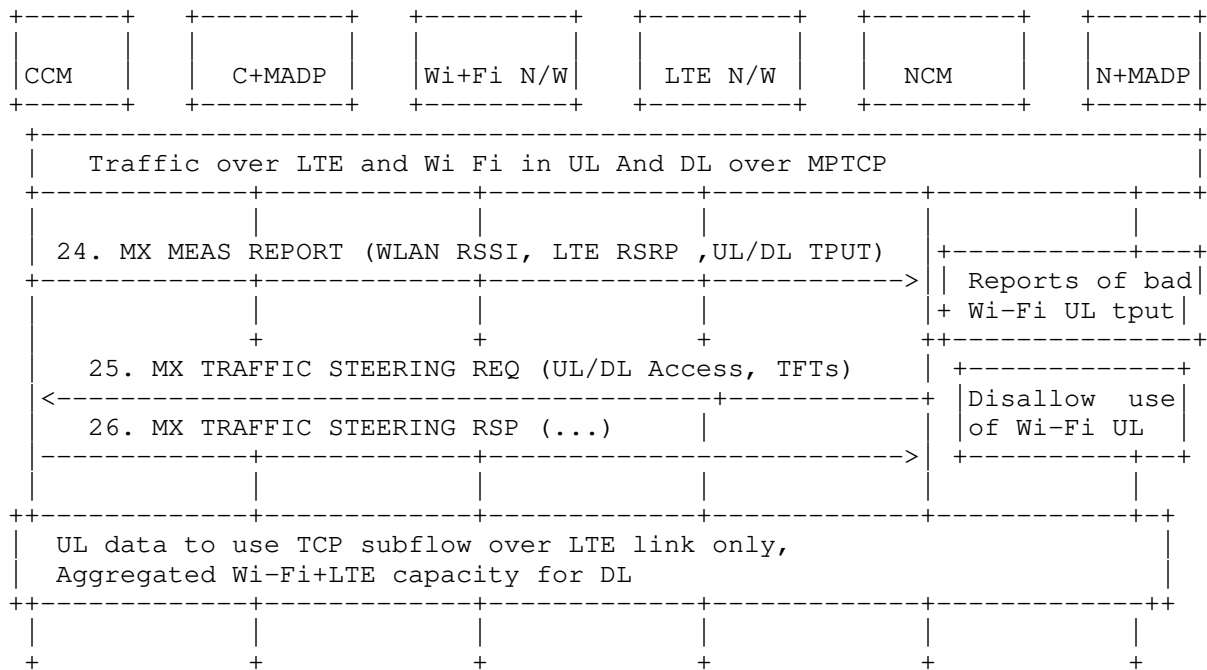


Figure 20: MAMS-assisted MPTCP Proxy as User Plane - Wi-Fi UL degrades

Figure 20 describes the steps, when the client reports that Wi-Fi link conditions degrade in UL. MAMS control plane is used to continuously monitor the access link conditions on Wi-Fi and LTE connections. The NCM may at some point determine increase in UL traffic on Wi-Fi, and trigger the client to only LTE in the UL via Traffic Steering Request to improve UL performance.

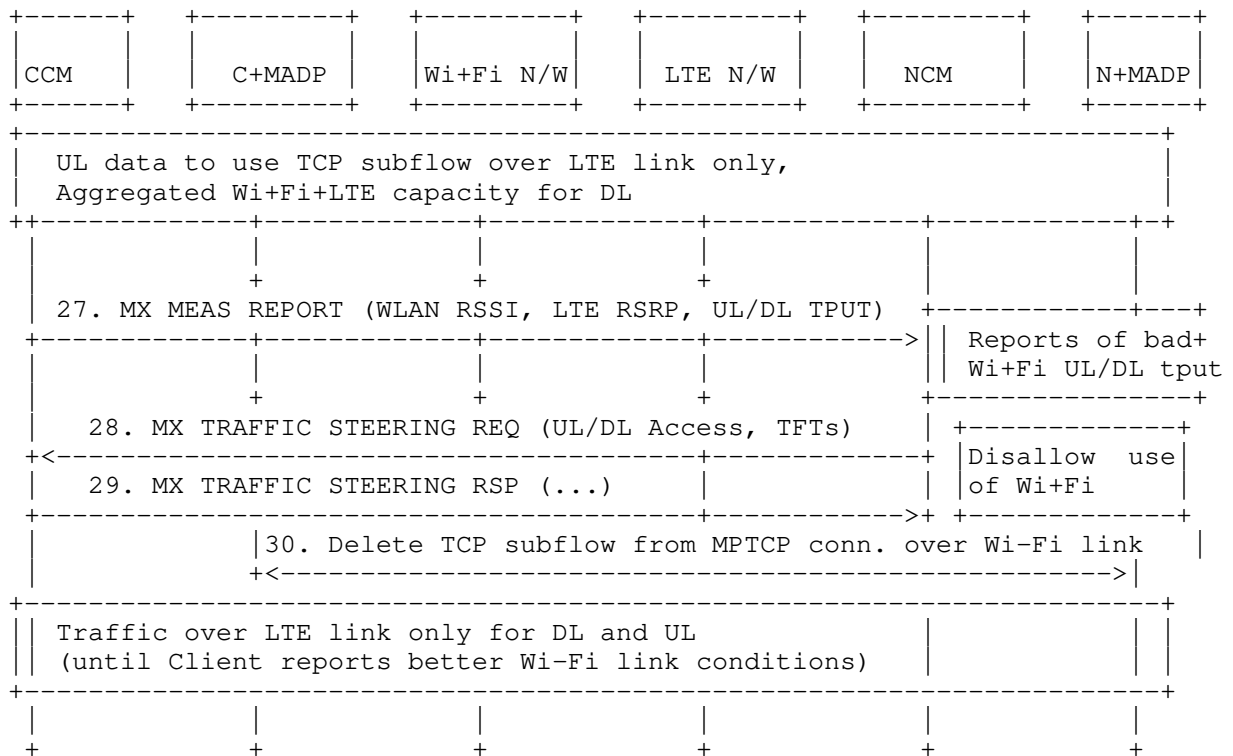


Figure 21: MAMS-assisted MPTCP Proxy as User Plane - Part 4

Figure 21 describes the steps, when the client reports that Wi-Fi link conditions degrade in both UL and DL. As the Wi-Fi link conditions deteriorate further, the NCM may determine to send Traffic Steering Request guiding the client to stop using Wi-Fi, and to use only LTE access in both UL and DL. This condition may be maintained until NCM determines, based on reported measurements that Wi-Fi link has become usable.

12. Applying MAMS Control Procedures for Network Assisted Traffic Steering when there is No Convergence Layer

Figure 22 shows the call flow describing MAMS control procedures applied for dynamic optimal path selection in a scenario convergence and Adaptation layer protocols are not omitted. This scenario indicates the applicability of a MAMS Control Plane only solution.

In the capability exchange messages, NCM and CCM negotiate that Convergence and Adaptation layer protocols are not needed (or

supported). CCM informs the NCM of the availability of the LTE and Wi-Fi links. NCM determines the access links, Wi-Fi or LTE to be used dynamically based on the reported link quality measurements.



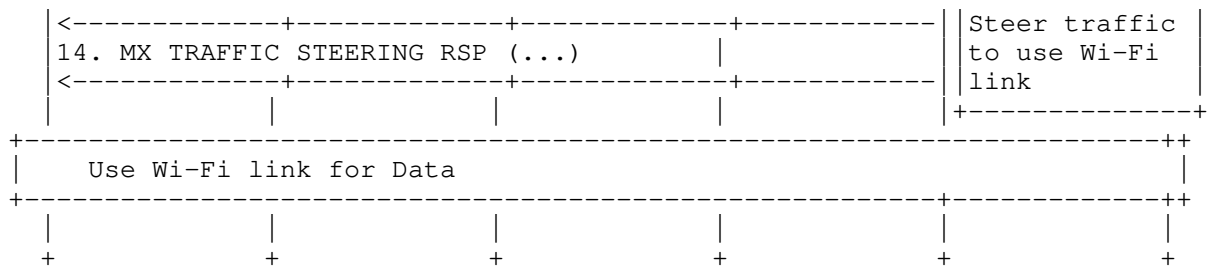


Figure 22: MAMS With No Convergence Layer

13. Co-existence of MX Adaptation and MX Convergence Layers

MAMS user plane supports multiple instances and combinations of protocols to be used at the MX Adaptation and the Convergence layer.

For example, one instance of the MX Convergence Layer can be MPTCP Proxy and another instance can be GMA. The MX Adaptation for each can be either UDP tunnel or IPsec. IPsec may be set up when network path needs to be secured, e.g. to protect the TCP subflow traversing the network path between the client and MPTCP proxy.

Each of the instances of MAMS user plane, i.e. combination of MX Convergence and MX Adaptation layer protocols, can coexist simultaneously and independently handle different traffic types.

14. Security Considerations

14.1. MAMS Control Plane Security

The NCM functional element is hosted on a network node which is assumed to be within a secure network, e.g. within the operator's network, and is assumed to be protected against hijack attacks.

For deployment scenarios, where the client is configured (e.g. by the network operator) to use a specific network path for exchanging control plane messages and if the network path is assumed to be secure, MAMS control messages will rely on security provided by the underlying network.

For deployment scenarios where the security of the network path cannot be assumed, NCM and CCM implementations MUST support the "wss" URI scheme [RFC6455] and Transport Layer Security (TLS) [RFC5246] to secure control plane message exchange between the NCM and CCM.

For deployment scenarios where client authentication is desired, the WebSocket server can use any client authentication mechanisms available to a generic HTTP server, such as cookies, HTTP authentication, or TLS authentication.

14.2. MAMS User Plane Security

User data in MAMS framework relies on the security of the underlying network transport paths. When this cannot be assumed, NCM configures use of protocols, like IPsec [RFC4301] [RFC3948] in the MX Adaptation Layer, for security.

15. Implementation Considerations

MAMS builds on commonly available functions available on terminal devices that can be delivered as a software update over the popular end-user device operating systems, enabling rapid deployment and addressing the large deployed device base.

16. Applicability to Multi Access Edge Computing

Multi-access Edge Computing (MEC), previously known as Mobile Edge Computing, is an access-edge cloud platform being considered at ETSI [ETSIMEC], whose initial focus was to improve quality of experience by leveraging intelligence at the cellular (e.g., 3GPP technologies like LTE) access edge, and the scope is now being extended to support access technologies beyond 3GPP. The applicability of the framework described in this document to the MEC platform has been evaluated and tested in different network configurations by the authors.

The NCM can be hosted on a MEC cloud server that is located in the user plane path at the edge of the multi-technology access network. The NCM and CCM can negotiate the network path combinations based on application needs and the necessary user plane protocols to be used across the multiple paths. The network conditions reported by the CCM to the NCM can be complemented by a Radio Analytics application [ETSIRNIS] residing at the MEC to configure the uplink and downlink access paths according to changing radio and congestion conditions.

The user plane functional element, N-MADP, can either be collocated with the NCM at the MEC cloud server (e.g., MEC hosted applications), or placed at a separate network element like a common user plane gateway across the multiple networks.

Also, even in scenarios where N-MADP is not deployed, the NCM can be used to augment the traffic steering decisions at the device.

The aim of these enhancements is to improve the end-user's quality of experience by leveraging the best network path based on application needs and network conditions, and building on the advantages of significantly reduced latency and the dynamic and real-time exposure of radio network information available at the MEC.

17. Related work in other Industry and Standards Forums

The MAMS framework described in this document has been incorporated as a solution to address multi access integration in multiple industry forums and standards. This section describes the related work in other industry forums and the standards organizations.

Wireless Broadband Alliance Industry partners have published a whitepaper that describes applicability of different technologies for multi access integration to different deployments as part of their project named, Unlicensed Integration with 5G Networks [WBAUnl5G]. The whitepaper includes MAMS framework described in this document as a technology for integrating Unlicensed (WiFi) networks with 5G networks above the 5G core network.

3GPP is developing a technical report as part of its work item Study on Access Traffic Steering, Switching and Splitting (ATSSS). That report, TR23.793 [GPPATSSS], contains a number of potential solutions and Solution 1 utilizes a separate control plane for flexible negotiation of user plane protocols and path measurements in a way that is similar the MAMS architecture described in this document.

The Small Cell Forum (SCF) [SCFTECH5G] plans to develop a while paper as part of its work item on LTE/5G and WiFi. There is a proposal to include MAMS in this whitepaper.

The ETSI Multi-access Edge Computing Phase 2 technical work is examining many aspects of this work including use cases for optimizing Quality of Experience (QoE) and resource utilization. The MAMS architecture and procedures outlined in this document is included in the use cases and requirements document[ETSIMAMS].

18. Contributing Authors

The editors gratefully acknowledge the following additional contributors in alphabetical order: A Krishna Pramod/Nokia, Hannu Flinck/Nokia, Hema Pentakota/Nokia, Nurit Sprecher/Nokia, Salil Agarwal/Nokia; Shuping Peng/Huawei, Vasudevan Subramanian/Nokia. Vasudevan Subramanian has been instrumental in conceptualization and development of solution principles for the MAMS framework. Shuping Peng has been a key contributor in refining the framework and control plane protocol aspects.

19. Acknowledgments

This protocol is the outcome of work by many engineers, not just the authors of this document. In alphabetical order, the contributors to the project are: Barbara Orlandi, Bongho Kim, David Lopez-Perez, Doru Calin, Jonathan Ling, Lohith Nayak, Michael Scharf.

20. IANA Considerations

This draft makes no requests of IANA

21. References

21.1. 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>>.
- [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>>.
- [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>>.

21.2. Informative References

- [ANDSF] "TS 24.312 version 15.0, 21 June 2018: 3GPP Specification on Access Network Discovery and Selection Function (ANDSF) Management Object (MO)", http://www.3gpp.org/ftp//Specs/archive/24_series/24.312/24312-f00.zip", <TS24.312>.
- [E212] "ITU-T E.212: The international identification plan for public networks and subscriptions, <https://www.itu.int/rec/T-REC-E.212-201609-I/en>", <ITU-T E.212>.
- [ETSIMAMS] "Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements, https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/02.01.01_60/gs_MEC002v020101p.pdf", <ETSI GS MEC 002>.

- [ETSI MEC] "Multi-access Edge Computing (MEC), ETSI",
<<https://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing>>.
- [ETSI RNIS] "Mobile Edge Computing (MEC) Radio Network Information API", <ETSI GS MEC 012>.
- [GPP TS] "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on Access Traffic Steering, Switching and Splitting support in the 5G system architecture (Release 16),
<https://www.3gpp.org>, work in progress", <TR23.793>.
- [I-D.deconinck-multipath-quic] Coninck, Q. and O. Bonaventure, "Multipath Extension for QUIC", draft-deconinck-multipath-quic-00 (work in progress), October 2017.
- [I-D.ietf-tcpm-convergers] Bonaventure, O., Boucadair, M., Gundavelli, S., Seo, S., and B. Hesmans, "0-RTT TCP Converge Protocol", draft-ietf-tcpm-convergers-06 (work in progress), March 2019.
- [I-D.zhu-intarea-gma] Zhu, J. and S. Kanugovi, "Generic Multi-Access (GMA) Convergence Encapsulation Protocols", draft-zhu-intarea-gma-02 (work in progress), March 2019.
- [I-D.zhu-intarea-mams-user-protocol] Zhu, J., Seo, S., Kanugovi, S., and S. Peng, "User-Plane Protocols for Multiple Access Management Service", draft-zhu-intarea-mams-user-protocol-07 (work in progress), April 2019.
- [IEEE] "IEEE Standard for Information technology: Telecommunications and information exchange between systems Local and metropolitan area networks: Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.", <IEEE 802.11-2016>.
- [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>>.

- [RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <<https://www.rfc-editor.org/info/rfc2890>>.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <<https://www.rfc-editor.org/info/rfc3948>>.
- [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>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [SCFTECH5G] "Small Cell Forum, <https://scf.io/>", <Small Cell Forum>.
- [WBAUnl5G] "Wireless Broadband Alliance Project - Unlicensed Integration with 5G Networks, <https://www.wballiance.com/resource/unlicensed-integration-with-5g-networks/>.", <Unlicensed Integration with 5G Networks>.

Appendix A. MAMS Control Plane Optimization over Secure Connections

If the connection between CCM and NCM over which the MAMS control plane messages are transported is assumed to be secure, UDP is used as the transport for management & control messages between NCM and UCM (see Figure 23).

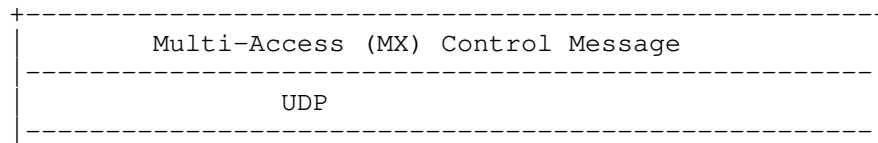


Figure 23: UDP-based MAMS Control plane Protocol Stack

Appendix B. MAMS Application Interface

B.1. Overall Design

CCM hosts an HTTPS server for applications to communicate and request services. It is assumed in this draft that all CCM and the communicating Application instances are hosted in a single administrative domain from security point of view.

The content of messages is defined in "Java Script Object Notation" (JSON) format. They offer RESTful APIs for communication.

The exact mechanism regarding how Application knows about the end point of CCM is not covered as part of this document. It may be provided as part of the Application Settings.

B.2. Notation

The documentation of APIs are provided in OpenAPI format using swagger v2.0 (TBD - Add section in appendix)

B.3. Error Indication

For every API, there could be an error response in case the objective of API could not be met as defined in [RFC2616].

B.4. CCM APIs

The following sections describe the APIs exposed by CCM to the applications

B.4.1. Get Capabilities

The CCM provides a HTTPS GET interface as `"/ccm/v1.0/capabilities"` for the Application to query about the capabilities supported by the CCM instance.

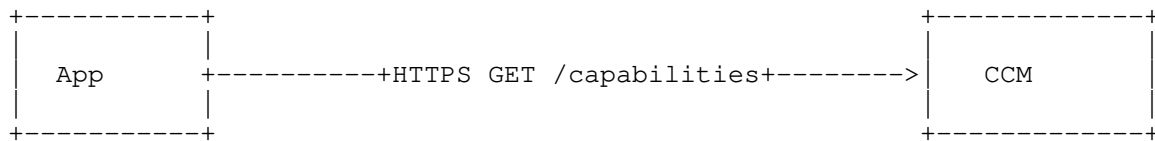


Figure 24: CCM API - GET Procedures

The CCM shall provide information of its capabilities as follows:

- o Supported Features: One of more of as defined in MX Feature Activation List parameter of MX Capability REQ.
- o Supported Connections: Supported connection types and connection IDs
- o Supported MX Adaptation Sublayers: List of MX adaptation sublayer protocols supported by the N-MADP instance alongwith the connection type where these are supported and their respective parameters.
- o Supported MX Convergence Sublayers: List of supported MX Convergence Sublayer protocols alongwith the parameters associated with respective convergence technique.

B.4.2. Post App Requirements

The CCM provides a HTTPS POST interface as `"/ccm/v1.0/app_requirements"` for the Application to post the needs of the application data streams to the CCM instance.



Figure 25: CCM API - POST Procedures

The CCM shall provide for the application to post the following requirements of its different data streams:

- o Number of data stream types For each data stream type, the following link feature preferences,
- o Protocol Type: Transport layer protocol associated with the application data stream packets.
- o Port Range: Supported connection types and connection IDs.

- o Traffic QoS: Quality of service parameters as follows

- * Bandwidth
- * Latency
- * Jitter

B.4.3. Get Predictive Link Parameters

The CCM provides a HTTPS GET interface as `"/ccm/v1.0/predictive_link_params"` for the Application to get the predicted link parameters from the CCM instance.

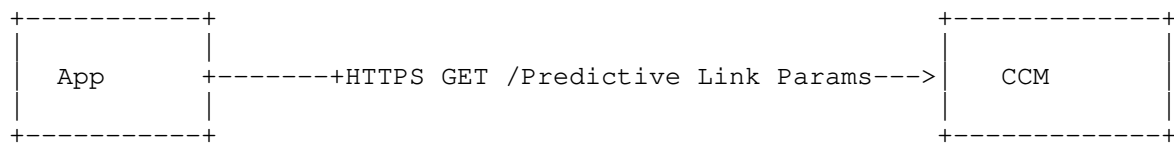


Figure 26: CCM API - GET Predictive Link Parameters Procedures

CCM requests the NCM for link parameters via the MAMS Network Analytics Request Procedure (Section 8.12) and includes the information in response to the API invocation.

- o Number of Delivery Connections For Each Delivery Connection,

- * Access Link Identifier
 - + Connection Type
 - + Connection ID
- * Link Quality Indicator
 - + Bandwidth
 - Predicted Value (in Mbps)
 - - Likelihood (in Percentage)
 - Prediction Validity (Validity Time in s)
 - + Jitter
 - Predicted Value (in s)
 - - Likelihood (in Percentage)
 - Prediction Validity (Validity Time in s)
 - + Latency
 - Predicted Value (in s)

- - Likelihood (in Percentage)
- Prediction Validity (Validity Time in s)
- + Signal Quality
- if Delivery Connection Type is LTE, LTE_RSRP Predicted Value (in dBm)
- if Delivery Connection Type is LTE, LTE_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRP Predicted Value (in dBm)
- if Delivery Connection Type is NR, NR_RSRQ Predicted Value (in dBm)
- if Delivery Connection Type is WiFi, WLAN_RSSI Predicted Value (in dBm)
- - Likelihood (in Percentage)
- Prediction Validity (Validity Time in s)

Appendix C. JSON Specification for MAMS Control Plane

MAMS Control plane messages are exchanged between the CCM and the NCM. This section, specifies the format and content of messages in "Java Script Object Notation" (JSON) format.

C.1. Protocol Specification: General Processing

C.1.1. Notation

This document uses JSONString, JSONNumber, and JSONBool to indicate the JSON string, number, and boolean types, respectively. The type JSONValue indicates a JSON value, as specified in Section 3 of [RFC7159].

This document uses an adaptation of the C-style struct notation to define JSON objects. A JSON object consists of name/value pairs. This document refers to each pair as a field. In some context, this document also refers to a field as an attribute. The name of a field/attribute may be referred to as the key. An optional field is enclosed by []. In the definitions, the JSON names of the fields are case sensitive. An array is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of values and n is the maximum. When this document uses * for n, it means no upper bound.

For example, the definition below defines a new type Type4, with three fields named "name1", "name2", and "name3", respectively. The field named "name3" is optional, and the field named "name2" is an array of at least one value.

```
object { Type1 name1; Type2 name2<1..*>; [Type3 name3;] } Type4;
```

This document uses subtyping to denote that one type is derived from another type. The example below denotes that `TypeDerived` is derived from `TypeBase`. `TypeDerived` includes all fields defined in `TypeBase`. If `TypeBase` does not have a field named "name1", `TypeDerived` will have a new field named "name1". If `TypeBase` already has a field named "name1" but with a different type, `TypeDerived` will have a field named "name1" with the type defined in `TypeDerived` (i.e., `Type1` in the example).

```
object { Type1 name1; } TypeDerived : TypeBase;
```

Note that, despite the notation, no standard, machine-readable interface definition or schema is provided in this document. Extension documents may describe these as necessary.

C.1.2. Discovery Procedure

C.1.2.1. MX Discovery Message

This message is the first message sent by CCM to discover the presence of NCM in the network. It contains only the base information as described in Appendix C.2.1 with `message_type` set as `mx_discover`.

Following is the representation of the message:

```
object {  
  
  [JSONString MCC_MNC_Tuple;]  
  
  } MXDiscover : MXBase;
```

C.1.3. System Information Procedure

C.1.3.1. MX System Information Message

This message is sent by NCM to CCM to inform the endpoints that NCM supports for MAMS functionality. In addition to base information (Appendix C.2.1) it contains following information:

- a) NCM Connections (described in Appendix C.2.3)

Following is the representation of the message:

```
object {
```



```
NCMConnections ncm_connections;  
  
} MXSystemInfo : MXBase;
```

C.1.4. Capability Exchange Procedure

C.1.4.1. MX Capability Request

This message is sent by CCM to NCM to indicate the capabilities of the CCM instance available to the NCM indicated in System Info message earlier. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Features Activation Status: Described in Appendix C.2.5
- (b) Number of anchor connections: Number of anchor connection (towards core) supported by the NCM.
- (c) Anchor Connections: Described in sec Appendix C.2.6
- (d) Number of Delivery connections: Number of delivery connection (towards access) supported by the NCM.
- (e) Delivery Connections: Described in Appendix C.2.7
- (f) Convergence Methods: Described in Appendix C.2.9
- (g) Adaptation Methods: Described in Appendix C.2.10

Following is the representation of the message:

```
object {  
  FeaturesActive feature_active;  
  JSONNumber num_anchor_connections;  
  AnchorConnections anchor_connections;  
  JSONNumber num_delivery_connections;  
  DeliveryConnections delivery_connections;  
  ConvergenceMethods convergence_methods;  
  AdaptationMethods adaptation_methods  
} MXCapabilityReq : MXBase;
```

C.1.4.2. MX Capability Response

This message is sent by NCM to CCM to indicate the capabilities of the NCM instance and unique session identifier for CCM. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Features Activation Status: Described in Appendix C.2.5
- (b) Number of anchor connections: Number of anchor connection (towards core) supported by the NCM.
- (c) Anchor Connections: Described in Appendix C.2.6
- (d) Number of Delivery connections: Number of delivery connection (towards access) supported by the NCM.

- (e) Delivery Connections: Described in Appendix C.2.7
- (f) Convergence Methods: Described in Appendix C.2.9
- (g) Adaptation Methods: Described in Appendix C.2.10
- (h) Unique Session Id: This uniquely identifies the session between CCM and NCM in a network. Described in Appendix C.2.2

Following is the representation of the message:

```
object {  
  FeaturesActive feature_active;  
  JSONNumber num_anchor_connections;  
  AnchorConnections anchor_connections;  
  JSONNumber num_delivery_connections;  
  DeliveryConnections delivery_connections;  
  ConvergenceMethods convergence_methods;  
  AdaptationMethods adaptation_methods  
  UniqueSessionId unique_session_id;  
} MXCapabilityRsq : MXBase;
```

C.1.4.3. MX Capability Acknowledge

This message is sent by CCM to NCM to indicate acceptance of capabilities advertised by NCM in earlier MX Capability Response message. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
- (b) Capability Acknowledgement: Either Accept or Reject of the capabilities sent by CCM. Can take either "MX_ACCEPT" or "MX_REJECT" as acceptable values.

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
  JSONString capability_ack;  
} MXCapabilityAck : MXBase;
```

C.1.5. User Plane Configuration Procedure

C.1.5.1. MX User Plane Configuration Request

This message is sent by NCM to CCM to configure the user plane for MAMS. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Number of Anchor Connection: Number of anchor connections supported by NCM.
- (b) Setup of Anchor Connections: Described in Appendix C.2.11.

Following is the representation of the message:

```
object {  
  JSONNumber num_anchor_connections;  
  SetupAnchorConns anchor_connections;  
} MXUPSetupConfigReq : MXBase;
```

C.1.5.2. MX User Plane Configuration Confirmation

This message is the confirmation of user plane setup message sent from CCM after successfully configuring the user plane at user equipment. This message contains following information:

- (a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
- (b) MX Probe Parameters (included if probing is supported)
 - (1) Probe Port: UDP port for accepting probe message.
 - (2) Anchor connection Id: Identifier of the anchor connection to be used for probe function, provided in user plane setup request.
 - (3) MX Configuration Id: For the given anchor connection, which configuration id is to be used for probe, this is present only if provided in the user plane setup request.
- (c) For each delivery connection following is required:
 - (1) Connection ID: Delivery connection id supported by UE.
 - (2) Client Adaptation Layer Parameters: If UDP adaptation layer is in use then the UDP port to be used at C-MADP side.

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
  [ProbeParam probe_param;]  
  JSONNumber num_delivery_conn;  
  ClientParam client_params <1...*>;  
} MXUPSetupConfigCnf : MXBase;
```

Where ProbeParam is defined as following:

```
object {  
  JSONNumber probe_port;  
  JSONNumber anchor_conn_id;
```

```
[JSONNumber mx_configuration_id;]
} ProbeParam;
```

Where ClientParam is defined as following:

```
object {
JSONNumber connection_id;
[AdaptationParam adapt_param;]
} ClientParam;
```

Where AdaptationParam is defined as following:

```
object {
JSONNumber udp_adapt_port;
} AdaptationParam;
```

C.1.6. Reconfiguration Procedure

C.1.6.1. Reconfiguration Request

This message is sent by CCM to NCM in case of reconfiguration of any the connections from user equipment's side. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Unique Session Id: Identifier for CCM-NCM association
Appendix C.2.2.
- (b) Reconfiguration Action: Type of reconfiguration action can be one of "setup", "release" or "modify".
- (c) Connection Id: Connection Id for which the reconfiguration is taking place.
- (d) IP address: IP address in case of setup and modify type of reconfiguration.
- (e) SSID: If the connection type is WiFi, in that case the SSID the UE has attached to is contained in this parameter.
- (f) MTU of connection: MTU of the delivery path that is calculated at the UE for use by NCM to configure fragmentation and concatenation procedures at N-MADP.
- (g) Connection Status: This parameter informs if the connection is currently "disabled", "enabled" or "connected". Default: "connected".
- (h) Delivery Node Id: Identity of the node to which the client is attached. ECGI in case of LTE and WiFi AP Id or MAC address in case of WiFi.

Following is the representation of the message:

```
object {
UniqueSessionId unique_session_id;
```

```
JSONString reconf_action;  
JSONNumber connection_id;  
JSONString ip_address;  
JSONString ssid;  
JSONNumber mtu_size;  
JSONString connection_status;  
[JSONString delivery_node_id;]  
} MXReconfReq : MXBase;
```

C.1.6.2. Reconfiguration Response

This message is sent by NCM to CCM as a confirmation towards reconfiguration requirement after taking the reconfiguration into use and contains only the base information (as defined in Appendix C.2.1).

Following is the representation of the message:]

```
object {  
} MXReconfRsp : MXBase;
```

C.1.7. Path Estimation Procedure

C.1.7.1. Path Estimation Request

This message is sent by NCM towards CCM to configure the CCM to send path estimation reports. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Connection Id: Id of the connection for which the path estimation report is required.
- (b) Init Probe Test Duration: Duration of initial probe test in milliseconds. [TBD: Range of values]
- (c) Init Probe Test Rate: Initial testing rate in Mega Bits per Second. [TBD: Range of values]
- (d) Init Probe Size: Size of each packet for initial probe in Bytes. [TBD: Range of values]
- (e) Init Probe Ack: If an acknowledgement for probe is required. [Possible values: "yes", "no"]
- (f) Active Probe Frequency: Frequency in milliseconds at which the active probes shall be sent. [TBD: Range of values]
- (g) Active Probe Size: Size of the active probe in Bytes. [TBD: Range of values]
- (h) Active Probe Duration: Duration in seconds for which the active probe shall be performed. [TBD. Range of values]
- (i) Active Probe Ack. If an acknowledgement for probe is required. [Possible values: "yes", "no"]

Following is the representation of the message:

```
object {  
  JSONNumber connection_id;  
  JSONNumber init_probe_test_duration_ms;  
  JSONNumber init_probe_test_rate_Mbps;  
  JSONNumber init_probe_size_bytes;  
  JSONString init_probe_ack_req;  
  JSONNumber active_probe_freq_ms;  
  JSONNumber active_probe_size_bytes;  
  JSONNumber active_probe_duration_sec;  
  JSONString active_probe_ack_req;  
} MXPathEstReq : MXBase;
```

C.1.7.2. Path Estimation Report

This message is sent by CCM to NCM as report to the probe estimation configured by NCM. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
- (b) Connection Id: Id of the connection for which the path estimation report is required.
- (c) Init Probe Results: Defined in section Appendix C.2.12.
- (d) Active Probe Results: Defined in section Appendix C.2.13.

Following is the representation of the message:

```
object {  
  JSONNumber connection_id;  
  UniqueSessionId unique_session_id;  
  [InitProbeResults init_probe_results;]  
  [ActiveProbeResults active_probe_results;]  
} MXPathEstResults : MXBase;
```

C.1.8. Traffic Steering Procedure

C.1.8.1. Traffic Steering Request

This message is sent by NCM to CCM for enabling traffic steering at delivery side in uplink and downlink configuration. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Connection id: Anchor connection number for which the traffic steering is getting defined.
- (b) MX Configuration Id: MX configuration for which the traffic steering is getting defined.

- (c) Downlink Delivery: Defined in Appendix C.2.14.
- (d) Default UL Delivery: The default delivery connection for uplink. All traffic should be delivered on this connection in uplink direction and the TFT filter should be applied only for the traffic mentioned in Uplink Delivery.
- (e) Uplink Delivery: Defined in Appendix C.2.15.
- (f) Features Activated: Defined in Appendix C.2.5.

Following is the representation of the message:

```
object {  
  JSONNumber connection_id;  
  [JSONNumber mx_configuration_id;]  
  DLDelivery downlink_delivery;  
  JSONNumber default_uplink_delivery;  
  ULDelivery uplink_delivery;  
  FeaturesActive feature_activation;  
} MXTraffiSteeringReq : MXBase;
```

C.1.8.2. Traffic Steering Response

This message is response to Traffic Steering request from CCM to NCM. In addition to base information (Appendix C.2.1) it contains following information:

- (a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
- (b) Features Activated: Defined in Appendix C.2.5.

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
  FeaturesActive feature_activation;  
} MXTraffiSteeringResp : MXBase;
```

C.1.9. MAMS Application MADP Association

C.1.9.1. MAMS Application MADP Association Request

This message is sent by CCM to NCM to select MADP instances provided earlier in User Plane Setup Request based on requirement of the applications.

In addition to the base information (Appendix C.2.1) it contains following:

- (a) Unique Session Id: This uniquely identifies the session between CCM and NCM in a network. Described in Appendix C.2.2, and a list of following MX Application MADP Associations
 - (a) Connection id: Defines the anchor connection number of the MADP instance
 - (b) MX Configuration Id: identify the MX configuration of the MADP instance
 - (c) Traffic Template Uplink: Traffic template as defined in 5.16 to be used in uplink direction.
 - (d) Traffic Template Downlink: Traffic template as defined in 5.16 to be used in downlink direction.

Following is the representation of the message:

```
object {
  UniqueSessionId unique_session_id;
  MXAppMADPAssoc app_madp_assoc_list <1..*>;
} MXAppMADPAssocReq : MXBase;
```

Where each measurement MXAppMADPAssoc is represented by following:

```
object {
  JSONNumber connection_id;
  JSONNumber mx_configuration_id
  TrafficFlowTemplate tft_ul_list <1..*>;
  TrafficFlowTemplate tft_dl_list <1..*>;
} MXAppMADPAssoc;
```

C.1.9.2. MAMS Application MADP Association Response

This message is sent by NCM to CCM to confirm the selected MADP instances provided in request message by CCM.

In addition to the base information (Appendix C.2.1), it contains information if the request has been successful.

Following is the representation of the message:

```
object {
  JSONBool is_success;
} MXAppMADPAssocResp : MXBase;
```

C.1.10. SSID Indication

This message is sent by NCM to CCM to indicate the list of allowed SSID which are supported by MAMS entity at the network side. It contains the list of SSIDs.

Each SSID consists of the type of SSID (which can be one of the "SSID", "BSSID" or "HESSID" and the SSID itself.

Following is the representation of the message:

```
object {  
  SSID ssid_list<1..*>;  
} MXSSIDIndication : MXBase;
```

Where each SSID is defined as following:

```
object {  
  JSONString ssid_type;  
  JSONString ssid;  
} SSID;
```

C.1.11. Measurements

C.1.11.1. Measurement Configuration

This message is sent from NCM to CCM to configure the period measurement reporting at CCM. The message contains a list of measurement configuration with each element containing following information:

- (a) Connection Id: Connection id of the delivery connection for which the reporting is being configured.
- (b) Connection Type: Connection Type for which the reporting is being configured, can be "lte", "wifi", "5g-nr" etc.
- (c) Measurement Report Configuration: Actual report configuration based on the connection type, as defined in Appendix C.2.17

Following is the representation of the message:

```
object {  
  MeasReportConf measurement_configuration <1..*>;  
} MXMeasReportConf : MXBase;
```

Where each measurement MeasReportConf is represented by following:

```
object {  
  JSONNumber connection_id;  
  JSONString connection_type;  
  MeasReportConfs meas_rep_conf <1..*>;  
} MeasReportConf;
```

C.1.11.2. Measurement Report

This message is periodically sent by CCM to NCM after measurement configuration. In addition to the base information it contains following information:

- (a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
- (b) Measurement report for each delivery connection is measured by the client device as defined in Appendix C.2.18.

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
  MXMeasRep measurment_reports <1..*>;  
} MXMeasurementReport : MXBase;
```

C.1.12. Keep Alive

C.1.12.1. Keep Alive Request

A Keep Alive Request message can be sent from either NCM or CCM on expiry of MAMS_KEEP_ALIVE timer or a handover event. This request shall be responded by the peer with Keep Alive Response. In case of no response from peer the MAMS connection shall be assumed to be broken and new connection shall be established again by CCM by sending MX Discover messages.

In addition to the base information it contains following information:

- (a) Keep Alive Reason: Reason for sending this message, can be "Timeout" or "Handover".
- (b) Unique Session Id: Identifier for CCM-NCM association Appendix C.2.2.
- (c) Connection Id: Connection id for which handover is detected, in case the reason is "Handover".
- (d) Delivery Node Id: The target delivery node id (ECGI or WiFi Access Point Id/MAC) to which the handover is executed.

Following is the representation of the message:

```
object {  
  JSONString keep_alive_reason;  
  UniqueSessionId unique_session_id;  
  JSONNumber connection_id;  
  JSONString delivery_node_id;
```

```
} MXKeepAliveReq : MXBase;
```

C.1.12.2. Keep Alive Response

On receiving Keep Alive Request from peer, NCM/CCM shall immediately respond with a Keep Alive Response message on the same delivery path from where the request arrived. In addition to base information it contains the unique session identifier for the CCM-NCM association (defined in Appendix C.2.2)

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
} MXKeepAliveResp : MXBase;
```

C.1.13. Session Termination Procedure

C.1.13.1. Session Terminate Request

In the event where NCM or CCM can no longer handle MAMS for any reason then it can send MX session termination request to the peer. In addition to base information it contains Unique Session Id and reason for termination, this can be "MX_NORMAL_RELEASE", "MX_NO_RESPONSE" or "INTERNAL_ERROR".

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
  JSONString reason;  
} MXSessionTerminationReq : MXBase;
```

C.1.13.2. Session Terminate Response

On reception of MX session termination request from peer, NCM/CCM shall respond with MX Session Termination Response on the same delivery path where the request arrived and clean the MAMS related resources and settings. CCM shall re-initiate a new session with MX Discover messages again.

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
} MXSessionTerminationResp : MXBase;
```

C.1.14. Network Analytics

C.1.14.1. MX Network Analytics Request

This message is sent by CCM to NCM to request for parameters like bandwidth, jitter, latency and signal quality predicted by network analytics function. In addition to the base information it contains following parameter:

- (a) Unique Session Id: Same identifier as provided in MX Capability RSP. Described in Appendix C.2.2.
- (b) Parameter List: List of parameters which the CCM is interested in namely one or more of, "bandwidth", "jitter", "latency" and "signal_quality".

Following is the representation of the message:

```
object {  
  UniqueSessionId unique_session_id;  
  JSONString params<1..*>;  
} MXNetAnalyticsReq : MXBase;
```

Where the params object can take one or more of the following values:

```
"bandwidth"  
"jitter"  
"latency"  
"signal_quality"
```

C.1.14.2. MX Network Analytics Response

This message is sent by NCM to CCM in response to the analytics request. For each delivery connection that the client has NCM reports the requested parameter predictions and their respective likelihood (between 1-100).

In addition to the base information it contains following parameters:

- (a) Number of delivery connections: The number of delivery connections configured for the client currently.
- (b) For each delivery connection following is provided:
 - (a) Connection Id: Connection ID of the delivery connection for which the parameters are being predicted.
 - (b) Connection Type: Type of connection, can be "Wi-Fi", "5G NR", "Multi-Fire" and "LTE".
 - (c) List of Parameters for which Prediction is requested, where each of the predicted parameters consists of following:

- (a) Parameter name: Name of the parameter being predicted. Can be one of "bandwidth", "jitter", "latency", "signal_quality".
- (b) Additional Parameter: If Parameter name is "signal_quality", then this qualifies the quality parameter like "lte_rsrp", "lte_rsrq", "nr_rsrp", "nr_rsrq", or "wifi_rssi".
- (c) Predicted value: Provides the predicted value of the parameter and, if applicable, the additional parameter.
- (d) Likelihood: Provides a stochastic likelihood of about the predicted value.
- (e) Validity Time: the time horizon until which the predictions are valid.

Following is the representation of the message:

```
object {
  MXAnalyticsList param_list<1..*>;
} MXNetAnalyticsResp : MXBase;
```

Where MXAnalyticsList is defined as following::

```
object{
  JSONNumber connection_id;
  JSONString connection_type;
  ParamPredictions predictions <1..*>;
} MXAnalyticsList;
```

Where each ParamPredictions is defined as:

```
object{
  JSONString param_name;
  [JSONString additional_param;]
  JSONNumber prediction;
  JSONNumber likelihood;
  JSONNumber validity_time;
} ParamPredictions;
```

C.2. Protocol Specification: Data Types

C.2.1. MXBase

This is the base information that every message between CCM and NCM exchanges shall have as mandatory information. It contains following information:

- (a) Version: Version of MAMS in used

- (b) Message Type: Message type being sent with following as valid values:
- (a) "mx_discover"
 - (b) "mx_system_info"
 - (c) "mx_capability_req"
 - (d) "mx_capability_resp"
 - (e) "mx_capability_ack"
 - (f) "mx_up_setup_conf_req"
 - (g) "mx_up_setup_cnf"
 - (h) "mx_reconf_req"
 - (i) "mx_reconf_rsp"
 - (j) "mx_path_est_req"
 - (k) "mx_path_est_results"
 - (l) "mx_traffic_steering_req"
 - (m) "mx_traffic_steering_rsp"
 - (n) "mx_ssid_indication"
 - (o) "mx_keep_alive_req"
 - (p) "mx_keep_alive_rsp"
 - (q) "mx_measurement_conf"
 - (r) "mx_measurement_report"
 - (s) "mx_session_termination_req"
 - (t) "mx_session_termination_resp"
 - (u) "mx_app_madp_assoc_req"
 - (v) "mx_app_madp_assoc_resp"
 - (w) "mx_network_analytics_req"
 - (x) "mx_network_analytics_resp"
- (c) Sequence Number: Sequence number to uniquely identify a transaction of message exchange, e.g. MX Capability REQ/RSP/ACK.

Following is the representation of this data type:

```
object {  
  JSONString version;  
  JSONString message_type;  
  JSONNumber sequence_num;  
} MXBase;
```

C.2.2. Unique Session Id

This data type defines the unique session id between a CCM and NCM entity, it contains a NCM id which is unique in the network and a session id allocated by NCM for that session. On reception, of discovery message if the session is existing then the old session id is returned in System Info message otherwise NCM allocates a new session id to the CCM and sends in response with System Info message.

Following is the representation of this data type:

```
object {  
  JSONNumber ncm_id;  
  JSONNumber session_id;  
} UniqueSessionId;
```

C.2.3. NCM Connections

This data type defines the connection available at NCM for MAMS connectivity towards the User Equipment. It contains a list of NCM connections available where each connection has following information:

- (a) Connection Information: As defined in Appendix C.2.4
- (b) NCM End Point information: This contains IP Address and Port exposed by NCM end point for CCM.

Following is the representation of this data type:

```
object {  
  NCMConnection items<1..*>;  
} NCMConnections;
```

where NCMConnection is defined as:

```
object {  
  NCMEndPoint ncm_end_point;  
} NCMConnection : ConnectionInfo;
```

where NCMEndPoint is defined as:

```
object {  
  JSONString ip_address;  
  JSONNumber port;  
} NCMEndPoint;
```

C.2.4. Connection Information

This data type provides the mapping of connection Id and connection type. It contains following information:

- (a) Connection Id: Number indicating the connection can be 0,1,2 and 3.
- (b) Connection type: Type of connect can be "Wi-Fi", "5G NR", "Multi-Fire" and "LTE".

The two are considered a mapping like 0-"Wi-Fi", 1-"5G NR", 2-"Multi-Fire" and 3-"LTE".

Following is the representation of this data type:

```
object {  
  JSONNumber connection_id;  
  JSONString connection_type;  
} ConnectionInfo;
```

C.2.5. Features Activation Status

This data type provides the list of all features with their activation status. Each feature status contains following:

(a) Feature Name: Name of the feature can be one of the following:

- (a) "lossless_switching"
- (b) "fragmentation"
- (c) "concatenation"
- (d) "uplink_aggregation"
- (e) "downlink_aggregation"
- (f) "measurement"

(b) Active status: Activation status of the feature, "true" means feature is active, "false" means feature is inactive.

Following is the representation of this data type:

```
object {  
  FeatureInfo items<1..*>;  
} FeaturesActive;
```

where FeatureInfo is defined as:

```
object {  
  JSONString feature_name;  
  JSONBool active;  
} FeatureInfo;
```

C.2.6. Anchor Connections

This data type contains the list of Connection Information (Appendix C.2.4) that are supported at anchor (core) side.

Following is the representation of this data type:

```
object {  
  ConnectionInfo items<1..*>;
```



```
    } AnchorConnections;
```

C.2.7. Delivery Connections

This data type contains the list of Connection Information (Appendix C.2.4) that are supported at delivery (access) side.

Following is the representation of this data type:

```
object {  
  ConnectionInfo items<1..*>;  
} DeliveryConnections;
```

C.2.8. Method Support

This data type provides the support for a particular convergence or adaptation method. It consists of following:

- (a) Method: Name of the method.
- (b) Supported: Whether the method named above is supported or not. Possible values are "true" and "false".

Following is the representation of this data type:

```
object {  
  JSONString method;  
  JSONBool supported;  
} MethodSupport;
```

C.2.9. Convergence Methods

This data type contains the list of all convergence methods and their support status. Convergence Methods possible are:

```
"GMA"  
"MPTCP_Proxy"  
"GRE_Aggregation_Proxy"  
"MPQUIC"
```

Following is the representation of this data type:

```
object {  
  MethodSupport items<1..*>;  
} ConvergenceMethods;
```

C.2.10. Adaptation Methods

This data type contains the list of all convergence methods and their support status. Converge Methods possible are:

```
"UDP_without_DTLS"  
"UDP_with_DTLS"  
"IPSec"  
"Client_NAT"
```

Following is the representation of this data type:

```
object {  
  MethodSupport items<1..*>;  
} AdaptationMethods;
```

C.2.11. Setup of Anchor Connections

This data type defines the setup configuration for each of the anchor connection that is required at the user equipment side. It contains following information in addition to the connection id and type of the anchor connection:

- (a) Number of Active MX configurations: If more than one active configurations are present for this anchor then this identifies the number of such connections
- (b) For each active configuration following convergence parameters are provided:
 - (a) MX Configuration Identifier: This identifier is present in case there are multiple active configuration and identifies the configuration for this MADP instance id.
 - (b) Convergence Method: Converge method selected, has to be one of the supported convergence method as listed in section Appendix C.2.9.
 - (c) Convergence Method Parameters: Described in section Appendix C.2.11.1
 - (d) Number of Delivery Connections: Number of delivery connections (access side) that are supported for this anchor connection.
 - (e) Setup Delivery Connections: Described in section Appendix C.2.11.2.

Following is the representation of this data type:

```
object {  
  SetupAnchorConn items<1..*>;  
} SetupAnchorConns;
```

Where each Anchor connection configuration is defined as following:

```
object {  
  [JSONNumber num_active_mx_conf;]  
  ConvergenceConfig convergence_config  
} SetupAnchorConn : ConnectionInfo;
```

where each Convergence configuration is defined as following:

```
object {  
  [JSONNumber mx_configuration_id;]  
  JSONString convergence_method;  
  ConvergenceMethodParam convergence_method_params;  
  JSONNumber num_delivery_connections;  
  SetupDeliveryConns delivery_connections;  
} ConvergenceConfig;
```

C.2.11.1. Convergence Method Parameters

This data type defines the parameters used for convergence method and contains following:

- (a) Proxy IP: IP Address of proxy that is provided by Convergence Method selected.
- (b) Proxy Port: Port of the proxy that is provided by Convergence Method selected.

Following in the representation of this data type:

```
object {  
  JSONString proxy_ip;  
  JSONString proxy_port;  
  JSONString client_key;  
} ConvergenceMethodParam;
```

C.2.11.2. Setup Delivery Connections

This is the list of delivery connections and their parameters to be configured at the user equipment. Each delivery connection defined by its connection information (Appendix C.2.4) contains optionally following:

- (a) Adaptation Method: Selected adaptation method name, this shall be one of the names as listed in Appendix C.2.10.
- (b) Adaptation Method Parameters: Depending on the adaptation method one or more of the following parameters shall be provided.
 - (a) Tunnel IP address

- (b) Tunnel Port number
- (c) Shared Secret
- (d) MX header optimization: If the adaptation method is UDP_and convergence is GMA then this flag represents if the checksum field and the length field in the IP header of a MX PDU should be recalculated or not by the MX convergence sublayer. The possible values are "true" and "false". If it is "true", both fields remain unchanged; otherwise, both fields should be recalculated. If this field is not present then the default of "false" should be considered.

Following in the representation of this data type:

```
object {  
  SetupDeliveryConn items<1..*>;  
} SetupDeliveryConns;
```

where each Setup Delivery Connection consists of following:

```
object {  
  [JSONString adaptation_method;]  
  [AdaptationMethodParam adaptation_method_param;]  
} SetupDeliveryConn : ConnectionInfo;
```

where Adaptation Method Param is defined as:

```
object {  
  JSONString tunnel_ip_addr;  
  JSONString tunnel_end_port;  
  JSONString shared_secret;  
  [JSONBool mx_header_optimization;]  
} AdaptationMethodParam;
```

C.2.12. Init Probe Results

This data type defines the results of init probe request made by NCM. It consists of following information:

- (a) Lost Probes: Percentage or probes lost.
- (b) Probe Delay: Average delay of probe message in microseconds.
- (c) Probe Rate: Probe rate achieved in Mega Bits per second.

Following in the representation of this data type:

```
object {  
  JSONNumber lost_probes_percentage;  
  JSONNumber probe_rate_Mbps;  
} InitProbeResults;
```

C.2.13. Active Probe Results

This data type defines the results of init probe request made by NCM. It consists of following information:

- (a) Average Probe Throughput: Average active probe throughput achieved in Mega Bits per second.

Following in the representation of this data type:

```
object {  
  JSONNumber avg_tput_last_probe_duration_Mbps;  
} ActiveProbeResults;
```

C.2.14. Downlink Delivery

This data type defines the list of connections which are enabled in delivery side to be used in downlink direction.

Following in the representation of this datatype:

```
object {  
  JSONNumber connection_id <1..*>;  
} DLDelivery;
```

C.2.15. Uplink Delivery

This data type defines the list of connections and parameters enabled for deliver side to be used in uplink direction.

The uplink delivery consists of multiple uplink delivery entities, where each entity consists of a traffic flow template (TFT) Appendix C.2.16 and list of connection ids in uplink where traffic qualifying for such traffic flow template can be redirected.

Following in the representation of this data type:

```
object {  
  ULDeliveryEntity ul_del <1..*>;  
} ULDelivery;
```

Where each uplink delivery entity consists of following data type:

```
object {  
  TrafficFlowTemplate ul_tft <1..*>;  
  JSONNumber connection_id <1..*>;  
} ULDeliveryEntity;
```

C.2.16. Traffic Flow Template

Traffic flow template follows in general guidelines specified in 3GPP TS 23.060.

Traffic flow template in MAMS consists of one or more of following:

- (a) Remote Address and Mask: IP address and subnet for remote addresses represented in CIDR notation. Default: "0.0.0.0/0".
- (b) Local Address and Mask: IP address and subnet for local addresses represented in CIDR notation. Default: "0.0.0.0/0"
- (c) Protocol Type: IP protocol number of the payload being carried by IP packet. e.g. UDP, TCP etc. Default: 255.
- (d) Local Port Range: Range of ports for local ports for which the flow template is applicable. Default: Start=0, End=65535.
- (e) Remote Port Range: Range of ports for remote ports for which the flow template is applicable. Default: Start=0, End=65535.
- (f) Traffic Class: Represented by Type of Service in IPv4 and Traffic Class in IPv6. Default: 255
- (g) Flow Label: Flow label for IPv6, applicable only for IPv6 protocol type. Default: 0.

Following in the representation of this data type:

```
object {  
  JSONString remote_addr_mask;  
  JSONString local_addr_mask;  
  JSONNumber protocol_type;  
  PortRange local_port_range;  
  PortRange remote_port_range;  
  JSONNumber traffic_class;  
  JSONNumber flow_label;  
} TrafficFlowTemplate;
```

Where the port range is defined as following:

```
object {  
  JSONNumber start;  
  JSONNumber end;  
} PortRange;
```

C.2.17. Measurement Report Configuration

This data type defines the configuration done by NCM towards CCM for reporting measurement events.

- (a) Measurement Report Parameter: Parameter which shall be measured and reported. This is dependent on the connection type:

- (a) For connection type "wifi" allowed measurement parameters are "WLAN_RSSI", "WLAN_LOAD", "UL_TPUT", "DL_TPUT", "EST_UL_TPUT" and "EST_DL_TPUT".
- (b) For connection type "lte" allowed measurement parameters are "LTE_RSRP", "LTE_RSRQ", "UL_TPUT" and "DL_TPUT".
- (c) For connection type "5g-nr" allowed measurement parameters are "NR_RSRP", "NR_RSRQ", "UL_TPUT" and "DL_TPUT".
- (b) Threshold: High and Low threshold for reporting.
- (c) Period: Period for reporting in milliseconds.

Following is the representation of this data type:

```
object {
  JSONString meas_rep_param;
  Threshold meas_threshold;
  JSONNumber meas_period;
} MeasReportConfs;
```

Where Threshold is defined as following:

```
object {
  JSONNumber high;
  JSONNumber low;
} Threshold;
```

C.2.18. Measurement Report

This data type defines the measurements reported by CCM for each access network measured. This type contains the connection information, delivery node id which identifies the cell (ECGI) or the WiFi Access Point Id or MAC address (or equivalent identifier in other technologies) and the actual measurement performed by CCM in the last measurement period.

Following is the representation of this data type:

```
object {
  JSONNumber connection_id;
  JSONString connection_type;
  JSONString delivery_node_id;
  Measurement measurements <1..*>;
}MXMeasRep;
```

Where Measurement is defined as the key value pair of measurement type and value. The exact type and value are defined on a per delivery type and defined in Appendix C.2.17.

```
object{
```

```
JSONString measurement_type;  
JSONNumber measurement_value;  
} Measurement;
```

C.3. Schemas in JSON

C.3.1. MX Base Schema


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {
    "message_type_def": {
      "enum": [
        "mx_discover",
        "mx_system_info",
        "mx_capability_req",
        "mx_capability_resp",
        "mx_capability_ack",
        "mx_up_setup_conf_req",
        "mx_up_setup_cnf",
        "mx_reconf_req",
        "mx_reconf_rsp",
        "mx_path_est_req",
        "mx_path_est_results",
        "mx_traffic_steering_req",
        "mx_traffic_steering_rsp",
        "mx_ssid_indication",
        "mx_keep_alive_req",
        "mx_keep_alive_rsp",
        "mx_measurement_conf",
        "mx_measurement_report",
        "mx_session_termination_req",
        "mx_session_termination_resp",
        "mx_app_madp_assoc_req",
        "mx_app_madp_assoc_resp",
        "mx_network_analytics_req",
        "mx_network_analytics_resp"
      ],
      "type": "string"
    },
    "sequence_num_def": {
      "minimum": 1,
      "type": "integer"
    },
    "version_def": {
      "type": "string"
    }
  },
  "id": "http://www.ietf.org/mams/mx_base_def.json"
}
```

C.3.2. MX Definitions

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {
```

```
"adapt_method": {
  "enum": [
    "UDP_without_DTLS",
    "UDP_with_DTLS",
    "IPSec",
    "Client_NAT"
  ],
  "type": "string"
},
"conv_method": {
  "enum": [
    "GMA",
    "MPTCP_Proxy",
    "GRE_Aggregation_Proxy",
    "MPQUIC"
  ],
  "type": "string"
},
"supported": {
  "type": "boolean"
},
"active": {
  "type": "boolean"
},
"connection_id": {
  "type": "integer"
},
"feature_name": {
  "enum": [
    "lossless_switching",
    "fragmentation",
    "concatenation",
    "uplink_aggregation",
    "downlink_aggregation",
    "measurement"
  ],
  "type": "string"
},
"connection_type": {
  "enum": [
    "wi-fi",
    "5g-nr",
    "multi-fire",
    "lte"
  ],
  "type": "string"
},
"ip_address": {
```

```
        "type": "string"
    },
    "port": {
        "maximum": 65535,
        "minimum": 1,
        "type": "integer"
    },
    "adaptation_method": {
        "allof" : [
            { "$ref": "#/definitions/adapt_method" },
            { "$ref": "#/definitions/supported" }
        ]
    },
    "connection": {
        "allof" : [
            { "$ref": "#/definitions/connection_id" },
            { "$ref": "#/definitions/connection_type" }
        ]
    },
    "convergence_method": {
        "allof": [
            { "$ref": "#/definitions/conv_method" },
            { "$ref": "#/definitions/supported" }
        ]
    },
    "feature_status": {
        "allof": [
            { "$ref": "#/definitions/feature_name" },
            { "$ref": "#/definitions/active" }
        ]
    },
    "ncm_end_point": {
        "allof" : [
            { "$ref" : "#/definitions/ip_address" },
            { "$ref" : "#/definitions/port" }
        ]
    },
    "capability_acknowledgement" : {
        "enum" : [
            "MX_ACCEPT",
            "MX_REJECT"
        ],
        "type" : "string"
    },
    "threshold" : {
        "high" : {
            "type" : "integer"
        }
    },
    },
```

```
        "low" : {
            "type" : "integer"
        },
        "type" : "object"
    },
    "meas_report_param" : {
        "enum" : [
            "WLAN_RSSI",
            "WLAN_LOAD",
            "LTE_RSRP",
            "LTE_RSRQ",
            "UL_TPUT",
            "DL_TPUT",
            "EST_UL_TPUT",
            "EST_DL_TPUT",
            "NR_RSRP",
            "NR_RSRQ",
        ],
        "type" : "string"
    },
    "meas_report_conf" : {
        "meas_rep_param" : {
            "$ref" : "#definitions/meas_report_param"
        },
        "meas_threshold" : {
            "$ref" : "#definitions/threshold"
        },
        "meas_period_ms" : {
            "type" : "integer"
        },
        "type" : "object"
    },
    "ssid_types" : {
        "enum" : [
            "ssid",
            "bssid",
            "hessid"
        ],
        "type" : "string"
    },
    "ip_addr_mask" : {
        "type" : "string",
        "default" : "0.0.0.0/0"
    },
    "port_range" : {
        "start" : {
            "type" : "integer",
            "default" : 0
        }
    }
}
```

```

    },
    "end" : {
        "type" : "integer",
        "default" : 65535
    }
},
"traffic_flow_template" : {
    "remote_addr_mask" : { "$ref" : "#definitions/ip_addr_ma
sk" },
    "local_addr_mask" : { "$ref" : "#definitions/ip_addr_ma
sk" },
    "protocol_type" : {
        "type" : "integer",
        "minimum" : 0,
        "maximum" : 255
    },
    "local_port_range" : { "$ref" : "#definitions/port_rang
e" },
    "remote_port_range" : { "$ref" : "#definitions/port_rang
e" },
    "traffic_class" : {
        "type" : "integer",
        "default" : 255
    },
    "flow_label" : {
        "type" : "integer",
        "default" : 0
    }
},
"delivery_node_id" : {
    "type" : "string"
},
"unique_session_id" : {
    "type" : "object",
    "ncm_id" : {
        "type" : "integer"
    },
    "session_id" : {
        "type" : "integer"
    }
},
"keep_alive_reason" : {
    "enum" : [
        "Timeout",
        "Handover"
    ],
    "type" : "string"
},
"connection_status" : {
    "enum" : [
        "disabled",
        "enabled",

```

```

        "connected"
      ],
      "type" : "string",
      "default" : "connected"
    },
    "adaptation_param" : {
      "udp_adapt_port" : {
        "type" : "integer"
      }
    },
    "probe_param" : {
      "probe_port" : {
        "type" : "integer"
      },
      "anchor_conn_id" : {
        "type" : "integer"
      },
      "mx_configuration_id" : {
        "type" : "integer"
      },
    },
  },
  "client_param" : {
    "connection_id" : {
      "type" : "integer"
    },
    "adapt_param" : {
      "type" : { "$ref" : "#definitions/adaptation_param" }
    }
  },
  },
  "adapt_param": {
    "tunnel_ip_addr": {
      "type": "string"
    },
    "tunnel_end_port": {
      "type": "integer"
    },
    "shared_secret": {
      "type": "string"
    },
    "mx_header_optimization": {
      "type": "boolean",
      "default": false
    }
  },
  "delivery_connection": {
    "connection_id": {
      "$ref": "#definitions/connection_id"
    }
  }
}

```

```

    },
    "connection_type": {
        "$ref": "#definitions/connection_type"
    },
    "adaptation_method": {
        "$ref": "#definitions/adapt_method"
    },
    "adaptation_method_param": {
        "$ref": "#definitions/adapt_param"
    }
},
"app_madp_assoc": {
    "anchor_conn_id" : {
        "type" : "integer"
    },
    "mx_configuration_id" : {
        "type" : "integer"
    }
    "ul_tft_list": {
        "items": {
            "$ref": "#definitions/traffic_flow_templ
ate"
        },
        "type": "array"
    },
    "dl_tft_list": {
        "items": {
            "$ref": "#definitions/traffic_flow_templ
ate"
        },
        "type": "array"
    }
}
"predict_param_name": {
    "enum": [
        "validity time",
        "bandwidth",
        "jitter",
        "latency",
        "signal_quality"
    ],
    "type": "string"
},
"predict_add_param_name": {
    "enum": [
        "WLAN_RSSI",
        "WLAN_LOAD",
        "LTE_RSRP",
        "LTE_RSRQ",

```

```
        "NR_RSRP",
        "NR_RSRQ"
    ],
    "type": "string"
},
{
    "id": "http://www.ietf.org/mams/definitions.json"
}
```

C.3.3. MX Discover

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "id": "http://www.ietf.org/mams/mx_discover.json",
    "properties": {
        "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
        "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
        "version" : {"$ref": "mx_base_def.json#/version_def"}
    },
    "type": "object"
}
```

C.3.4. MX System Update


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_system_info.json",
  "properties": {
    "message_type": {
      "$ref": "mx_base_def.json#/message_type_def"
    },
    "sequence_num": {
      "$ref": "mx_base_def.json#/sequence_num_def"
    },
    "version": {
      "$ref": "mx_base_def.json#/version_def"
    },
    "ncm_connections": {
      "type": "array",
      "items": [
        { "$ref": "definitions.json#/connection" },
        { "$ref": "definitions.json#/ncm_end_point" }
      ]
    }
  },
  "type": "object"
}
```

C.3.5. MX Capability Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_capability_req.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def",
      "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def",
        "version" : { "$ref": "mx_base_def.json#/version_def",
          "adaptation_methods": {
            "items": { "$ref" : "definitions.json#/adaptation_method",
              "type": "array"
            },
            "type": "array"
          },
          "anchor_connections": {
            "items": { "$ref" : "definitions.json#/connection",
              "type": "array"
            },
            "type": "array"
          },
          "convergence_methods": {
            "items": { "$ref" : "definitions.json#/convergence_method",
              "type": "array"
            },
            "type": "array"
          },
          "delivery_connections": {
            "items": { "$ref" : "definitions.json#/connection",
              "type": "array"
            },
            "type": "array"
          },
          "feature_active": {
            "items": { "$ref" : "definitions.json#/feature_status",
              "type": "array"
            },
            "type": "array"
          },
          "num_anchor_connections": {
            "type": "integer"
          },
          "num_delivery_connections": {
            "type": "integer"
          }
        },
        "type": "object"
      }
    },
    "type": "object"
  }
}
```

C.3.6. MX Capability Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_capability_resp.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def" },
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def" },
    "version" : { "$ref": "mx_base_def.json#/version_def" },
    "adaptation_methods": {
      "items": { "$ref": "definitions.json#/adaptation_method" },
      "type": "array"
    },
    "anchor_connections": {
      "items": { "$ref": "definitions.json#/connection" },
      "type": "array"
    },
    "convergence_methods": {
      "items": { "$ref": "definitions.json#/convergence_method" },
      "type": "array"
    },
    "delivery_connections": {
      "items": { "$ref": "definitions.json#/connection" },
      "type": "array"
    },
    "feature_active": {
      "items": { "$ref": "definitions.json#/feature_status" },
      "type": "array"
    },
    "num_anchor_connections": {
      "type": "integer"
    },
    "num_delivery_connections": {
      "type": "integer"
    },
    "unique_session_id" : {
      "$ref": "definitions.json#/unique_session_id"
    }
  },
  "type": "object"
}
```

C.3.7. MX Capability Ack

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_capability_ack.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" },
    "capability_ack": {"$ref" : "definitions.json#/capability_acknowledgement"}
  },
  "type": "object"
}
```

C.3.8. MX Reconfiguration Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_reconf_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : {
      "$ref": "definitions.json#/unique_session_id"
    },
    "connection_id" : {"$ref": "definitions.json#/connection_id" },
    "ip_address": {"$ref": "definitions.json#/ip_address" },
    "mtu_size": {
      "maximum": 65535,
      "minimum": 1,
      "type": "integer"
    },
    "ssid" : {
      "type": "string"
    },
    "reconf_action": {
      "enum": [
        "release",
        "setup",
        "update"
      ],
      "id": "/properties/reconf_action",
      "type": "string"
    },
    "connection_status" : {"$ref": "definitions.json#/connection_status"},
    "delivery_node_id" : {"$ref": "definitions.json#/delivery_node_id"}
  },
  "type": "object"
}
```

C.3.9. MX Reconfiguration Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_reconf_rsp.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"}
  },
  "type": "object"
}
```

C.3.10. MX UP Setup Configuration

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "definitions": {
    "convergence_configuration" : {
      "mx_configuration_id": { "type" : "integer"},
      "convergence_method": { "$ref" : "definitions.json#/conv_method" }
    },
    "convergence_method_params": {
      "properties": {
        "proxy_ip": { "$ref" : "definitions.json#/ip_address" },
        "proxy_port": { "$ref" : "definitions.json#/port" },
        "client_key": { "$ref" : "definitions.json#/client_key" }
      },
      "type": "object"
    },
    "num_delivery_connections": {
      "type": "integer"
    },
    "delivery_connections": {
      "items": { "$ref" : "definitions.json#/delivery_connection" },
      "type": "array"
    }
  },
  "id": "http://www.ietf.org/mams/mx_up_setup_conf_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "num_anchor_connections": {
      "type": "integer"
    },
    "anchor_connections": {
```

```

    "items": {
      "properties": {
        "connection_id": { "$ref" : "definitions.json#/connection_id" },
        "connection_type": { "$ref" : "definitions.json#/connection_type" },
        "num_active_mx_conf" : { "type" : "integer" },
        "convergence_config" : {
          "items": { "$ref" : "definitions/convergence_configuratio
n" },
          "type" : "array"
        }
      },
      "type": "object"
    },
    "type": "array"
  }
},
"type": "object"
}

```

C.3.11. MX UP Setup Confirmation

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_up_setup_cnf.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : { "$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" },
    "probe_param" : { "$ref": "definitions.json#/probe_param" },
    "num_delivery_conn" : {
      "type" : "integer"
    },
    "client_params" : {
      "type" : "array",
      "items" : [
        { "$ref": "definitions.json#/client_param" }
      ]
    }
  },
  "type": "object"
}

```

C.3.12. MX Traffic Steering Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {
    "conn_list" : {
      "items" : { "$ref" : "definitions.json#/connection_id" },
      "type": "array"
    },
    "ul_delivery" : {
      "ul_tft" : { "$ref" : "definitions.json#/traffic_flow_template" }
    },
    "connection_list" : { "$ref" : "#definitions/conn_list" }
  },
  "id": "http://www.ietf.org/mams/mx_traffic_steering_req.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def" },
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def" },
    "version" : { "$ref": "mx_base_def.json#/version_def" },
    "connection_id": { "$ref" : "definitions.json#/connection_id" },
    "mx_configuration_id": { "type" : "integer" },
    "downlink_delivery": {
      "items": { "$ref" : "definitions.json#/connection_id" },
      "type": "array"
    },
    "feature_activation": {
      "items": { "$ref" : "definitions.json#/feature_status" },
      "type": "array"
    },
    "default_uplink_delivery": {
      "type": "integer"
    },
    "uplink_delivery": {
      "items": { "$ref" : "#definitions/ul_delivery" },
      "type": "array"
    }
  },
  "type": "object"
}
```

C.3.13. MX Traffic Steering Response


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://example.com/example.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" },
    "feature_activation": {
      "items": {"$ref" : "definitions.json#/feature_status" },
      "type": "array"
    }
  },
  "type": "object"
}
```

C.3.14. MX Application MADP Association Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://example.com/example.json",
  "properties": {
    "message_type": {
      "$ref": "mx_base_def.json#/message_type_def"
    },
    "sequence_num": {
      "$ref": "mx_base_def.json#/sequence_num_def"
    },
    "version": {
      "$ref": "mx_base_def.json#/version_def"
    },
    "unique_session_id": {
      "$ref": "definitions.json#/unique_session_id"
    },
    "app_madp_assoc_list": {
      "items": {
        "$ref": "definitions.json#/app_madp_assoc"
      },
      "type": "array"
    }
  },
  "type": "object"
}
```

C.3.15. MX Application MADP Association Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://example.com/example.json",
  "properties": {
    "message_type": {
      "$ref": "mx_base_def.json#/message_type_def"
    },
    "sequence_num": {
      "$ref": "mx_base_def.json#/sequence_num_def"
    },
    "version": {
      "$ref": "mx_base_def.json#/version_def"
    },
    "unique_session_id": {
      "$ref": "definitions.json#/unique_session_id"
    },
    "is_success": {
      "type": "boolean"
    }
  },
  "type": "object"
}
```

C.3.16. MX Path Estimation Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_path_est_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "active_probe_ack_req": {
      "enum": [
        "no",
        "yes"
      ],
      "type": "string"
    },
    "active_probe_freq_ms": {
      "maximum": 10000,
      "minimum": 100,
      "type": "integer"
    }
  }
}
```

```
    },
    "active_probe_size_bytes": {
        "maximum": 1500,
        "minimum": 100,
        "type": "integer"
    },
    "active_probe_duration_sec" : {
        "maximum" : 100,
        "minimum" : 10,
        "type" : "integer"
    },
    "connection_id": { "$ref" : "definitions#/connection_id" },
    "init_probe_ack_req": {
        "enum": [
            "no",
            "yes"
        ],
        "type": "string"
    },
    "init_probe_size_bytes": {
        "maximum": 1500,
        "minimum": 100,
        "type": "integer"
    },
    "init_probe_test_duration_ms": {
        "maximum": 10000,
        "minimum": 100,
        "type": "integer"
    },
    "init_probe_test_rate_Mbps": {
        "maximum": 100,
        "minimum": 1,
        "type": "integer"
    }
},
"type": "object"
}
```

C.3.17. MX Path Estimation Report

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_path_est_results.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def" },
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def" },
    "version" : { "$ref": "mx_base_def.json#/version_def" },
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" },
    "active_probe_results": {
      "properties": {
        "avg_tput_last_probe_duration_Mbps": {
          "maximum": 100,
          "minimum": 1,
          "type": "number"
        }
      },
      "type": "object"
    },
    "connection_id": { "$ref": "definitions.json#/connection_id" },
    "init_probe_results": {
      "properties": {
        "lost_probes_percentage": {
          "maximum": 100,
          "minimum": 1,
          "type": "integer"
        },
        "probe_rate_Mbps": {
          "maximum": 100,
          "minimum": 1,
          "type": "number"
        }
      },
      "type": "object"
    }
  },
  "type": "object"
}
```

C.3.18. MX SSID Indication

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_ssid_indication.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "ssid_list": {
      "items": {
        "properties": {
          "ssid_type": { "$ref" : "definitions.json#/ssid_types" },
          "ssid_id" : {
            "type" : "integer"
          }
        }
      },
      "type": "array"
    },
    "type": "object"
  }
}

```

C.3.19. MX Measurements Configuration

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "definitions" : {
    "meas_conf" : {
      "connection_id" : { "$ref" : "definitions.json#/connection_id" }
    },
    "connection_type" : { "$ref" : "definitions.json#/connection_type" },
    "meas_rep_conf" : {
      "items" : { "$ref" : "definitions.json#/meas_report_conf" },
      "type" : "array"
    }
  },
  "id": "http://www.ietf.org/mams/mx_measurement_conf.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def" },
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def" },
    "version" : { "$ref": "mx_base_def.json#/version_def" },
    "measurement_configuration" : {
      "items" : { "$ref" : "#definitions/meas_conf" },
      "type" : "array"
    }
  },
  "type": "object"
}
```

C.3.20. MX Measurements Report

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "http://www.ietf.org/mams/mx_measurement_report.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : {"$ref": "definitions.json#/unique_sessio
n_id" },
    "measurment_reports": {
      "items": {
        "properties": {
          "connection_id": {
            "$ref" : "definitions.json#/connection_id"
          },
          "connection_type" : {
            "$ref" : "definitions.json#/conn
ection_type"
          },
          "delivery_node_id" : {
            "$ref" : "definitions.json#/deli
very_node_id"
          },
          "measurements": {
            "items": {
              "properties": {
                "measurement_type": {
                  "$ref" : "definitions.json#/meas_report_param"
                },
                "measurement_value": {
                  "type": "integer"
                }
              },
              "type": "object"
            },
            "type": "array"
          }
        },
        "type": "object"
      },
      "type": "array"
    }
  },
  "type": "object"
}

```

C.3.21. MX Keep Alive Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_keep_alive_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "keep_alive_reason" : {"$ref": "definitions.json#/keep_alive_reason"},
    "unique_session_id" : {"$ref": "definitions.json#/unique_session_id"},
    "connection_id" : {"$ref": "definitions.json#/connection_id"},
    "delivery_node_id" : {"$ref": "definitions.json#/connection_id"}
  },
  "type": "object"
}
```

C.3.22. MX Keep Alive Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_keep_alive_rsp.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : {"$ref": "definitions.json#/unique_session_id"}
  },
  "type": "object"
}
```

C.3.23. MX Session Termination Request


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_keep_alive_req.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def" },
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def" },
    "version" : { "$ref": "mx_base_def.json#/version_def" },
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" },
    "reason" : {
      "enum" : [
        "MX_NORMAL_RELEASE",
        "MX_NO_RESPONSE",
        "INTERNAL_ERROR"
      ],
      "type" : "string"
    }
  },
  "type": "object"
}
```

C.3.24. MX Session Termination Response

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_session_termination_resp.json",
  "properties": {
    "message_type" : { "$ref": "mx_base_def.json#/message_type_def" },
    "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def" },
    "version" : { "$ref": "mx_base_def.json#/version_def" },
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" }
  },
  "type": "object"
}
```

C.3.25. MX Network Analytics Request

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "id": "http://www.ietf.org/mams/mx_network_analytics_req.json",
  "properties": {
    "message_type" : {"$ref": "mx_base_def.json#/message_type_def"},
    "sequence_num" : {"$ref": "mx_base_def.json#/sequence_num_def"},
    "version" : {"$ref": "mx_base_def.json#/version_def"},
    "unique_session_id" : { "$ref": "definitions.json#/unique_session_id" },
    "params" : {
      "items" : {"$ref": "definitions.json#/predict_param_name"},
      "type" : "array"
    }
  },
  "type": "object"
}
```

C.3.26. MX Network Analytics Response

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "additionalProperties": false,
  "definitions" : {
    "ParamPredictions" : {
      "param_name" : { "$ref" : "definitions.json#/predict_param_name"},
      "additional_param" : { "$ref" : "definitions.json#/predict_add_param_name"
    },
    "prediction" : { "type" : "integer" },
    "likelihood" : { "type" : "integer" },
    "validity_time" : { "type" : "integer" }
  },
  "MXAnalyticsList" : {
    "connection_id" : { "$ref" : "definitions.json#/connection_id" }
  },
  "connection_type" : { "$ref" : "definitions.json#/connection_type" },
  "predictions" : {
    "items" : { "$ref" : "#definitions/ParamPredictions" },
    "type" : "array"
  }
},
"id": "http://www.ietf.org/mams/mx_network_analytics_resp.json",
"properties": {
  "message_type" : { "$ref": "mx_base_def.json#/message_type_def"},
  "sequence_num" : { "$ref": "mx_base_def.json#/sequence_num_def"},
  "version" : { "$ref": "mx_base_def.json#/version_def"},
  "param_list" : {
    "items" : { "$ref": "#definitions/MXAnalyticsList"
  },
  "type" : "array"
},
"type": "object"
}

```

C.4. Examples in JSON

C.4.1. MX Discover

```

{
  "version" : "1.0",
  "message_type" : "mx_discover",
  "sequence_num" : 1
}

```

C.4.2. MX System Update

```
{
  "version" : "1.0",
  "message_type" : "mx_system_info",
  "sequence_num" : 2,
  "ncm_connections" : [
    {
      "connection_id" : 0,
      "connection_type" : "lte",
      "ncm_end_point" : {
        "ip_address" : "192.168.1.10",
        "port" : 1234
      }
    },
    {
      "connection_id" : 1,
      "connection_type" : "wifi",
      "ncm_end_point" : {
        "ip_address" : "192.168.1.10",
        "port" : 1234
      }
    }
  ]
}
```

C.4.3. MX Capability Request

```
{
  "version" : "1.0",
  "message_type" : "mx_capability_req",
  "sequence_num" : 3,
  "feature_active" : [
    {
      "feature_name" : "lossless_switching",
      "active" : true
    },
    {
      "feature_name" : "fragmentation",
      "active" : false
    }
  ],
  "num_anchor_connections" : 2,
  "anchor_connections" : [
    {
      "connection_id" : 0,
      "connection_type" : "lte"
    },
    {
      "connection_id" : 1,

```

```
        "connection_type" : "wifi"
    },
    ],
    "num_delivery_connections" : 2,
    "delivery_connections" : [
        {
            "connection_id" : 0,
            "connection_type" : "lte"
        },
        {
            "connection_id" : 1,
            "connection_type" : "wifi"
        }
    ],
    "convergence_methods" : [
        {
            "method" : "GMA",
            "supported" : true
        },
        {
            "method" : "MPTCP_Proxy",
            "supported" : false
        }
    ],
    "adaptation_methods" : [
        {
            "method" : "UDP_without_DTLS",
            "supported" : false
        },
        {
            "method" : "UDP_with_TLS",
            "supported" : false
        },
        {
            "method" : "IPSec",
            "supported" : true
        },
        {
            "method" : "Client_NAT",
            "supported" : false
        }
    ]
}
```

C.4.4. MX Capability Response

```
{
  "version" : "1.0",
  "message_type" : "mx_capability_resp",
  "sequence_num" : 3,
  "feature_active" : [
    {
      "feature_name" : "lossless_switching",
      "active" : true
    },
    {
      "feature_name" : "fragmentation",
      "active" : false
    }
  ],
  "num_anchor_connections" : 2,
  "anchor_connections" : [
    {
      "connection_id" : 0,
      "connection_type" : "lte"
    },
    {
      "connection_id" : 1,
      "connection_type" : "wifi"
    }
  ],
  "num_delivery_connections" : 2,
  "delivery_connections" : [
    {
      "connection_id" : 0,
      "connection_type" : "lte"
    },
    {
      "connection_id" : 1,
      "connection_type" : "wifi"
    }
  ],
  "convergence_methods" : [
    {
      "method" : "GMA",
      "supported" : true
    },
    {
      "method" : "MPTCP_Proxy",
      "supported" : false
    }
  ],
}
```

```
"adaptation_methods" : [  
  {  
    "method" : "UDP_without_DTLS",  
    "supported" : false  
  },  
  {  
    "method" : "UDP_with_TLS",  
    "supported" : false  
  },  
  {  
    "method" : "IPSec",  
    "supported" : true  
  },  
  {  
    "method" : "Client_NAT",  
    "supported" : false  
  }  
],  
"unique_session_id" : {  
  "ncm_id" : 110,  
  "session_id" : 1111  
}  
}
```

C.4.5. MX Capability Ack

```
{  
  "version" : "1.0",  
  "message_type" : "mx_capability_ack",  
  "sequence_num" : 3,  
  "unique_session_id" : {  
    "ncm_id" : 110,  
    "session_id" : 1111  
  },  
  "capability_ack" : "MX ACCEPT"  
}
```

C.4.6. MX Reconfiguration Request

```

{
  "version" : "1.0",
  "message_type" : "mx_reconf_req",
  "sequence_num" : 4,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "reconf_action" : "setup",
  "connection_id" : 0,
  "ip_address" : "192.168.110.1",
  "ssid" : "SSID_1",
  "mtu_size" : 1300,
  "connection_status" : "connected",
  "delivery_node_id" : "2A12C"
}

```

C.4.7. MX Reconfiguration Response

```

{
  "version" : "1.0",
  "message_type" : "mx_reconf_rsp",
  "sequence_num" : 4
}

```

C.4.8. MX UP Setup Configuration Request

```

{
  "version": "1.0",
  "message_type": "mx_up_setup_conf_req",
  "sequence_num": 5,
  "num_anchor_connections": 2,
  "anchor_connections": [{
    "connection_type": "wifi",
    "num_active_mx_conf" : 2,
    "convergence_config" : [
      {
        "mx_configuration_id" : 1,
        "convergence_method": "GMA",
        "convergence_method_params": {},
        "num_delivery_connections": 2,
        "delivery_connections": [{
          "connection_id": 0,
          "connection_type": "lte",
          "adaptation_method": "UDP_withou
t_DTLS",
          "adaptation_method_param": {
            "tunnel_ip_addr": "6.6.6
.6",
            "tunnel_end_port": 9999,

```



```

                                "mx_header_optimization"
: true
                                }
                                },
                                {
                                    "connection_id": 1,
                                    "connection_type": "wifi"
                                }
                            ]
                        },
                        {
                            "mx_configuration_id" : 2,
                            "convergence_method": "GMA",
                            "convergence_method_params": {},
                            "num_delivery_connections": 1,
                            "delivery_connections": [{
                                "connection_id": 0,
                                "connection_type": "lte",
                                "adaptation_method": "UDP_withou
t_DTLS",
                                "adaptation_method_param": {
                                    "tunnel_ip_addr": "6.6.6
.6",
                                    "tunnel_end_port": 8877
                                }
                            }
                        ]
                    }
                ],
                {
                    {
                        "connection_id": 0,
                        "connection_type": "lte",
                        "udp_port": 8888,
                        "num_delivery_connections": 2,
                        "delivery_connections": [{
                            "connection_id": 0,
                            "connection_type": "lte"
                        },
                        {
                            "connection_id": 1,
                            "connection_type": "wifi",
                            "adaptation_method": "UDP_without_DTLS",
                            "adaptation_method_param": {
                                "tunnel_ip_addr": "192.168.3.3",
                                "tunnel_end_port": "6000"
                            }
                        }
                    ]
                }
            ]
        ]
    }

```

```
}
```

C.4.9. MX UP Setup Confirmation

```
{
  "version" : "1.0",
  "message_type" : "mx_up_setup_cnf",
  "sequence_num" : 5,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "probe_param" : {
    "probe_port" : 48700,
    "anchor_conn_id" : 0,
    "mx_configuration_id" : 1
  },
  "num_delivery_conn" : 2,
  "client_params" : [
    {
      "connection_id" : 0,
      "adapt_param" : {
        "udp_adapt_port" : 51000
      }
    },
    {
      "connection_id" : 1,
      "adapt_param" : {
        "udp_adapt_port" : 52000
      }
    }
  ]
}
```

C.4.10. MX Traffic Steering Request

```
{
  "version" : "1.0",
  "message_type" : "mx_traffic_steering_req",
  "sequence_num" : 6,
  "connection_id" : 0,
  "mx_configuration_id" : 1,
  "downlink_delivery" : [
    {
      "connection_id" : 0
    },
    {
      "connection_id" : 1
    }
  ]
}
```

```
    }
  ],
  "default_uplink_delivery" : 0,
  "uplink_delivery" : [
    {
      "ul_tft" : {
        "remote_addr_mask" : "10.10.0.0/24",
        "local_addr_mask" : "192.168.0.0/24",
        "protocol_type" : 6,
        "local_port_range" : {
          "start" : 100,
          "end" : 1000
        },
        "remote_port_range" : {
          "start" : 100,
          "end" : 1000
        },
        "traffic_class" : 20,
        "flow_label" : 100
      },
      "conn_list" : [
        {
          "connection_id" : 1
        }
      ]
    },
    {
      "ul_tft" : {
        "remote_addr_mask" : "10.10.0.0/24",
        "local_addr_mask" : "192.168.0.0/24",
        "protocol_type" : 6,
        "local_port_range" : {
          "start" : 2000,
          "end" : 2000
        },
        "remote_port_range" : {
          "start" : 100,
          "end" : 1000
        },
        "traffic_class" : 20,
        "flow_label" : 50
      },
      "conn_list" : [
        {
          "connection_id" : 1
        }
      ]
    }
  ]
}
```

```
    ],
    "feature_activation" : [
        {
            "feature_name" : "dl_aggregation",
            "active" : true
        },
        {
            "feature_name" : "ul_aggregation",
            "active" : false
        }
    ]
}
```

C.4.11. MX Traffic Steering Response

```
{
    "version": "1.0",
    "message_type": "mx_traffic_steering_rsp",
    "sequence_num": 6,
    "unique_session_id": {
        "ncm_id": 110,
        "session_id": 1111
    },
    "feature_activation": [{
        "feature_name": "lossless_switching",
        "active": true
    },
    {
        "feature_name": "fragmentation",
        "active": false
    }
    ]
}
```

C.4.12. MX Application MADP Association Request

```
{
  "version": "1.0",
  "message_type": "mx_app_madp_assoc_req",
  "sequence_num": 6,
  "unique_session_id": {
    "ncm_id": 110,
    "session_id": 1111
  },
  "app_madp_assoc_list": [{
    "connection_id" : 0,
    "mx_configuration_id" : 1,
    "ul_tft_list": [{
      "protocol_type": 17,
      "local_port_range": {
        "start": 8888,
        "end": 8888
      }
    }],
    "dl_tft_list": [{
      "protocol_type": 17,
      "remote_port_range": {
        "start": 8888,
        "end": 8888
      }
    }
  ]
}
```

C.4.13. MX Application MADP Association Response

```
{
  "version": "1.0",
  "message_type": "mx_app_madp_assoc_resp",
  "sequence_num": 6,
  "is_success": true
}
```

C.4.14. MX Path Estimation Request

```
{
  "version" : "1.0",
  "message_type" : "mx_path_est_req",
  "sequence_num" : 7,
  "connection_id" : 0,
  "init_probe_test_duration_ms" : 100,
  "init_probe_test_rate_Mbps" : 10,
  "init_probe_size_bytes" : 1000,
  "init_probe_ack_req" : "yes",
  "active_probe_freq_ms" : 10000,
  "active_probe_size_bytes" : 1000,
  "active_probe_duration_sec" : 10,
  "active_probe_ack_req" : "no"
}
```

C.4.15. MX Path Estimation Results

```
{
  "version" : "1.0",
  "message_type" : "mx_path_est_results",
  "sequence_num" : 8,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "connection_id" : 0,
  "init_probe_results" : {
    "lost_probes_percentage" : 1,
    "probe_rate_Mbps" : 9.9
  },
  "active_probe_results" : {
    "avg_tput_last_probe_duration_Mbps" : 9.8
  }
}
```

C.4.16. MX SSID Indication

```
{
  "version" : "1.0",
  "message_type" : "mx_ssid_indication",
  "sequence_num" : 9,
  "ssid_list" : [
    {
      "ssid_type" : "ssid",
      "ssid_id" : "SSID_1"
    },
    {
      "ssid_type" : "bssid",
      "ssid_id" : "xxx-yyy"
    }
  ]
}
```

C.4.17. MX Measurements Configuration

```
{
  "version" : "1.0",
  "message_type" : "mx_measurement_conf",
  "sequence_num" : 10,
  "measurement_configuration" : [
    {
      "connection_id" : 0,
      "connection_type" : "wi-fi",
      "meas_rep_conf" : [
        {
          "meas_rep_param" : "WLAN_RSSI",
          "meas_threshold" : {
            "high" : -10,
            "low" : -15
          },
          "meas_period_ms" : 500
        },
        {
          "meas_rep_param" : "WLAN_LOAD",
          "meas_threshold" : {
            "high" : -10,
            "low" : -15
          },
          "meas_period_ms" : 500
        },
        {
          "meas_rep_param" : "EST_UL_TPUT",
          "meas_threshold" : {
            "high" : 100,
            "low" : 30
          }
        }
      ]
    }
  ]
}
```

```
        },
        "meas_period_ms" : 500
    }
]
},
{
    "connection_id" : 1,
    "connection_type" : "lte",
    "meas_rep_conf" : [
        {
            "meas_rep_param" : "LTE_RSRP",
            "meas_threshold" : {
                "high" : -10,
                "low" : -15
            },
            "meas_period_ms" : 500
        },
        {
            "meas_rep_param" : "LTE_RSRQ",
            "meas_threshold" : {
                "high" : -10,
                "low" : -15
            },
            "meas_period_ms" : 500
        }
    ]
}
]
```

C.4.18. MX Measurements Report


```
{
  "version" : "1.0",
  "message_type" : "mx_measurement_report",
  "sequence_num" : 11,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "measurment_reports" : [
    {
      "connection_id" : 0,
      "connection_type" : "wi-fi",
      "delivery_node_id" : "2021A",
      "measurements" : [
        {
          "measurement_type" : "WLAN_RSSI",
          "measurement_value" : -12
        },
        {
          "measurement_type" : "UL_TPUT",
          "measurement_value" : 10
        },
        {
          "measurement_type" : "EST_UL_TPUT",
          "measurement_value" : 20
        }
      ]
    },
    {
      "connection_id" : 1,
      "connection_type" : "lte",
      "delivery_node_id" : "12323",
      "measurements" : [
        {
          "measurement_type" : "LTE_RSRP",
          "measurement_value" : -12
        },
        {
          "measurement_type" : "LTE_RSRQ",
          "measurement_value" : -12
        }
      ]
    }
  ]
}
```

C.4.19. MX Keep Alive Request

```
{
  "version" : "1.0",
  "message_type" : "mx_keep_alive_req",
  "sequence_num" : 12,
  "keep_alive_reason" : "Handover",
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "connection_id" : 0,
  "delivery_node_id" : "2021A"
}
```

C.4.20. MX Keep Alive Response

```
{
  "version" : "1.0",
  "message_type" : "mx_keep_alive_rsp",
  "sequence_num" : 12,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  }
}
```

C.4.21. MX Session Termination Request

```
{
  "version" : "1.0",
  "message_type" : "mx_session_termination_req",
  "sequence_num" : 13,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "reason" : "MX_NORMAL_RELEASE"
}
```

C.4.22. MX Session Termination Response

```
{
  "version" : "1.0",
  "message_type" : "mx_session_termination_resp",
  "sequence_num" : 13,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  }
}
```

C.4.23. MX Network Analytics Request

```
{
  "version" : "1.0",
  "message_type" : "mx_network_analytics_req",
  "sequence_num" : 20,
  "unique_session_id" : {
    "ncm_id" : 110,
    "session_id" : 1111
  },
  "parmas" : [
    "jitter",
    "latency"
  ]
}
```

C.4.24. MX Network Analytics Response

```

{
  "version": "1.0",
  "message_type": "mx_network_analytics_resp",
  "sequence_num": 20,
  "param_list": [{
    "connection_id": 1,
    "connection_type": "wifi",
    "predictions": [{
      "param_name": "jitter",
      "prediction": 100,
      "likelihood": 50,
      "validity_time": 10
    },
    {
      "param_name": "latency",
      "prediction": 19,
      "likelihood": 40,
      "validity_time": 10
    }
  ]
},
{
  "connection_id": 2,
  "connection_type": "lte",
  "predictions": [{
    "param_name": "jitter",
    "prediction": 10,
    "likelihood": 80,
    "validity_time": 10
  },
  {
    "param_name": "latency",
    "prediction": 4,
    "likelihood": 60,
    "validity_time": 10
  }
]
}
]
}

```

Appendix D. Definition of APIs provided by CCM to the Applications at the Client

This section provides an example implementation of the APIs exposed by the CCM to the Applications on the client, documented with OpenAPI using Swagger 2.0.

```

{
  "swagger": "2.0",
  "info": {
    "version": "1.0.0",
    "title": "Client Connection Manager (CCM)",
    "description": "API provided by CCM towards Application on a MAMS client."
  },
  "host": "MAMS.ietf.org",
  "basePath": "/ccm/v1.0",
  "schemes": [
    "https"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/capabilities": {
      "get": {
        "description": "This API can be used by application to request for capabilities of the CCM.",
        "produces": [
          "application/json",
          "text/html"
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/capability"
            }
          },
          "default": {
            "description": "unexpected error",
            "schema": {
              "$ref": "#/definitions/errorModel"
            }
          }
        }
      }
    },
    "/app_requirements": {
      "post": {
        "description": "This API is used by N-MADP to report any kind of MAMS user specific errors to NCM.",
        "produces": [
          "application/json",
          "text/html"
        ]
      }
    }
  }
}

```

```

    ],
    "parameters": [
      {
        "name": "app-requirements",
        "in": "body",
        "required": true,
        "schema": {
          "$ref": "#/definitions/app-requirements"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK"
      },
      "default": {
        "description": "unexpected error",
        "schema": {
          "$ref": "#/definitions/errorModel"
        }
      }
    }
  },
  "/predictive_link_params": {
    "get": {
      "description": "This API is used by applications to get the information
about predicted parameters for each delivery connection.",
      "produces": [
        "application/json",
        "text/html"
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/link-params"
          }
        },
        "default": {
          "description": "unexpected error",
          "schema": {
            "$ref": "#/definitions/errorModel"
          }
        }
      }
    }
  }
},

```

```
"definitions": {
  "connection-id": {
    "type": "integer",
    "format": "uint8"
  },
  "connection-type": {
    "enum": [
      "wi-fi",
      "5g-nr",
      "multi-fire",
      "lte"
    ],
    "type": "string"
  },
  "features": {
    "enum": [
      "lossless_switching",
      "fragmentation",
      "concatenation",
      "uplink_aggregation",
      "downlink_aggregation",
      "measurement"
    ],
    "type": "string"
  },
  "adaptation-methods": {
    "enum": [
      "UDP_without_DTLS",
      "UDP_with_DTLS",
      "IPSec",
      "Client_NAT"
    ],
    "type": "string"
  },
  "convergence-methods": {
    "enum": [
      "GMA",
      "MPTCP_Proxy",
      "GRE_Aggregation_Proxy",
      "MPQUIC"
    ],
    "type": "string"
  },
  "connection": {
    "type": "object",
    "properties": {
      "conn-id": {
        "$ref": "#/definitions/connection-id"
      }
    }
  }
}
```

```
    },
    "conn-type": {
      "$ref": "#/definitions/connection-type"
    }
  },
  "convergence-parameters": {
    "type": "object",
    "properties": {
      "conv-param-name": {
        "type": "string"
      },
      "conv-param-value": {
        "type": "string"
      }
    }
  },
  "convergence-details": {
    "type": "object",
    "properties": {
      "conv-method": {
        "$ref": "#/definitions/convergence-methods"
      },
      "conv-params": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/convergence-parameters"
        }
      }
    }
  },
  "capability": {
    "type": "object",
    "properties": {
      "connections": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/connection"
        }
      },
      "features": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/features"
        }
      },
      "adapt-methods": {
        "type": "array",

```



```
        "items": {
          "$ref": "#/definitions/adaptation-methods"
        }
      },
      "conv-methods": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/convergence-details"
        }
      }
    }
  },
  "qos-param-name": {
    "enum": [
      "jitter",
      "latency",
      "bandwidth"
    ],
    "type": "string"
  },
  "qos-param": {
    "type": "object",
    "properties": {
      "qos-param-name": {
        "$ref": "#/definitions/qos-param-name"
      },
      "qos-param-value": {
        "type": "integer"
      }
    }
  },
  "port-range": {
    "type": "object",
    "properties": {
      "start": {
        "type": "integer"
      },
      "end": {
        "type": "integer"
      }
    }
  },
  "protocol-type": {
    "type": "integer"
  },
  "stream-features": {
    "type": "object",
    "properties": {
```

```
    "proto": {
      "$ref": "#/definitions/protocol-type"
    },
    "port-range": {
      "$ref": "#/definitions/port-range"
    },
    "traffic-qos": {
      "$ref": "#/definitions/qos-param"
    }
  }
},
"app-requirements": {
  "type": "object",
  "properties": {
    "num-streams": {
      "type": "integer"
    },
    "stream-feature": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/stream-features"
      }
    }
  }
},
"param-name": {
  "enum": [
    "bandwidth",
    "jitter",
    "latency",
    "signal_quality"
  ],
  "type": "string"
},
"additional-param-name": {
  "enum": [
    "lte-rsrp",
    "lte-rspq",
    "nr-rsrp",
    "nr-rsrq",
    "wifi-rssi"
  ],
  "type": "string"
},
"link-parameter": {
  "type": "object",
  "properties": {
    "connection": {
```

```
        "$ref": "#/definitions/connection"
      },
      "param": {
        "$ref": "#/definitions/param-name"
      },
      "additional-param": {
        "$ref": "#/definitions/additional-param-name"
      },
      "prediction": {
        "type": "integer"
      },
      "likelihood": {
        "type": "integer"
      },
      "validity_time": {
        "type": "integer"
      }
    }
  },
  "link-params": {
    "type": "array",
    "items": {
      "$ref": "#/definitions/link-parameter"
    }
  },
  "errorModel": {
    "type": "object",
    "description": "Error indication containing the error code and message.",
    "required": [
      "code",
      "message"
    ],
    "properties": {
      "code": {
        "type": "integer",
        "format": "int32"
      },
      "message": {
        "type": "string"
      }
    }
  }
}
```

Appendix E. Implementation Example using Python for MAMS Client and Server

E.1. Client Side Implementation

A simple client side implementation using python can be as following:

```
#!/usr/bin/env python
import asyncio
import websockets
import json
import ssl
import time
import sys

context = ssl.SSLContext(ssl.PROTOCOL_TLS)
context.verify_mode = ssl.CERT_REQUIRED
context.set_ciphers("RSA")
context.check_hostname = False
context.load_verify_locations("/home/mecadmin/certs/rootca.pem")

discoverMsg = {'version':'1.0',
'message_type':'mx_discover'}

MXCapabilityRes = { 'version':'1.0',
'message_type':'mx_capability_res',
'FeatureActive': [{'feature_name':'fragmentation', 'active':'yes'}, {'feature_name':'lossless_switching', 'active':'yes'}],
'num_anchor_connections':1,
'anchor_connections': [{'connection_id':0, 'connection_type':'lte'}],
'num_delivery_connections':1,
'delivery_connections': [{'connection_id':1, 'connection_type':"wifi"}],
'convergence_methods': [{'method':'GMA', 'supported':'true'}],
'adaptation_methods': [{'method':'client_nat', 'supported':'false'}]
}

async def hello():
    async with websockets.connect('wss://localhost:8765', ssl=context) as websocket:
        try:
            loopFlag=False
            while True:
                await websocket.send(json.dumps(discoverMsg))
                json_message = await websocket.recv()
                message = json.loads(json_message)
                if "message_type" in message.keys():
                    print("Recieved message:{}".format(message["message_type"]), "version:{}".format(message["version"]))
                    if message["message_type"] == "mx_capability_req" :
                        await websocket.send(json.dumps(MXCapabilityRes))
                        loopFlag=True
                        while(loopFlag==True):
                            pass
        except:
            print("Client stopped")

asyncio.get_event_loop().run_until_complete(hello())
```

E.2. Server Side Implementation

A server client side implementation using python can be as following:

```
#!/usr/bin/env python
import asyncio
import websockets
import json
import ssl

ctx = ssl.SSLContext(ssl.PROTOCOL_TLS)
#ctx.set_ciphers("RSA-AES256-SHA")
ctx.load_verify_locations("/home/mecadmin/certs/rootca.pem")
certfile = "/home/mecadmin/certs/server.pem"
keyfile = "/home/mecadmin/certs/serverkey.pem"
ctx.load_cert_chain(certfile, keyfile, password=None)

MXCapabilityReq = { 'version':'1.0',
'message_type':'mx_capability_req',
'FeatureActive': [{'feature_name':'fragmentation', 'active':'yes'}, {'feature_name':'lossless_switching', 'active':'yes'}],
'num_anchor_connections':1,
'anchor_connections':[{'connection_id':0, 'connection_type':'lte'}],
'num_delivery_connections':1,
'delivery_connections':[{'connection_id':1, 'connection_type':"wifi"}],
'convergence_methods':[{'method':'GMA', 'supported':'true'}],
'adaptation_methods':[{'method':'client_nat', 'supported':'false'}]
}

async def hello(websocket, path):
    try:
        while True:
            name = await websocket.recv()
            msg = json.loads(name)
            if "message_type" in msg.keys():
                print("Recieved message:{}".format(msg["message_type"]), "version:{}".format(msg["version"]))
                if msg['message_type'] == 'mx_discover':
                    await websocket.send(json.dumps(MXCapabilityReq))

    except:
        print("client disconnected")

try:
    start_server = websockets.serve(hello, 'localhost', 8765, ssl=ctx)

    asyncio.get_event_loop().run_until_complete(start_server)
    asyncio.get_event_loop().run_forever()
except:
    print("server stopped")
```

Authors' Addresses

Satish Kanugovi
Nokia

Email: satish.k@nokia.com

Florin Baboescu
Broadcom

Email: florin.baboescu@broadcom.com

Jing Zhu
Intel

Email: jing.z.zhu@intel.com

Julius Mueller
AT&T

Email: jml69k@att.com

SungHoon Seo
Korea Telecom

Email: sh.seo@kt.com

Internet Area WG
Internet-Draft
Intended status: Informational
Expires: December 9, 2018

L. Muscariello
G. Carofiglio
J. Auge
M. Papalini
Cisco Systems Inc.
June 07, 2018

Hybrid Information-Centric Networking
draft-muscariello-intarea-hicn-00

Abstract

This document describes the hybrid information-centric networking (hICN) architecture for IPv6. The specifications describe a way to implement information-networking functionalities into IPv6. The objective is to use IPv6 without creating overlays with a new packet format as an additional encapsulation. The intent of the present design is to introduce some IPv6 routers in the network with additional packet processing operations to implement ICN functions. Moreover, the current design is tightly integrated into IPv6 to allow easy interconnection to IPv6 networks with the additional design objective to exploit existing IPv6 protocols as much as possible as they are, or extend them where needed.

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 December 9, 2018.

Copyright Notice

Copyright (c) 2018 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
2. Architecture	3
2.1. End-points	6
2.2. Naming	7
2.2.1. Name prefix	7
2.2.2. Name Suffix	8
2.3. Packet Format	8
2.3.1. Interest Packet	8
2.3.2. Data Packet	9
2.4. Packet cache	12
2.5. Forwarding	12
2.5.1. Interest Path	12
2.5.2. Data Path	14
3. Security	15
4. IANA Considerations	18
5. Acknowledgements	18
6. References	18
6.1. Normative References	18
6.2. Informative References	19
Authors' Addresses	21

1. Introduction

The objective of this document is to describe hybrid ICN, a network protocol that integrates ICN in IPv6, at a minimum cost in terms of required modifications in end-points and routers and in a way to guarantee transparent interconnection with IP without using overlays.

The ICN reference design used in this document is CCNx as described in [I-D.irtf-icnrg-ccnxsemantics] and [I-D.irtf-icnrg-ccnxmessages]. IPv6 is used as described in [RFC8200].

There are some basic design principles behind the hICN architecture that are implemented by the design reported below that can be summarized as follows:

- o (i) the network can transport many different kinds of applications as IPv6, i.e. hICN can serve content-distribution or real-time applications, to cite examples with very different requirements. hICN is not a content-distribution network;
- o (ii) it provides connection-less and location independent communications by identifying data with unique global names, instead of naming network interfaces (locator) or end-hosts (end-host identifiers) as in LISP [RFC6830].
- o (iii) data is retrieved by an end-point by issuing requests and a node accepts a data packet from an ingress interface if and only if at least one matching request packet is stored in the local cache of the node, otherwise the data packet is dropped;
- o (iv) basic security services are provided by the architecture: authenticity of the data producer and data integrity. A cryptographic signature over a security envelop is computed by the producer (using its own private key) and must be verified by the consumer (using the producer's public key). The security envelop can be as small as a single data packet or cover groups of packets using the technique of the transport manifest [MAN].

2. Architecture

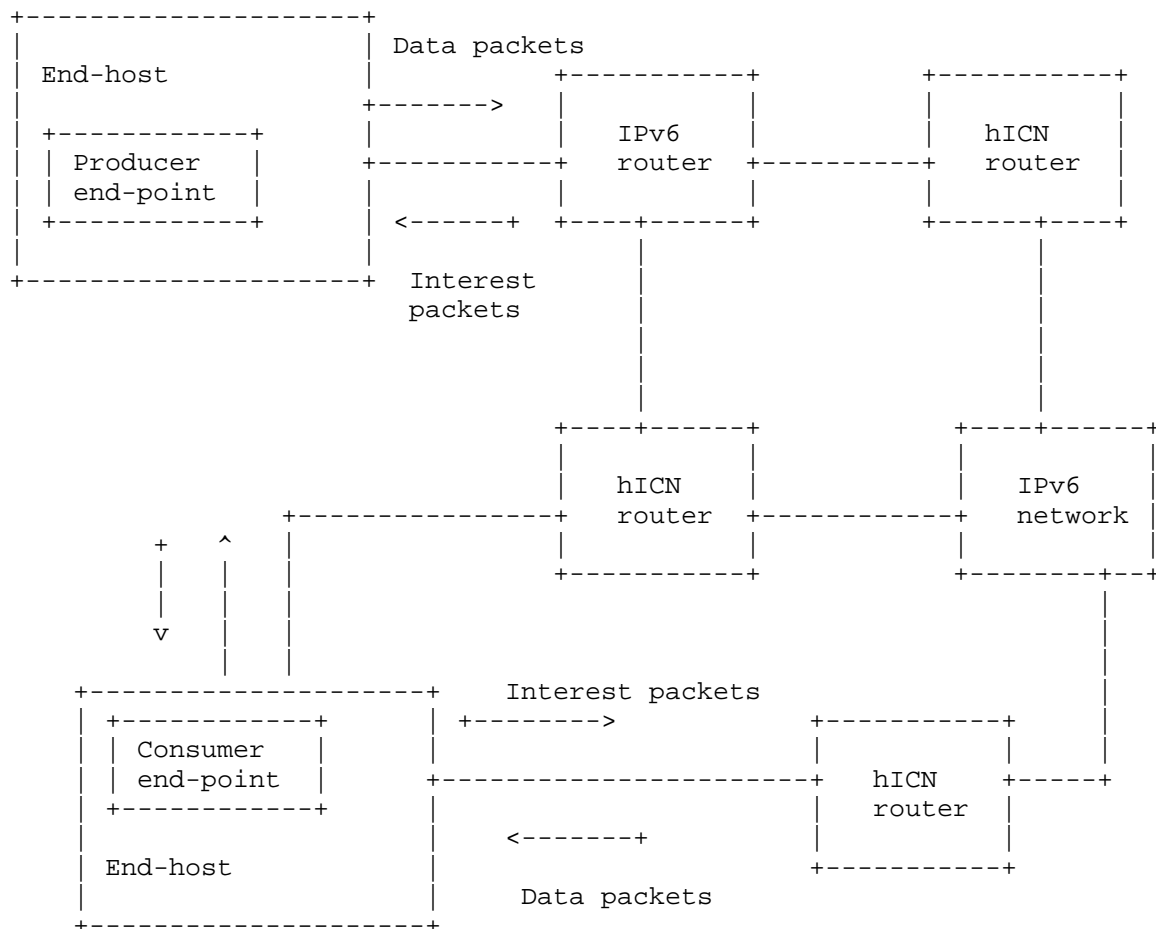


Figure 1: General overview of an hICN end-to-end communication.

The communication model described in this document covers the transport and the network layer.

The network layer includes the forwarding plane only and does not consider the routing plane. hICN network layer is about using the IPv6 FIB to determine a next hop router to forward requests or using a local packet cache to determine if an incoming request can be satisfied locally. The hICN forwarding plane takes care of forwarding replies by using information stored in cached requests. The packet pipeline of an hICN node always includes a lookup in a packet cache for both requests and replies. The packet cache is a mandatory component that is added to the usual IPv6 packet processing pipeline. Requests and replies carry an immutable data name end-to-

end, in packet header fields as described in the following sections. Moreover, requests and replies carry locators as mutable packet header fields. A locator, i.e. an interface identifier, is changed every time a packet is sent to another hICN node. A detailed description of how locators are modified along the path between end-points is reported in the following sections.

It is assumed that existing routing protocols, working for IPv6, should be reused as much as possible as they are. Improvements to existing routing protocols are out of scope and might be developed in other documents to better exploit features made available by the hICN forwarding plane. For instance, hICN forwarding plane can take advantage of the ability of a routing protocol to provide multiple routes for a given destination or more generally compute routes for destinations that are multi-instantiated [MIR]. This topic is important but out of scope for this document.

The hICN network architecture can run on top of any link-layer that supports IPv6. hICN data names are globally routable names which are visible to the transport layer end-points. Conversely, the transport layer has no visibility of addresses of network interfaces. The network layer forwards two kind of protocol data units: the request and the reply, called interest and data packets.

The hICN network layer offers a communication service to the transport layer in the end-points by means of a local unidirectional channel that we call local or application face. This channel is used by the transport layer to send requests and receive replies or to send replies upon receptions of requests.

A transport end-point is always bound to a unidirectional channel that is used to either send data or receive data. The former end-point is called data producer while the latter data consumer. The producer end-point produces data under a location independent name, which is an IPv6 prefix. A consumer end-point fetches data by using the non ambiguous name as provided by the producer. The producer end-point is responsible for managing the usage of the prefix in terms of provisioning, association to applications and its revocation.

The transport end-point offers two kinds of services to applications: a producer and a consumer service. The service is instantiated in the application by opening communication sockets with an API to perform basic transport service operations: allocation, initialization, configuration, data transmission and reception.

The producer and consumer sockets can implement different types of services such as stream or datagram, reliable or unreliable. In all

cases all transport services are connection-less, meaning that a producer transport service produces named data in a socket memory that is accessible by any valid request coming from one or multiple consumers. The consumer, on the other hand, retrieve named data using location independent names which are not tied to any interface identifier (also called locator). This transport model allows to implement reliable consumer mobility without any special mobility management protocol. hICN supports communication of multi-homed end-hosts without any special treatment in the transport layer. The hICN network layer can also implement robust usage of multi-path forwarding in IPv6 networks as balanced request/reply flows self-stabilize network congestion see [CCN],[NDN], [RAQ] .

A data packet is an IPv6 packet with a transport layer header carrying data from an application that produces data. An interest packet is an IPv6 packet with a transport layer header and is used to unambiguously fetch a data packet from a producer end point.

2.1. End-points

In hICN we introduce two new kinds of endpoints: the producer and the consumer. We identify two kind of communication sockets each with a specific API: the producer and consumer sockets. These socket types are designed to exchange data in a multi-point to multi-point manner. In (h)ICN we have the same concept that is applied to a network where memories are distributed across the communication path. The first memory in the path is the production buffer of the producer end-point that forges Data Packets and copies them into a shared memory isolated into a namespace. Consumer sockets can retrieve data from such memory by using the (h)ICN network layer. The model just described is an inter-process communication example (IPC) that requires data to cross a communication network by using a transport protocol.

The way consumers and producers synchronize depends on application requirements and the transport layer exposes a variety of services: stream/datagram, reliable/unreliable, with or without latency budgets etc. Independently of the specif requirements of the applications, producer sockets always perform data segmentation from the upper layer into Data Packets, as well as compute digital signatures on the packet security envelop. This envelop can also be computed across a group of packets, by including a cryptographic hash of each packet into the transport manifest, and eventually signing only such manifest.

The consumer socket, on the other end, always performs reassembly of Data Packets, hash integrity verification and signature verification. This is common to all architectures in (h)ICN. The usual assumption

is that the producer socket uses an authentic-able identity while using namespaces that it has been assigned. The end-point must be able to manage the mapping of her identity and the allocated namespace by issuing digital certificates about the mapping. The consumer end-point must retrieve the associated certificate to perform the basic operations. It is out of scope for this document how to design and implement a scalable system to perform such certificate operations.

A detailed description of transport end-points is out of scope for this document.

2.2. Naming

In hICN, two name components are defined: the name prefix and the name suffix. The name prefix is used to identify an application object, a service or in general an application level source of data in the network. This is incarnated by a listening socket that binds to the name prefix. The name suffix is used to index segmented data within the scope of the name prefix used by the application.

For instance an RTP [RFC3550] source with a given SSRC can be mapped into a name prefix. Single RTP sequence numbers can be mapped into name suffixes. For example an HTTP server can listen to a name prefix to serve HTTP requests. An HTTP reply with large payload with require the transport layer to segment the application data unit according to an MTU. Name suffixes are used to index each segment in the socket stream.

More details about how to use hICN to transport HTTP or RTP will be given in a different document.

2.2.1. Name prefix

The format of an hICN name prefix is the following:

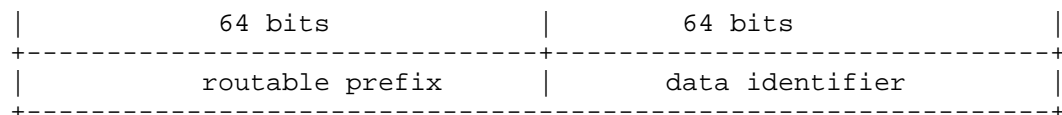


Figure 2: hICN IPv6 name prefix.

It is composed of a routable IPv6 /64 prefix as per [RFC3587] which SHOULD be globally routable. The data identifier is encoded in 64 bits. An application can use several identifiers if needed.

From the description given above, the name prefix is a location independent name encoded in an IPv6 address.

2.2.2. Name Suffix

The name suffix is used by the transport layer protocol to index segments. The segment **MUST** be indexed in the end-points and in the network with the same suffix. This implies that there is one transport segment per IP packet and that IP fragmentation is not allowed. It is up to the producer end-point to determine how to perform segmentation depending on the use case. An MTU path discovery protocol for hICN is out of scope of this document and additional work is required to extend existing protocols or design new ones.

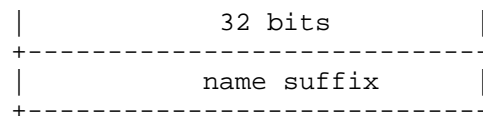


Figure 3: hICN name suffix.

2.3. Packet Format

Two protocol data units are defined below: the interest (request) and the data (reply).

They are composed of a network and transport header. The transport header is the same for both packet types while the network header is slightly different.

2.3.1. Interest Packet

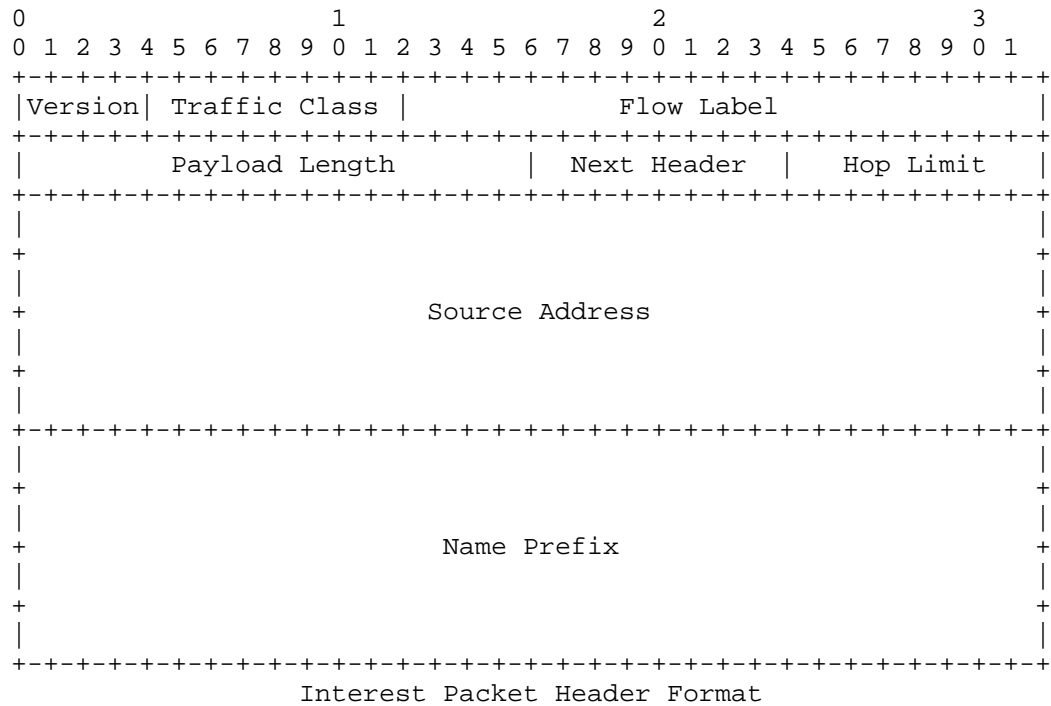


Figure 4: IPv6 interest packet L3 header.

Source Address: 128-bit address of the originator of the packet
(possibly not the end-host but a previous hICN node).

Name Prefix: 128-bit name prefix of the intended service.

2.3.2. Data Packet

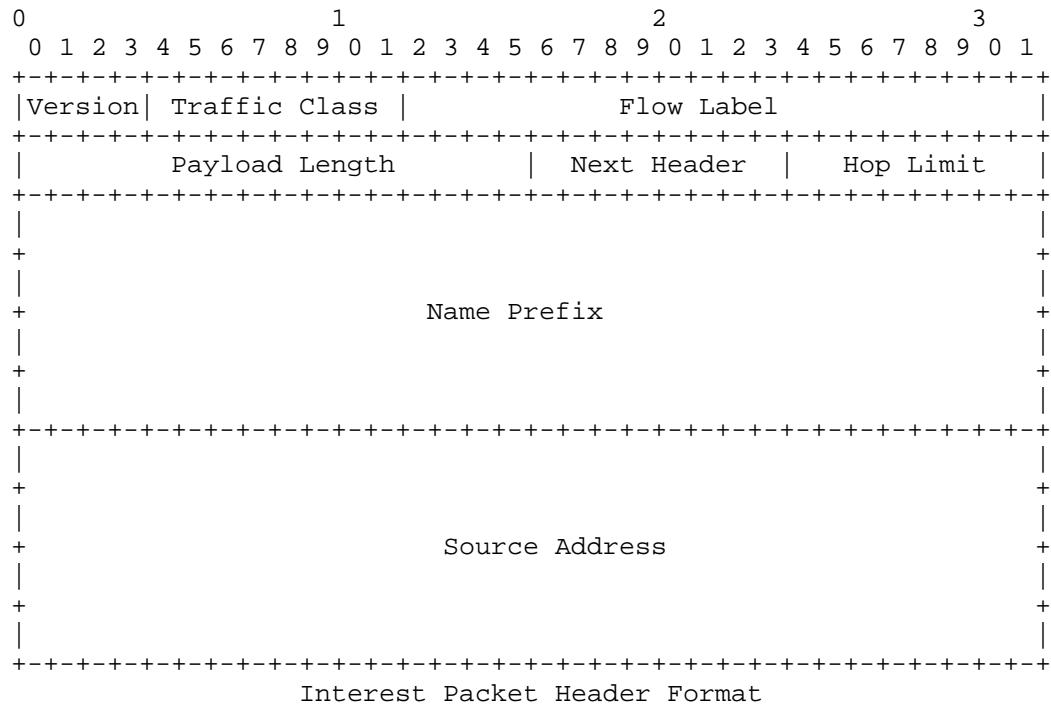


Figure 5: IPv6 data packet L3 header.

Name Prefix: 128-bit name prefix of the intended service.

Source Address: 128-bit address of the destination of the packet (possibly not the end-host but the next hICN node).

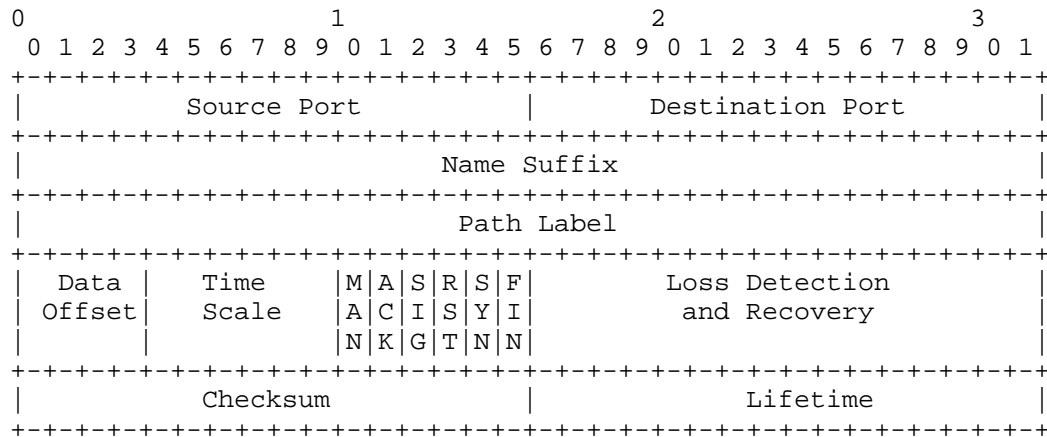


Figure 6: Transport header for data and interest packets.

Name Suffix:	32-bit name-suffix of the packet (possibly not the end-host but a previous hICN node).
Path Label:	32-bit label used to carry an encoding of the path between the consumer and data responder, be it an intermediate node or the producer end-point.
Time Scale:	6-bit natural number in the range 1-64 used as a scaling factor for time calculations. If not null it is used to scale lifetime.
Manifest:	flag to indicate the packet carries a transport manifest in the payload.
Signature:	flag to indicate the packet carries an authentication header with a signature. Interest packet do not carry signatures.
Loss Detection: and Recovery	16-bit natural number used to implement data sequencing on per adjacency basis to detect and recover losses using the mechanism WLDR described in {{WLD}}.
Lifetime:	16-bit unsigned integer to carry the packet lifetime in milliseconds.
Checksum:	Updated using RFC 1624.

The following sections describe the components of an hICN node and the packet processing operations.

2.4. Packet cache

The packet cache is a router local memory used to temporarily store requests and reply. The simplest incarnation of the packet cache MUST index packets by full name, i.e. the concatenation of the name prefix and suffix. Insertion and deletion of packets in the cache is described below.

2.5. Forwarding

The forwarding path in hICN is composed of two components: the interest and data path. Requests and replies are processed at the hICN node in a different way. Both forwarding paths require a packet cache to be incorporated into the router. The cache is used to temporarily store requests and replies for a relatively short amount of time.

By caching a request in an hICN node, the reply can be transmitted back to the right nodes as the source address field in the interest contains the interface identifier of the hICN node having transmitted the request. Replies are optionally cached if needed.

This means that the interest forwarding path is based on lookups in the IP FIB just like any other IP packet, with the additional processing due to a cache lookup to check if the actual reply is already present in the local cache for expedited reply.

On the other hand, data packet forwarding is similar to label swapping [RFC3031], being the packet name identifier (prefix plus suffix) the forwarding label. The next hop for the reply in transit is indeed selected by using information in a cached matching request.

The name prefix in the packet header is never modified along the path for both requests and replies, while the locator, i.e. the interface identifier written in the source or destination address field, for interest or data packets respectively, is modified at the egress of the router as reported below.

2.5.1. Interest Path

At the router ingress the incoming interest packet I is parsed to obtain the name prefix and the name suffix. An exact match look up is made in the packet cache using the full packet name as key. Based on the outcome of the lookup the following options are possible:

1. at least one match is found.

1.1. If one match is a data packet D, other matches are ignored, and D is prepared for transmission by setting D's destination address with I's source address. D is passed to the egress to further processing before transmission. For instance the next-hop MAY be selected by using the router IPv6 FIB (longest prefix match). The IPv6 FIB lookup MIGHT be saved in case the next-hop can be derived directly from information previously derived by processing the incoming interest packet I. I is eventually dropped.

1.2. There is one or multiple matches and all are interest packets.

- * One matching interest has the same source address and I is classified as duplicate and further processed as duplicate.
- * Matching interest packets have different source addresses and I is classified as filtered and stored in the cache.

2. a match is not found and I passed to the egress for further processing to determine the next-hop by using the router IP FIB.

Notice that the destination address field in the interest packet is polymorphic as it has two different types based on the data structured it is looked-up against. It has the type of a location independent name while used to find a match in the packet cache and it has an address prefix type to find the next-hop in the IPv6 FIB. Polymorphism is transparent for the forwarding plane while it has several implications in the control plane.

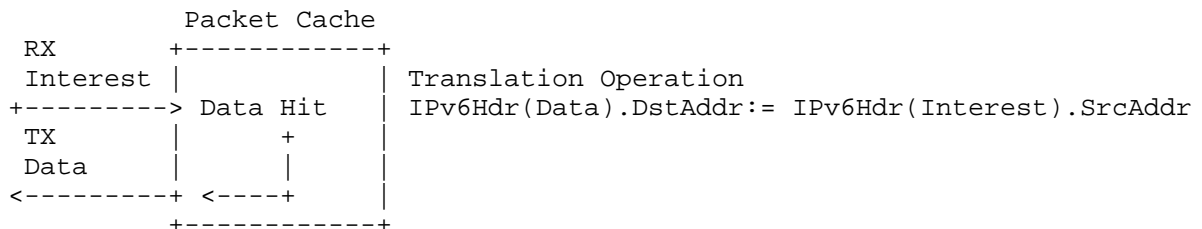


Figure 7: The interest packet hits a matching data packet in the packet cache.

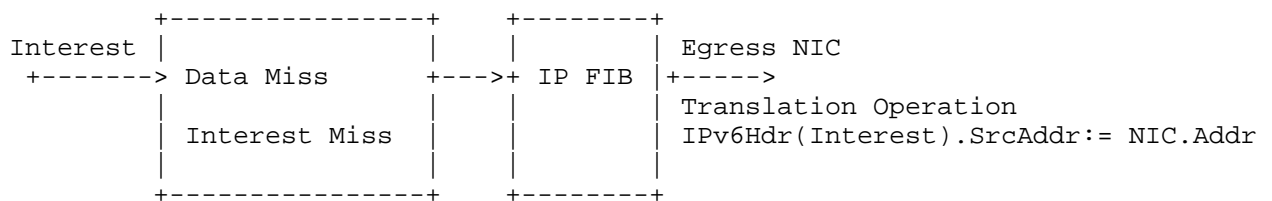


Figure 8: The interest packet finds no match in the packet cache and is processed to find a next-hop.

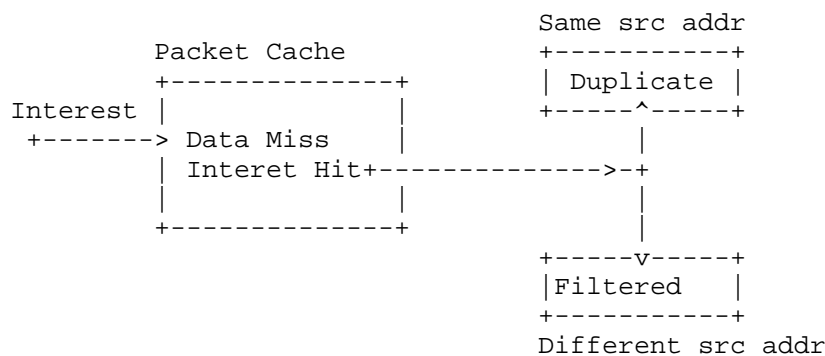


Figure 9: The interest packet hits an interest packet in the packet cache.

2.5.2. Data Path

At the router ingress the incoming data packet D is parsed to obtain the name prefix and the name suffix. An exact match look up is made in the packet cache using the full packet name as key. Based on the outcome of the lookup the following options are possible:

1. one or multiple matching interest packets are found 1.1. The data packet D is cloned to have as many copies as the number of matching interests including D. The destination address field of each copy of D is set with the source address field of each interest packet. All copies are passed to the egress to further processing before transmission in order to find each data packet's next-hop.
2. No matching is an interest packet and the D is dropped.

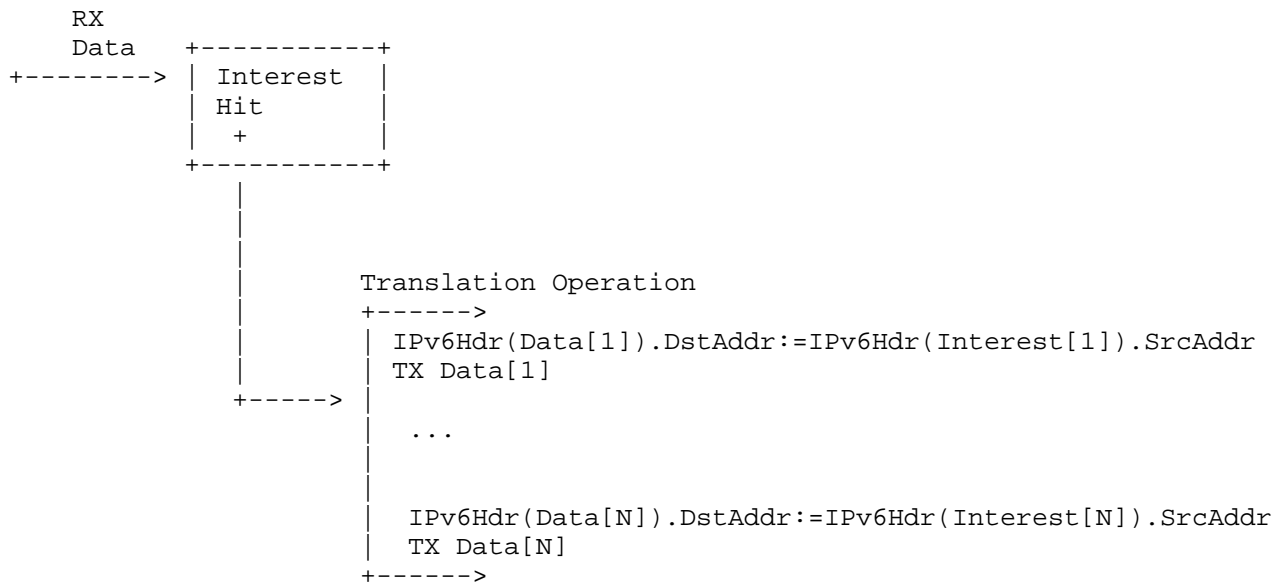


Figure 10: The data packet hits an interest packet in the packet cache.

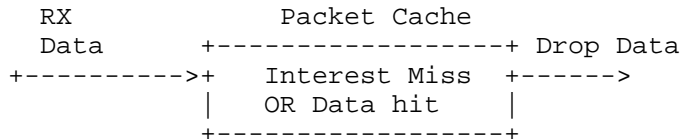


Figure 11: The data packet is drop in case no interest match is found in the packet cache.

3. Security

hICN inherits ICN data-centric security model: integrity, data-origin authenticity and confidentiality are tied to the content rather than to the channel.

Integrity and data-origin authenticity are provided through a digital signature computed by the producer and included in each data packet. Integrity and data-origin authenticity are provided in two ways using two approaches: the first one based on IP Authenticated Header [RFC4302] and the second one based on transport manifests. Notice that the IP AH is not used as an IPv6 extension header as it is appended after the transport header. However the choice of the IP AH

has been made in order to exploit existing protocol implementations in the end-points.

When using IP AH, the signature is computed over

- o (i) IP or extension header fields either immutable in transit or that are predictable in value upon arrival at the consumer,
- o (ii) the AH header with the signature field set to zero. We recall that in hICN the destination header field is not immutable nor predictable and must be set to zero for the signature computation. We also point out the AH is placed after the TCP header in order to prevent any kind of filtering from network devices such as middleboxes.

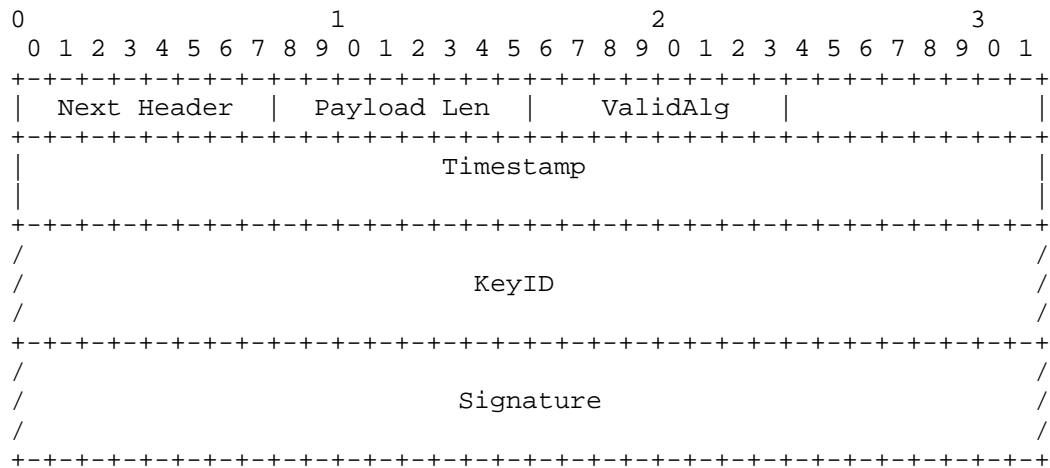


Figure 12: The IP authentication header appended after the transport header to carry packet signatures.

- ValidAlg: 8-bit index to indicate which validation algorithm must be used to verify the signature.
- Timestamp: 64-bit time stamp that indicates the validity of the signature.
- KeyID: 256-bit key identifier.
- Signature: Variable length field that carries the cryptographic signature of the security envelope. It is 128 bytes for RSA-1024, 256 bytes for RSA-2048, 56 bytes for EDCSA 192, 72 bytes for ECDSA 256.

The transport manifest is a L4 entity computed at the producer which contains the list of names of a group of data packets to convey to the consumer. hICN cryptographic hashes of data packets are then computed instead of signatures. The hashes are computed on immutable fields as explained above when using the IP AH. Moreover, the manifest must be signed to guarantee a level of security equivalent to packet-wise signatures.

hiCN is oblivious of the trust model adopted by consumers and works with any of the existing proposals.

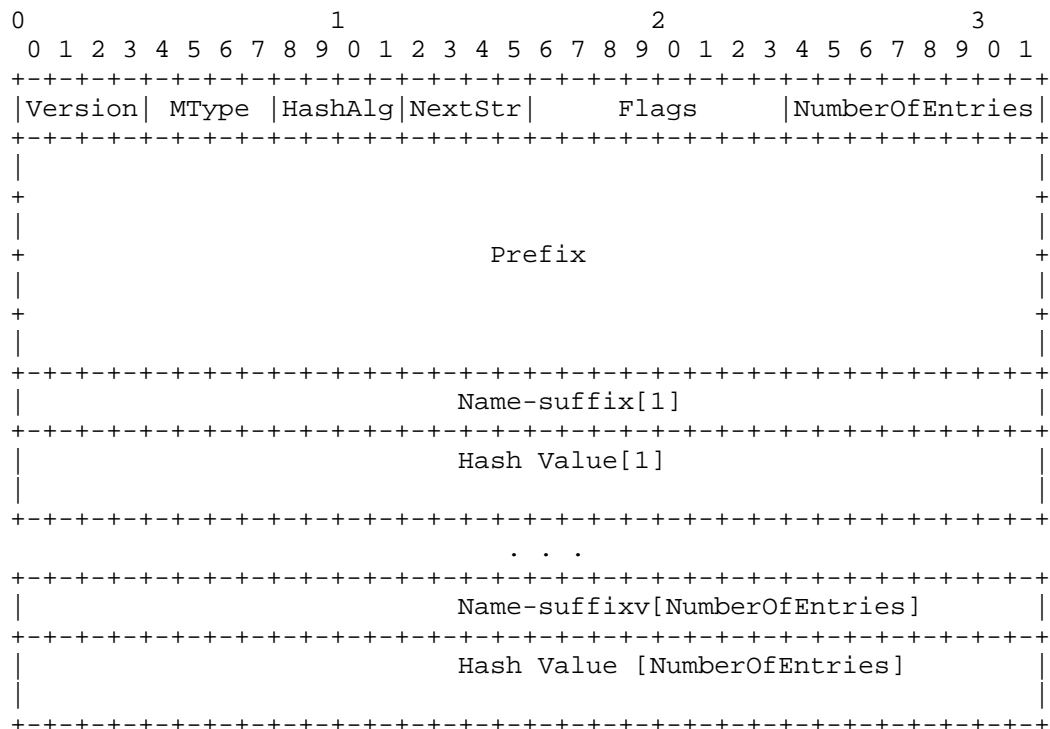


Figure 13: The transport manifest, generated by the producer end-point for the consumer end-point, contains names, integrity hashes and is signed with the producer end-point private key

Version: 8-bit index to indicate which validation algorithm must be used to verify the signature.

MType: 64-bit time stamp that indicates the validity of the signature.

HashAlg: 256-bit key identifier.

NextStr: Encode an operator use to predict the name-suffix sequence

Flags: Flags.

NumberOfEntries: 8-bit field that encodes the number of packets indexed in the manifest.

Name-prefix: 128-bit field carrying the name-prefix common to all packets indexed in the manifest.

Name-suffix: 32-bit field carrying the name-suffix.

Hash-value: 256-bit field carrying the SHA-256 hash of the packet security envelop.

4. IANA Considerations

There are no IANA considerations in this specification.

5. Acknowledgements

The authors would like to thank David Ward, David Oran, Paul Polakos, Mark Townsley, Mauro Sardara and Alberto Compagno for suggestions on how to improve the architecture and the current document.

6. References

6.1. Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC1081] Rose, M., "Post Office Protocol: Version 3", RFC 1081, DOI 10.17487/RFC1081, November 1988, <<https://www.rfc-editor.org/info/rfc1081>>.

- [RFC1624] Rijsinghani, A., Ed., "Computation of the Internet Checksum via Incremental Update", RFC 1624, DOI 10.17487/RFC1624, May 1994, <<https://www.rfc-editor.org/info/rfc1624>>.
- [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>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <<https://www.rfc-editor.org/info/rfc3587>>.
- [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>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [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>>.

6.2. Informative References

- [CCN] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking named content", Proceedings of the 5th international conference on Emerging networking experiments and technologies - CoNEXT '09, DOI 10.1145/1658939.1658941, 2009.

- [I-D.irtf-icnrg-ccnxmessages]
Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-07 (work in progress), March 2018.
- [I-D.irtf-icnrg-ccnxsemantics]
Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", draft-irtf-icnrg-ccnxsemantics-07 (work in progress), March 2018.
- [I-D.irtf-icnrg-mapme]
Auge, J., Carofiglio, G., Muscariello, L., and M. Papalini, "MAP-Me : Managing Anchorless Mobility in Content Centric Networking", draft-irtf-icnrg-mapme-00 (work in progress), March 2018.
- [I-D.irtf-icnrg-terminology]
Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology", draft-irtf-icnrg-terminology-00 (work in progress), December 2017.
- [MAN]
Baugher, M., Davie, B., Narayanan, A., and D. Oran, "Self-verifying names for read-only named data", 2012 Proceedings IEEE INFOCOM Workshops, DOI 10.1109/infcomw.2012.6193505, March 2012.
- [MIR]
Garcia-Luna-Aceves, J., Martinez-Castillo, J., and R. Menchaca-Mendez, "Routing to Multi-Instantiated Destinations: Principles, Practice, and Applications", IEEE Transactions on Mobile Computing Vol. 17, pp. 1696-1709, DOI 10.1109/tmc.2017.2734658, July 2018.
- [NDN]
Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., and B. Zhang, "Named data networking", ACM SIGCOMM Computer Communication Review Vol. 44, pp. 66-73, DOI 10.1145/2656877.2656887, July 2014.
- [RAQ]
Carofiglio, G., Gallo, M., Muscariello, L., Papalini, M., and , "Optimal multipath congestion control and request forwarding in Information-Centric Networks", 2013 21st IEEE International Conference on Network Protocols (ICNP), DOI 10.1109/icnp.2013.6733576, October 2013.

[WLD] Carofiglio, G., Muscariello, L., Papalini, M., Rozhnova, N., and X. Zeng, "Leveraging ICN In-network Control for Loss Detection and Recovery in Wireless Mobile networks", Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16, DOI 10.1145/2984356.2984361, 2016.

Authors' Addresses

Luca Muscariello
Cisco Systems Inc.

Email: lumuscar@cisco.com

Giovanna Carofiglio
Cisco Systems Inc.

Email: gcarofig@cisco.com

Jordan Auge
Cisco Systems Inc.

Email: augjorda@cisco.com

Michele Papalini
Cisco Systems Inc.

Email: mpapal@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

E. Nordmark
Zededa
July 2, 2018

Privacy issues in ID/locator separation systems
draft-nordmark-id-loc-privacy-00

Abstract

There exists several protocols and proposals for identifier/locator split which have some form of control plane by which participating nodes can use to share their current id to locator information with their peers. This document explores some of the privacy considerations for such a system.

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 <http://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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 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	2
2. Keywords and Terminology	3
3. Assumptions	3
4. Threats against Privacy	4
4.1. Location Privacy	4
4.2. Movement Privacy	4
5. Not everybody all the time	4
5.1. Optimized routing	4
5.2. Family and Friends	5
5.3. Business Assets	5
6. Boundary between ID/locator part and rest of Internet	5
7. Security Considerations	5
8. IANA Considerations	6
9. Normative References	6
Author's Address	6

1. Introduction

When the IP address is separated, one way or another, into an identifier and a locator there is typically a need to be able to look up an identifier to find possible locators which can be used to reach the identified endpoint. If such a system (think distributed database) was publicly available while identifiers are assigned to devices such as mobile phones which have a strong binding with an individual, then this would introduce additional privacy considerations which do not exist in the absence of the ID/locator split.

Without an ID/locator split a device is already providing its IP address (in the form of a source address) to any network device along the path, and also to the remote endpoint. That endpoint in particular might use IP geolocation databases to get a pretty good idea of where its peer is located, for instance to offer information and/or advertising relevant to that location.

However, such a device (e.g., a laptop or smartphone connected over WiFi) move e.g., from home to a coffee shop, the IP address changes. This makes it harder for network devices along the paths to realize that the its is the same mobile device. And if the mobile device is not retaining cookies or logged into websites, those remote peers would also have some difficulty determining it is the same mobile device. Furthermore, a mobile device which is using typical cellular network technologies end up with an IP address, at least as seen by remote peers outside of the cellular network, which is associated with the cellular operator but does not necessarily indicate a particular location of the mobile device.

Note that even if the IP address isn't always useful to track a mobile device today, there are several mechanisms higher in the stack which can do this. For instance cookies or SSL sessions, applications which share GPS location, or operators who offer additional location information (for instance based on which cellular base station a mobile device is using) to business partners.

With that baseline in mind, let's look at what additional privacy considerations can be introduced by a system which provides ID to locator mappings.

2. Keywords and Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

3. Assumptions

We assume that there are benefits associated with sharing ID to locator mappings with some peers sometimes. Those benefits can be

- o Lower latency and higher bandwidth: If two peer devices have some locators which are topologically closer, then sharing all the locators means that the devices can find a short path (fewer hops and/or shorter round-trip time), or find a path which offer higher throughput, then if the devices only shared some form of default locator.
- o Higher availability and robustness: If two peer devices share all their locators, then if there is some network outage the devices can autonomously discover a working path using the different locator pairs.

However, those benefits do not imply that it is a good idea to always share all of the locators with everybody. That would make tracking by third parties trivial.

A device can obfuscate itself by, instead of using a single long-lived identifier, using multiple short-lived identifiers. In that case the value to the ID/locator binding for any particular identifier would be lower. However, this assumes that the device can ensure unlinkability between the different identifiers it is using either concurrently or over time. Also, some of the benefits above implicitly assume that there can be some long-lived sessions or associations between pairs of identifiers. For instance, if a device would need to go fetch the current identifier of its peer from some remote system, then it might not experience improved robustness since that fetch might depend on the failed external connectivity. Thus we

believe that we can explore the core of the ID/locator privacy issue by looking at long-lived identifiers.

4. Threats against Privacy

This is the first version of this draft so this is very preliminary. But there seems to be at least two different privacy threats relating to ID/locator mapping systems.

4.1. Location Privacy

If a third party can at any time determine the IP location of some identifier, then the device can at one point be IP geolocated at home, and later a coffee shop.

4.2. Movement Privacy

If a third party can determine that an identifier has changed locator(s) at time T, then even without knowing the particular locators before and after, it can correlate this movement event with other information (e.g., security cameras) to create a binding between the identifier and a person.

5. Not everybody all the time

In order to see the benefits about but minimizing the privacy implication one can explore limiting to which peers and when the ID/locator binding are exposed.

A few initial examples help illustrate this.

5.1. Optimized routing

If some operator of a network where there is a large amount of mobility wants to ensure efficient routing, then a ID/locator split approach might make sense. Such a system can potentially be limited to the set of devices (routers etc) which are under the operators control. If this is the case, then the ID/locator mapping system can provide access control so that only those trusted devices can access the mappings.

Note that from a privacy perspective this isn't any different than the same operator using a link-state routing protocol to share host routes for all the mobile devices. In that case all participants in the link-state protocol can determine the location (attached to which router) and notice any mobility events. Of course, there are significant non-privacy differences between those two approaches.

Exposing the ID/locator mapping to attached devices (e.g., any mobile devices which wouldn't be trusted to participate in the link-state routing counterpart approach), will change the privacy implications.

5.2. Family and Friends

There are cases where it is quite reasonable to share location information with other family members or friends. For instance, young children might run applications which enable their parents to track them on their way to/from school. And I might share my location with friends so we can more easily find each other while out on town.

Today such location sharing happens at an application layer using GPS coordinates. But while such sharing is in effect, it wouldn't be unreasonable to also consider sharing IP locators to make it more efficient or more robust to e.g., route a video feed from one device to another.

5.3. Business Assets

In the area of Industrial IoT there are cases where an asset owner might want to ensure that their assets can communicate efficiently and robustly. In many cases those assets might be decoupled from any persons, but there can still be strong reasons to not share the ID/locator binding with third parties, such as enabling competitors to determine the number of deployed devices in a particular IP prefix.

6. Boundary between ID/locator part and rest of Internet

If the access to the ID/locator mapping are restricted as suggested above, then most of the potential peer devices would not have access to the ID/locator mappings. This means that there has to be a demarcation point between the part of the network which can access the ID/locator mappings for a particular identifier and the one which can not. There might be several choices how to handle this such as still using an ID/locator system but pointing a locator for some fixed anchor point, or injecting routing prefixes for the ID prefixes into the normal routing system, or not providing any stable locators across this boundary; only allow ephemeral IP addresses per session or otherwise limited exposure.

7. Security Considerations

This document discusses privacy considerations, but does not explore any security considerations.

8. IANA Considerations

There are no IANA actions needed for this document.

9. 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>>.

Author's Address

Erik Nordmark
Zededa
Santa Clara, CA
USA

Email: nordmark@sonic.net

Internet Area Working Group
Internet-Draft
Intended status: Experimental
Expires: September 6, 2018

V. Olteanu
D. Niculescu
University Politehnica of Bucharest
March 05, 2018

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-02

Abstract

The SOCKS protocol is used primarily to proxy TCP connections to arbitrary destinations via the use of a proxy server. Under the latest version of the protocol (version 5), it takes 2 RTTs (or 3, if authentication is used) before data can flow between the client and the server.

This memo proposes SOCKS version 6, which reduces the number of RTTs used, takes full advantage of TCP Fast Open, and adds support for 0-RTT authentication.

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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 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. Revision log	3
2. Requirements language	5
3. Mode of operation	5
4. Connection Requests	6
5. Version Mismatch Replies	7
6. Authentication Replies	8
7. Operation Replies	9
7.1. Handling CONNECT	10
7.2. Handling BIND	11
7.3. Handling UDP ASSOCIATE	11
8. SOCKS Options	11
8.1. Socket options	11
8.1.1. TFO options	12
8.1.2. Multipath TCP options	13
8.1.3. MPTCP Scheduler options	14
8.2. Authentication Method options	14
8.3. Authentication Data options	15
8.4. Idempotence options	16
8.4.1. Requesting a fresh token window	17
8.4.2. Spending a token	18
8.4.3. Handling Token Window Advertisements	19
8.5. Salt options	19
9. Username/Password Authentication	20
10. Security Considerations	20
10.1. Large requests	20
10.2. Replay attacks	21
10.3. Identical request profiling	21
11. IANA Considerations	21
12. Acknowledgements	22
13. References	22
13.1. Normative References	22
13.2. Informative References	22
Authors' Addresses	22

1. Introduction

Versions 4 and 5 [RFC1928] of the SOCKS protocol were developed two decades ago and are in widespread use for circuit level gateways or as circumvention tools, and enjoy wide support and usage from various software, such as web browsers, SSH clients, and proxifiers.

However, their design needs an update in order to take advantage of the new features of transport protocols, such as TCP Fast Open [RFC7413], or to better assist newer transport protocols, such as MPTCP [RFC6824].

One of the main issues faced by SOCKS version 5 is that, when taking into account the TCP handshake, method negotiation, authentication, connection request and grant, it may take up to 5 RTTs for a data exchange to take place at the application layer. This is especially costly in networks with a large delay at the access layer, such as 3G, 4G, or satellite.

The desire to reduce the number of RTTs manifests itself in the design of newer security protocols. TLS version 1.3 [I-D.ietf-tls-tls13] defines a zero round trip (0-RTT) handshake mode for connections if the client and server had previously communicated.

TCP Fast Open [RFC7413] is a TCP option that allows TCP to send data in the SYN and receive a response in the first ACK, and aims at obtaining a data response in one RTT. The SOCKS protocol needs to concern itself with at least two TFO deployment scenarios: First, when TFO is available end-to-end (at the client, at the proxy, and at the server); second, when TFO is active between the client and the proxy, but not at the server.

This document describes the SOCKS protocol version 6. The key improvements over SOCKS version 5 are:

- o The client sends as much information upfront as possible, and does not wait for the authentication process to conclude before requesting the creation of a socket.
- o The connection request also mimics the semantics of TCP Fast Open [RFC7413]. As part of the connection request, the client can supply the potential payload for the initial SYN that is sent out to the server.
- o The protocol can be extended via options without breaking backward-compatibility.
- o The protocol can leverage the aforementioned options to support 0-RTT authentication schemes.

1.1. Revision log

Typos and minor clarifications are not listed.

draft-02

- o Made support for Idempotence options mandatory for proxies.
- o Clarified what happens when proxies can not or will not issue tokens.
- o Limited token windows to $2^{31} - 1$.
- o Fixed definition of "less than" for tokens.
- o NOOP commands now trigger Operation Replies.
- o Renamed Authentication options to Authentication Data options.
- o Authentication Data options are no longer mandatory.
- o Authentication methods are now advertised via options.
- o Shifted some Request fields.
- o Option range for vendor-specific options.
- o Socket options.
- o Password authentication.
- o Salt options.

draft-01

- o Added this section.
- o Support for idempotent commands.
- o Removed version numbers from operation replies.
- o Request port number for SOCKS over TLS. Deprecate encryption/encapsulation within SOCKS.
- o Added Version Mismatch Replies.
- o Renamed the AUTH command to NOOP.
- o Shifted some fields to make requests and operation replies easier to parse.

2. 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].

3. Mode of operation

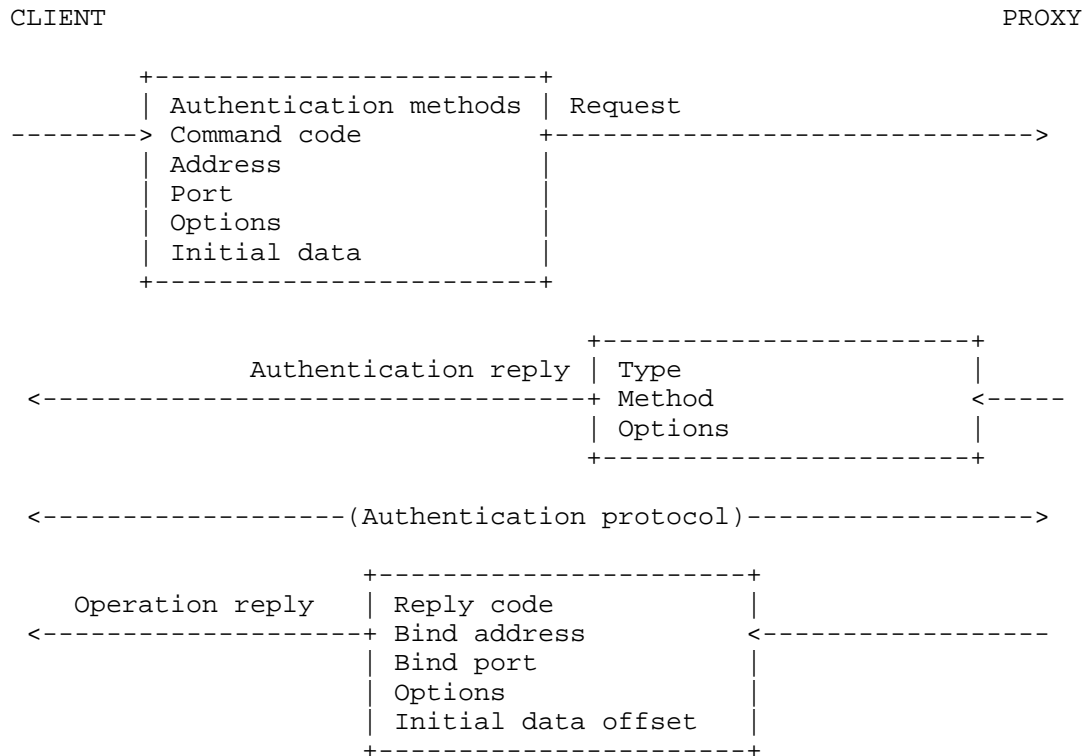


Figure 1: The SOCKS version 6 protocol message exchange

When a TCP-based client wishes to establish a connection to a server, it must open a TCP connection to the appropriate SOCKS port on the SOCKS proxy. The client then enters a negotiation phase, by sending the request in figure Figure 1, that contains, in addition to fields present in SOCKS 5 [RFC1928], fields that facilitate low RTT usage and faster authentication negotiation.

Next, the server sends an authentication reply. If the request did not contain the necessary authentication information, the proxy

indicates an authentication method that must proceed. This may trigger a longer authentication sequence that could include tokens for ulterior faster authentications. The part labeled "Authentication protocol" is specific to the authentication method employed and is not expected to be employed for every connection between a client and its proxy server. The authentication protocol typically takes up 1 RTT or more.

If the authentication is successful, an operation reply is generated by the proxy. It indicates whether the proxy was successful in creating the requested socket or not.

In the fast case, when authentication is properly set up, the proxy attempts to create the socket immediately after the receipt of the request, thus achieving an operational connection in one RTT (provided TFO functionality is available at the client, proxy, and server).

4. Connection Requests

The client starts by sending a request to the proxy.

Version		Command	Port	Address	Address
Major	Minor	Code		Type	
1	1	1	2	1	Variable
Number of Options		Options	Initial Data Size	Initial Data	
1		Variable	2	Variable	

Figure 2: SOCKS 6 Request

- o Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- o Command Code:
 - * 0x00 NOOP: authenticate the client and do nothing.
 - * 0x01 CONNECT: requests the establishment of a TCP connection.
 - * 0x02 BIND: requests the establishment of a TCP port binding.

- * 0x03 UDP ASSOCIATE: requests a UDP port association.
- o Address Type:
 - * 0x01: IPv4
 - * 0x03: Domain Name
 - * 0x04: IPv6
- o Address: this field's format depends on the address type:
 - * IPv4: a 4-byte IPv4 address
 - * Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
 - * IPv6: a 16-byte IPv6 address
- o Port: the port in network byte order.
- o Number of Options: the number of SOCKS options that appear in the Options field.
- o Options: see Section 8.
- o Initial Data Size: A two-byte number in network byte order. In case of NOOP, BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, this is the number of bytes of initial data that are supplied in the following field.
- o Initial Data: The first octets of the data stream.

Clients can advertise their supported authentication methods by including an Authentication Method option (see Section 8.2).

The server MAY truncate the initial data to an arbitrary size and disregard the rest. This is will be communicated later to the client, should the authentication process be successful (see Section 7). As such, server implementations do not have to buffer the initial data while waiting for the (potentially malicious) client to authenticate.

5. Version Mismatch Replies

Upon receipt of a request starting with a version number other than 6.0, the proxy sends the following response:

Version	
Major	Minor
1	1

Figure 3: SOCKS 6 Version Mismatch Reply

- o Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.

A client MUST close the connection after receiving such a reply.

6. Authentication Replies

Upon receipt of a valid request, the proxy sends an Authentication Reply:

Version		Type	Method	Number of Options	Options
Major	Minor				
1	1	1	1	1	Variable

Figure 4: SOCKS 6 Authentication Reply

- o Version: The major byte MUST be set to 0x06, and the minor byte MUST be set to 0x00.
- o Type:
 - * 0x00: authentication successful.
 - * 0x01: further authentication needed.
- o Method: The chosen authentication method.
- o Number of Options: the number of SOCKS options that appear in the Options field.
- o Options: see Section 8.

Multihomed clients SHOULD cache the chosen method on a per-interface basis and SHOULD NOT include Authentication Data options related to

any other methods in further requests originating from the same interface.

If the server signals that further authentication is needed and selects "No Acceptable Methods", the client **MUST** close the connection.

The client and proxy begin a method-specific negotiation. During such negotiations, the proxy **MAY** supply information that allows the client to authenticate a future request using an Authentication Data option. The client and proxy **SHOULD NOT** negotiate the encryption of the application data. Descriptions of such negotiations are beyond the scope of this memo.

7. Operation Replies

After the authentication negotiations are complete, the server sends an Operation Reply:

Reply Code	Address Type	Bind Port	Bind Address	Initial Data Offset
1	1	2	Variable	2

Number of Options	Options
1	Variable

Figure 5: SOCKS 6 Operation Reply

o Reply Code:

- * 0x00: Success
- * 0x01: General SOCKS server failure
- * 0x02: Connection not allowed by ruleset
- * 0x03: Network unreachable
- * 0x04: Host unreachable
- * 0x05: Connection refused

- * 0x06: TTL expired
- * 0x07: Command not supported
- * 0x08: Address type not supported
- o Address Type:
 - * 0x01: IPv4
 - * 0x03: Domain Name
 - * 0x04: IPv6
- o Bind Address: the proxy bound address in the following format:
 - * IPv4: a 4-byte IPv4 address
 - * Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
 - * IPv6: a 16-byte IPv6 address
- o Bind Port: the proxy bound port in network byte order.
- o Number of Options: the number of SOCKS options that appear in the Options field.
- o Options: see Section 8.
- o Initial Data Offset: A two-byte number in network byte order. In case of BIND or UDP ASSOCIATE, this field MUST be set to 0. In case of CONNECT, it represents the offset in the plain data stream from which the client is expected to continue sending data.

If the proxy returns a reply code other than "Success", the client MUST close the connection.

If the client issued an NOOP command, the client MUST close the connection after receiving the Operation Reply.

7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass. The client MUST resume the data stream at the offset indicated by the Initial Data Offset field.

7.2. Handling BIND

In case the client has issued a BIND request, it must wait for a second Operation reply from the proxy, which signifies that a host has connected to the bound port. The Bind Address and Bind Port fields contain the address and port of the connecting host. Afterwards, application data may pass.

7.3. Handling UDP ASSOCIATE

The relay of UDP packets is handled exactly as in SOCKS 5 [RFC1928].

8. SOCKS Options

SOCKS options have the following format:

+-----+-----+		
Kind Length Option Data		
+-----+-----+		
1	1	Variable
+-----+-----+		

Figure 6: SOCKS 6 Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: The length of the option.
- o Option Data: The contents are specific to each option kind.

8.1. Socket options

Socket options are be used by clients to alter the behavior of the sockets created by the proxy. A socket option can affect either the proxy's socket on the client-proxy leg or on the proxy-server leg. Clients can only place Socket options inside SOCKS Requests.

Proxies MAY include Socket options in their Operation Replies to signal their sockets' behavior. Said options MAY be unsolicited, i. e. the proxy MAY send them to signal behaviour that was not explicitly requested by the client.

Kind	Length	Leg	Level	Code	Data
1	1	2 bits	6 bits	1	Variable

Figure 7: Socket Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: The length of the option.
- o Leg:
 - * 0x1: Client-Proxy Leg
 - * 0x2: Proxy-Server Leg
 - * 0x3: Both Legs
- o Level:
 - * 0x01: Socket
 - * 0x02: IPv4
 - * 0x03: IPv6
 - * 0x04: TCP
 - * 0x05: UDP
- o Code: Option code
- o Data: Option-specific data

8.1.1. TFO options

Kind	Length	Leg	Level	Code
1	1	2 bits	6 bits	1

Figure 8: TFO Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: MUST be 4.
- o Leg: MUST be 0x2 (Proxy-Server Leg).
- o Level: 0x04 (TCP).
- o Code: 0x17

If a SOCKS Request contains a TFO option, the proxy SHOULD attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. Otherwise, the proxy MUST NOT attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command.

In case of a CONNECT command, the proxy MAY include a TFO option in the Operation reply if TFO was attempted, the operation succeeded and the remote server supports TFO. In case of a BIND command, the proxy MAY include a TFO option in the first Operation reply to signal that it will accept an incoming TFO connection.

8.1.2. Multipath TCP options

In case of a CONNECT command, the proxy can inform the client that the connection to the server is an MPTCP connection.

Kind	Length	Leg	Level	Code
1	1	2 bits	6 bits	1

Figure 9: Multipath TCP Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: 4.
- o Leg: MUST be 0x2 (Proxy-Server Leg).
- o Level: 0x04 (TCP).
- o Code: 0x2a

8.1.3. MPTCP Scheduler options

In case of a CONNECT or BIND command, a client can use an MPTCP Scheduler option to indicate its preferred scheduler for the connection.

A proxy can use an MPTCP Scheduler option to inform the client about what scheduler is in use.

Kind	Length	Leg	Level	Code	Scheduler
1	1	2 bits	6 bits	1	1

Figure 10: MPTCP Scheduler Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: MUST be 5.
- o Leg: Either 0x01, 0x02, or 0x03 (Client-Proxy, Proxy-Client or Both legs).
- o Level: 0x04 (TCP).
- o Code: 0x2b
- o Scheduler:
 - * 0x00: Default
 - * 0x01: Round-Robin
 - * 0x02: Redundant

8.2. Authentication Method options

Authentication Method options are used by clients to advertise supported authentication methods. They can be part of SOCKS Requests.

Kind	Length	Methods
1	1	Variable

Figure 11: Authentication Method Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: The length of the option.
- o Methods: One byte per advertised method. Method numbers are assigned by IANA.

Clients MUST support the "No authentication required" method.
 Clients MAY omit advertising the "No authentication required" option.

8.3. Authentication Data options

Authentication Data options carry method-specific authentication data. They can be part of SOCKS Requests and Authentication Replies.

Authentication Data options have the following format:

Kind	Length	Method	Authentication Data
1	1	1	Variable

Figure 12: Authentication Data Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: The length of the option.
- o Method: The number of the authentication method. These numbers are assigned by IANA.
- o Authentication Data: The contents are specific to each method.

Clients MAY omit advertising authentication methods for which they have included at least an Authentication Data option.

8.4. Idempotence options

To protect against duplicate SOCKS Requests, authenticated clients can request, and then spend, idempotence tokens. A token can only be spent on a single SOCKS request.

Tokens are 4-byte unsigned integers in a modular 4-byte space. Therefore, if x and y are tokens, x is less than y if $0 < (y - x) < 2^{31}$ in unsigned 32-bit arithmetic.

Proxies grant contiguous ranges of tokens called token windows. Token windows are defined by their base (the first token in the range) and size. Windows can be shifted (i. e. have their base increased, while retaining their size) unilaterally by the proxy.

Requesting and spending tokens is done via Idempotence options:

Kind	Length	Type	Option Data
1	1	1	Variable

Figure 13: Idempotence Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: The length of the option.
- o Type:
 - * 0x00: Token Request
 - * 0x01: Token Window Advertisement
 - * 0x02: Token Expenditure
 - * 0x03: Token Expenditure Reply
- o Option Data: The contents are specific to each type.

All proxy implementations MUST support Idempotence options, even if they do not issue token windows.

8.4.1. Requesting a fresh token window

A client can obtain a fresh window of tokens by sending a Token Request option as part of a SOCKS Request:

Kind	Length	Type	Window Size
1	1	1	4

Figure 14: Token Request

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: 7
- o Type: 0x00 (Token Request)
- o Window Size: The requested window size.

If a token window is issued, the proxy then includes a Token Window Advertisement option in the corresponding Operation Reply:

Kind	Length	Type	Window Base	Window Size
1	1	1	4	4

Figure 15: Token Window Advertisement

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: 11
- o Type: 0x01 (Token Grant)
- o Window Base: The first token in the window.
- o Window Size: The window size. This value SHOULD be lower or equal to the requested window size. Window sizes MUST be less than 2^{31} .

If no token window is issued, the proxy MUST silently ignore the Token Request.

8.4.2. Spending a token

The client can attempt to spend a token by including a Token Expenditure option in its SOCKS request:

Kind	Length	Type	Token
1	1	1	4

Figure 16: Token Expenditure

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: 7
- o Type: 0x02 (Token Expenditure)
- o Token: The token being spent.

Clients SHOULD prioritize spending the smaller tokens.

The server responds by sending a Token Expenditure Reply option as part of the Operation Reply:

Kind	Length	Type	Response Code
1	1	1	1

Figure 17: Token Expenditure Response

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: 4
- o Type: 0x03 (Token Expenditure Response)
- o Response Code:
 - * 0x00: Success: The token was spent successfully.
 - * 0x01: No Window: The proxy does not have a token window associated with the client.

- * 0x02: Out of Window: The token is not within the window.
- * 0x03: Duplicate: The token has already been spent.

If eligible, the token is spent as soon as the client authenticates. If the token is not eligible for spending, the proxy **MUST NOT** attempt to honor the client's SOCKS Request; further, it **MUST** indicate a General SOCKS server failure in the Operation Reply.

Proxy implementations **SHOULD** also send a Token Window Advertisement if:

- o the token is out of window, or
- o by the proxy's internal logic, successfully spending the token caused the window to shift.

Proxy implementations **SHOULD NOT** shift the window's base beyond the highest unspent token.

Proxy implementations **MAY** include a Token Window Advertisement in any Operation Reply.

8.4.3. Handling Token Window Advertisements

Even though the proxy increases the window's base monotonically, there is no mechanism whereby a SOCKS client can receive the Token Window Advertisements in order. As such, clients **SHOULD** disregard unsolicited Token Window Advertisements with a Window Base less than the previously known value.

8.5. Salt options

Clients can use Salt options so that otherwise identical requests are unique. (See Section 10.3.)

+-----+-----+			
Kind	Length	Salt	
+-----+-----+-----+			
1	1	4	
+-----+-----+-----+			

Figure 18: Salt Option

- o Kind: **MUST** be allocated by IANA. (See Section 11.)
- o Length: 6

- o Salt: An arbitrary value.

Proxies MUST silently ignore Salt options.

9. Username/Password Authentication

Username/Password authentication is carried out as in [RFC1929].

Clients can also attempt to authenticate by placing the Username/Password request in an Authentication Data Option, provided that it is no longer than 252 bytes.

Kind	Length	Method	Username/Password request
1	1	1	Variable

Figure 19: Password authentication via a SOCKS Option

- o Kind: MUST be allocated by IANA. (See Section 11.)
- o Length: The length of the option.
- o Method: 0x02 (Username/Password).
- o Username/Password request: The Username/Password request, as described in [RFC1929].

10. Security Considerations

10.1. Large requests

Given the format of the request message, a malicious client could craft a request that is in excess of 100 KB and proxies could be prone to DDoS attacks.

To mitigate such attacks, proxy implementations SHOULD be able to incrementally parse the requests. Proxies MAY close the connection to the client if:

- o the request is not fully received after a certain timeout, or
- o the number of options exceeds an imposed hard cap, or
- o the total size of the options exceeds an imposed hard cap, or

- o the size of the initial data exceeds a hard cap.

Further, the server MAY choose not to buffer any initial data beyond what would be expected to fit in a TFO SYN's payload.

10.2. Replay attacks

In TLS 1.3, early data (which is likely to contain a full SOCKS request) is prone to replay attacks.

While Token Expenditure options can be used to mitigate replay attacks, the initial Token Request is still vulnerable. As such, client implementations SHOULD NOT make use of TLS early data when sending a Token Request.

10.3. Identical request profiling

If sent via TLS early data, identical SOCKS requests can also be identical on the wire. An attacker with the capability to capture a client's SOCKS traffic can attempt to profile it by identifying identical requests.

A client can use Salt options to make all of its requests unique.

11. IANA Considerations

This document requests that IANA allocate 1-byte option codes for SOCKS 6 options. Further, this document requests option codes for:

- o Socket options
- o Authentication Method options
- o Authentication Data options
- o Idempotence options
- o Salt options
- o Vendor-specific options

This document also requests that IANA allocate a port for SOCKS over TLS.

12. Acknowledgements

The protocol described in this draft builds upon and is a direct continuation of SOCKS 5 [RFC1928].

13. References

13.1. Normative References

[RFC1929] Leech, M., "Username/Password Authentication for SOCKS V5", RFC 1929, DOI 10.17487/RFC1929, March 1996, <<https://www.rfc-editor.org/info/rfc1929>>.

[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>>.

13.2. Informative References

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-26 (work in progress), March 2018.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <<https://www.rfc-editor.org/info/rfc1928>>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<https://www.rfc-editor.org/info/rfc7413>>.

Authors' Addresses

Vladimir Olteanu
University Politehnica of Bucharest

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu
University Politehnica of Bucharest

Email: dragos.niculescu@cs.pub.ro

intarea
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

R. Patterson
Sky UK
M. Abrahamsson
T-Systems
July 02, 2018

IP over Ethernet (IPoE) Session Health Checking
draft-patterson-intarea-ipoe-health-04

Abstract

PPP over Ethernet clients have the functionality to detect path unavailability by using PPP Keepalives. IP over Ethernet does not have this functionality, and it is not specified in the IETF when an IP over Ethernet client should consider its WAN connectivity down, unless there is a physical layer link down event.

This document describes a mechanism for IP over Ethernet clients to achieve connectivity validation, similar to that of PPP over Ethernet, by using BFD Echo, or ARP and Neighbor Discovery functions.

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 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. Requirements Language	3
1.2. Terminology	3
2. Alternative Mitigations	4
3. IPoE Health Checks	4
3.1. Parameters	4
3.2. BFD Echo	5
3.3. Neighbor Discovery	5
3.4. ARP	5
3.5. Alternative Target	6
3.6. Passive Checks	6
4. IPoE Health Check DHCP Options	7
4.1. DHCPv6	7
4.2. DHCPv4	8
5. Action Behaviours	9
5.1. Behaviour 0: Renew (Default)	9
5.2. Behaviour 1: Rebind	10
5.3. Behaviour 2: Solicit	10
5.4. Behaviour 3: Expire & Release	11
5.5. LAN Considerations	11
6. Multihomed Clients	11
6.1. Neighbor Discovery	12
6.2. ARP	12
7. Security Considerations	12
8. IANA Considerations	13
9. Acknowledgements	13
10. Appendix A. Changes from -00	13
11. Appendix B. Changes from -01	14
12. Appendix C. Changes from -02	14
13. Appendix D. Changes from -03	14
14. References	14
14.1. Normative References	14
14.2. Informative References	15
14.3. URIs	16
Authors' Addresses	16

1. Introduction

PPP [RFC1661] makes use of regular LCP echos and replies to continually test the data link layer, if the peer fails to respond to a predetermined number of LCP echos, the PPP session is terminated and will return to the Link Dead phase, ready for reestablishing. IPoE currently lacks this functionality.

Physical link state change on an IPoE client can trigger the renewing of a DHCP lease, however any indirect upstream network changes are not visible to the IPoE client.

An outage or planned maintenance work on, for example, a Broadband Network Gateways (BNG) or intermediate DHCP Relay, can leave an IPoE client with a stale DHCP lease for up to the Valid Lifetime.

IPoE Session Health Check allows for an IPoE client to proactively or passively monitor the state of upstream connectivity, and defines several actions that may be taken to help the client recover.

[TR-146], Section 6.2 describes this problem, while [TR-124] identifies some requirements to solve the problem.

Several vendors have implemented subscriber connectivity checking on their BNG, using ARP and Neighbor Discovery as per Sections 6.2.4 and 6.2.5 of [TR-146]. This allows the BNG to detect loss of connectivity and to update local session state and DHCP lease bindings. Without reciprocal checking, this puts the CE at further risk of being in a stale state.

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.

1.2. Terminology

This document makes use of the following terms:

- o BNG: Broadband Network Gateway. Often also running a DHCP server or relay.
- o CE: Customer Equipment. aka. Customer Premise Equipment (CPE), Residential Gateway (RG).
- o IPoE: IP over Ethernet.

- o IPoE Client: A network device, often a CE, running a DHCPv4 and/or DHCPv6 client.
- o IPoE Health Check: The name of the process described in this document.

2. Alternative Mitigations

- o Short DHCP lease times reduce the time a client may be left in a stale state, but scale poorly, putting extra load on the DHCP server.
- o Broadband Forum's [TR-146] and [TR-124] discuss this problem and recommend the use of BFD echo [RFC5880]. This document acknowledges TR-146 and recommends the use of BFD echo for health checks, but like the Broadband Forum, acknowledges that it is not widely available within consumer CEs. This document also introduces alternative actions, as the renew approach taken in TR-146 is susceptible to the issues described in Behaviour 0 (Section 5.1).
- o For planned maintenance, network engineers could include DHCPv4 Force Renew [RFC3203] or DHCPv6 Reconfigure [RFC3315]-bis in their maintenance plans, however neither of these have been widely adopted by CE or BNG vendors due to authentication complexity.

3. IPoE Health Checks

3.1. Parameters

IPoE Health Check uses the following parameters:

- o Interval (Integer): The frequency in seconds, which health checks are sent by the IPoE client. Default value: 120 seconds.
- o Retry Interval (Integer): The frequency in seconds, which health checks are sent by the IPoE client, after a failure. Default value: 10 seconds.
- o Limit (Integer): The number of failed consecutive checks before an action is taken. Default value: 3.
- o Behaviour (Integer): Specifies what actions are to be taken when triggered. Default value: 0.
- o Passive (Boolean): Forces passive health checks instead of active. Default: False.

- o Layer2 (Boolean): Forces layer 2 health checks instead of BFD echo. Default: False.
- o Alternative Target Address (IP address): Overrides the default target of health checks.

An IPoE client supporting IPoE Health Check SHOULD begin sending health checks at the Interval specified, upon successful binding of a lease that contains a valid IPoE Health Check DHCP Option (Section 4).

An IPoE client MAY be statically configured for IPoE health checks. Non-default static parameters SHOULD override any signalled via a dynamic means (e.g, DHCP or TR-69).

An IPoE client MAY use default parameters in lieu of manually configured, or dynamically signalled parameters (DHCP for example). Statically configured or dynamically signalled parameters SHOULD override any default parameters.

3.2. BFD Echo

Unless instructed otherwise, an IPoE client SHOULD use BFD Echo [RFC5880] as the health check mechanism.

The use of BFD Echo as the health check mechanism provides the added benefit of validating the DHCP lease state, proving layer 3 as well as layer 2 connectivity.

If BFD echo messages are used, the destination IP address MUST be locally bound on the IPoE client and SHOULD be from the lease triggering the IPoE Health Check.

3.3. Neighbor Discovery

If an IPoE client with active DHCPv6 leases is unable to send BFD echo messages or IPoE client is explicitly configured, it MUST send Neighbor Solicits (Section 4.3 of [RFC4861]) for the target address. If no Alternative Target Address is set, the target address SHOULD be the default gateway as obtained from the Operating System. Neighbor Solicits SHOULD be sent at the frequency set by the Interval parameter under normal conditions or Retry Interval when a failure is encountered (Section 3.1).

3.4. ARP

If an IPoE client with active DHCPv4 leases is unable to send BFD echo messages or IPoE client is explicitly configured, it MUST send ARP requests [RFC0826] for the target address. If no Alternative

Target Address is set, the target address SHOULD be the client's default gateway, as received within the DHCPv4 Option 3 Router option of the lease [RFC2132].

ARP requests SHOULD be sent at the frequency set by the Interval parameter under normal conditions or Retry Interval when a failure is encountered (Section 3.1).

3.5. Alternative Target

An alternative target IP address MAY be included in the IPoE health check DHCP option, or statically configured. If an alternative target address is specified, it MUST be used as the target for health checks instead of the default gateway.

If an Alternative Target Address provided is outside of a locally attached route, health checks SHOULD implicitly fail until a matching local route is installed. If a matching locally attached route is subsequently installed, health checks SHOULD continue as normal.

The Alternative Target IP Address MUST be a valid IP address. An IPoE client MUST silently discard loopback and multicast addresses.

3.6. Passive Checks

If an IPoE client is unable to proactively send health checks itself, it SHOULD passively check the operating system's ARP and Neighbor cache tables.

In IPoE Health Check passive mode, alternate target addresses outside of locally attached routes MUST NOT be supported.

Passive IPoE health checks SHOULD use the health check parameters signalled by DHCP or configured statically (Section 3.1). The IPoE client SHOULD passively check the ARP or Neighbor cache tables for the target address, every Interval seconds. If the neighbor entry is in state INCOMPLETE, subsequent checks SHOULD BE made every Retry Interval seconds. When Limit checks have failed, the specified IPoE Health Check Action MUST be taken.

Passive-only mode can be forced either by local configuration, or by a DHCP server setting the Passive flag in the DHCP option. If passive-only mode has been set, the IPoE client MUST only use passive checking for that particular lease health check.

4. IPoE Health Check DHCP Options

This document defines a new option for both DHCPv4 and DHCPv6 servers to signal suggested health check parameters to clients. IPoE clients SHOULD use these values when no statically configured parameters have been defined.

The option data fields are common between DHCPv6 and DHCPv4, with the exception of the alternate target address field, which is 32 bits in the DHCPv4 option and 128 bits in the DHCPv6 option.

4.1. DHCPv6

For DHCPv6, this option (Figure 1) MUST be within a specific Identity Association as an IPoE client MAY have multiple IAs with different health check parameters.

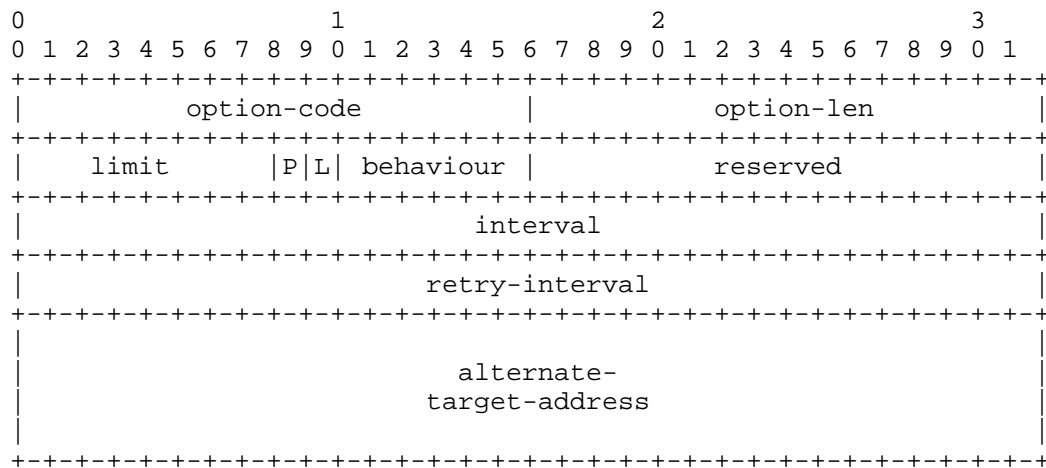


Figure 1: DHCPv6 IPoE Check Option Format

The description of the fields is as follows:

option-code: OPTION_IPOE_HEALTH (TBA1).
 option-len: 28.
 limit: Consecutive failed checks, before an action is taken.
 P: Passive Flag. Force passive-only health checks.
 L: Layer 2 Flag. Force ARP/ND-only health checks.
 behaviour: The following behaviours are defined:
 0: Trigger Renew (Section 5.1).
 1: Trigger Rebind (Section 5.2).
 2: Expire lease, start solicit phase (Section 5.3).
 3: Release (Section 5.4).
 4 - 63: Unassigned.
 New behaviour codes can be assigned in the future.
 interval: Indicates how often a health check should be sent when
 no failure is encountered. Expressed in units of seconds.
 retry-interval: Indicates how often a health check should be sent
 after a previous failure. Expressed in units of seconds.
 alternate-target-address: Optional health check target address.
 MUST always present; it MUST be set to zero if no
 alternate IP address is to be used. This field MUST NOT
 include loopback or multicast addresses.

4.2. DHCPv4

The DHCPv4 client can retrieve IPoE Health Check information by including OPTION_IPOE_HEALTH in a Parameter Request List option [RFC2132].

Figure 2 shows the DHCPv4 IPoE Health Check option.

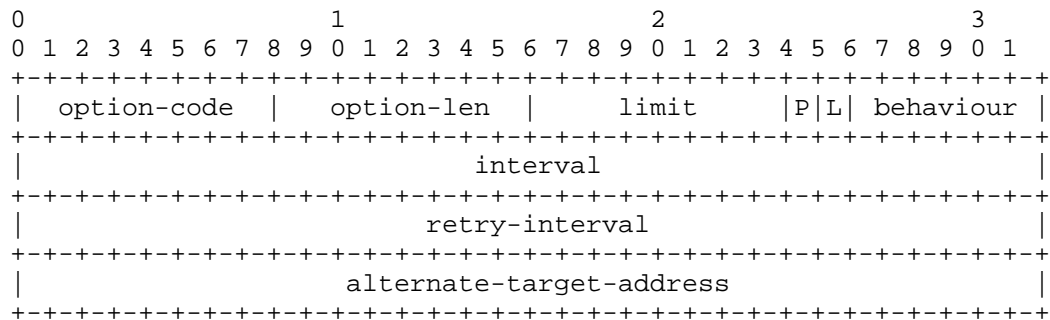


Figure 2: DHCPv4 IPoE Health Check Option Format

The description of the fields is as follows:

option-code: OPTION_IPOE_HEALTH (TBA2).
option-len: 14.
limit: Consecutive failed checks, before an action is taken.
P: Passive Flag. Force passive-only health checks.
L: Layer 2 Flag. Force ARP/ND-only health checks.
behaviour: The following behaviours are defined:
0: Trigger Renew (Section 5.1).
1: Trigger Rebind (Section 5.2).
2: Expire lease, start solicit phase (Section 5.3).
3: Release (Section 5.4).
4 - 63: Unassigned.
New behaviour codes can be assigned in the future.
interval: Indicates how often a health check should be sent when
no failure is encountered. Expressed in units of seconds.
retry-interval: Indicates how often a health check should be sent
after a previous failure. Expressed in units of seconds.
alternate-target-address: Optional health check target address.
MUST always present; it MUST be set to zero if no
alternate IP address is to be used. This field MUST NOT
include loopback or multicast addresses.

5. Action Behaviours

IPoE Health Check defines four configurable behaviours once the timeout threshold has been reached. All these behaviours make use of existing procedures outlined in [RFC2131], Section 4.4.5 for DHCPv4, and [RFC3315]-bis, Sections 18.2.4, 18.2.5 for DHCPv6.

The IPoE Health Check behaviour MAY be signalled per lease by DHCP, or statically configured. Statically configured, non-default, behaviour settings SHOULD take precedence over those signalled by DHCP.

When triggering a DISCOVER or SOLICIT, an IPoE client may also choose to use Rapid Commit [RFC4039], [RFC3315], Section 22.14 to help expedite the recovery process.

5.1. Behaviour 0: Renew (Default)

After Limit (Section 3.1) consecutive check failures, the IPoE client MUST set T1 of the specified lease, to zero. This will trigger a RENEW to the original DHCP server, as per [RFC3315]-bis and [RFC2131].

If connectivity to the original DHCP server has recovered, and the server can satisfy the request, the lease may be renewed and timers updated.

If the original DHCP server cannot satisfy the request, it may reject the request, to which the DHCP client should begin discovery or solicit phase anew.

Neither of the above two responses are guaranteed, and as such, an administrator may elect to use one of the below additional behaviours to help expedite the IPoE client's recovery process.

Unless specified otherwise, additional actions **MUST** also be taken if the IPoE Health Check DHCP option Behaviour bits are non-zero. Some behaviours may offer alternative actions instead of compound ones, they will state this specifically.

5.2. Behaviour 1: Rebind

If the Behaviour field is set to 1, T2 **MUST** also be set to zero, along with T1. This tells the IPoE client to immediately move to the rebind phase, attempting to renew the lease from any available server.

This method can be useful in a resilient layer 2 access topology, with multiple active DHCP servers.

5.3. Behaviour 2: Solicit

If the Behaviour field is set to 2, T1 and T2 **MUST** both be set to zero as per previous behaviours.

The IPoE client **MUST** skip the renew and rebind phases, moving straight to the discovery or solicit phase.

The IPoE client **MUST NOT** send a DHCP RELEASE.

The IPoE client **MUST** keep the address or prefix in the preferred state until the preferred lifetime expires, and **MUST** keep the address or prefix until the valid lifetime expires.

The IPoE client **SHOULD** include the lease address or prefix in the DISCOVER or SOLICIT.

The DUID and IAID **MUST** be the same as used in the current lease.

This method can be useful when using DHCP servers that silently discard unknown renew attempts instead of sending back a DHCPv4 NAK or DHCPv6 Reply.

5.4. Behaviour 3: Expire & Release

If the Behaviour field is set to 3, T1, T2, and Valid Lifetime MUST all be set to 0, and the IPoE Client MUST send a DHCP RELEASE message as per [RFC2131], Section 3.1 for DHCPv4 and [RFC3315]-bis, Section 18.2.7

Once the RELEASE process has completed, the client returns to the discovery or solicit phase.

If the IPoE client is already in the renew or rebind state when Behaviour 3 is triggered, the client MUST cease renew or rebind attempts and wait for any outstanding messages to time out before sending a RELEASE. If an outstanding renew or rebind attempt is successful, the IPoE client MUST update T1, T2 and lease lifetimes appropriately, and MUST NOT continue with Behaviour 3.

This method can be useful to clean out state within the network. For example, a DHCP relay may be left with stale lease information after an outage or maintenance on a DHCP server.

5.5. LAN Considerations

If all DHCPv6 leases have expired, either naturally or proactively with IPoE health checks, it is expected that an IPoE client acting as a router, would withdraw itself as a default router on the LAN, following requirement G-5 of [RFC7084], Section 4.1.

6. Multihomed Clients

An IPoE client may have multiple leases from the same, or different DHCP servers. These leases may have different IPoE health check parameters, and health checks MUST be treated distinctly, tracking the particular lease that they belong to.

Each distinct IPoE health check MUST use an appropriate target address as per IPoE Health Check (Section 3).

If an IPoE client is configured with multiple IPoE Health Checks that use the same target address, it SHOULD suppress additional checks, preferring the parameters with the lowest timeout value.
I.e. $\text{Timeout} = \text{Interval} + \text{Retry Interval} * (\text{Limit} - 1)$

Local network administrators may choose to override DHCP-signalled parameters in order to facilitate appropriate IPoE Health Check operation in a multihomed environment.

6.1. Neighbor Discovery

As DHCPv6 does not convey default gateway or other routing information, an IPoE client using the ND health check method SHOULD obtain the target address by querying the operating system for default routes.

If multiple default routes exist, ND-based IPoE health checks SHOULD attempt to match the target address to the lease by the interface the lease is bound to.

If only a single default route exists, and that default route is not routed out the interface the lease was bound to, ND-based health checks for that particular lease SHOULD be paused.

6.2. ARP

ARP-based IPoE health checks for DHCPv4 make use of the default gateway address specified in the lease. As a route for each gateway should exist regardless of current route preference, health checks SHOULD be run for each lease that is configured for IPoE health check.

7. Security Considerations

IPoE Health Check frequency would typically be controlled by the network using DHCP options, but overly aggressive, statically configured IPoE Health Checks, could have an adverse impact. For example, this may induce an overload on the IP access nodes.

ARP and Neighbor Discovery may be subject to protections outlined in [RFC6192]; Interval and Retry Interval values SHOULD NOT be set too aggressively and upstream routers SHOULD ensure that sufficient quantities of this traffic are permitted to safely ingress the control plane.

Unlike ARP and ND, BFD echo uses an IP packet destined for the IPoE client, the peer forwards the packet back to the IPoE client without any local processing.

Behaviour 2 (Section 5.3) introduces a privacy risk, possibly leaking lease information if the IPoE client has been moved to a different network, e.g., from one fixed line provider to another.

8. IANA Considerations

IANA is requested to assign a new DHCPv6 Option Code in the registry maintained in <http://www.iana.org/assignments/dhcpv6-parameters> [1]:

Option Name	Value

OPTION_IPOE_HEALTH	TBA1

Also, IANA is requested to assign the following new DHCPv4 Option Code in the registry maintained in <http://www.iana.org/assignments/bootp-dhcp-parameters> [2]:

Option Name	Tag	Data Length	Meaning

OPTION_IPOE_HEALTH	TBA2	14	Provides a set of IPoE check configuration information.

This document requests IANA to create a new registry called "IPoE Check Behaviours" (under <https://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml>).

This registry is initially populated with the following values:

Value	Description	Reference
0	Trigger Renew	[ThisRFC]
1	Trigger Rebind	[ThisRFC]
2	Expire lease	[ThisRFC]
3	Release	[ThisRFC]

Values in the range 4-63 are assigned via the "IETF Review" policy defined in [RFC8126].

The same registry is used for both DHCPv4 and DHCPv6.

9. Acknowledgements

The authors would like thank Ian Farrer, Dusan Mudric, Mohamed Boucadair, Jean-Yves Cloarec, Bernie Volz, Dave Freedman, and Job Snijders for their review and comments on this and previous versions of this document.

10. Appendix A. Changes from -00

This section should be removed by the RFC Editor.

- o Added reference to TR-146.

- o Added BFD Echo section, and wording to prefer it as the health check mechanism over ARP/ND, if available.

11. Appendix B. Changes from -01

This section should be removed by the RFC Editor.

- o Emphasised preference for use of BFD echo as the health check mechanism.
- o Removed lifetime expiration from Behaviour 2 and clarified usage.
- o Updated Behaviour 3 with instructions for whilst mid-renew/rebind.
- o Reworded multihoming section.
- o Added Acknowledgements.

12. Appendix C. Changes from -02

This section should be removed by the RFC Editor.

- o Added DHCP option flag to force ARP/ND for health checks.
- o Populated IANA Considerations.
- o Added Retry Interval distinct timer for between failed checks.
- o Added default parameter values.

13. Appendix D. Changes from -03

This section should be removed by the RFC Editor.

- o Reduced default Limit value.
- o Formatting and minor cosmetic changes.

14. References

14.1. Normative References

- [RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <<https://www.rfc-editor.org/info/rfc826>>.

- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<https://www.rfc-editor.org/info/rfc2132>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [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>>.
- [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>>.

14.2. Informative References

- [RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994, <<https://www.rfc-editor.org/info/rfc1661>>.
- [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>>.
- [RFC3203] T'Joens, Y., Hublet, C., and P. De Schrijver, "DHCP reconfigure extension", RFC 3203, DOI 10.17487/RFC3203, December 2001, <<https://www.rfc-editor.org/info/rfc3203>>.
- [RFC4039] Park, S., Kim, P., and B. Volz, "Rapid Commit Option for the Dynamic Host Configuration Protocol version 4 (DHCPv4)", RFC 4039, DOI 10.17487/RFC4039, March 2005, <<https://www.rfc-editor.org/info/rfc4039>>.
- [RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <<https://www.rfc-editor.org/info/rfc6192>>.

- [RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<https://www.rfc-editor.org/info/rfc7084>>.
- [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>>.
- [TR-124] "Functional Requirements for Broadband Residential Gateway Devices", 2006, <<https://www.broadband-forum.org/technical/download/TR-124.pdf>>.
- [TR-146] "Subscriber Sessions", 2013, <<https://www.broadband-forum.org/technical/download/TR-146.pdf>>.

14.3. URIs

- [1] <http://www.iana.org/assignments/dhcpv6-parameters>
- [2] <http://www.iana.org/assignments/bootp-dhcp-parameters>

Authors' Addresses

Richard Patterson
Sky UK

Email: richard.patterson@sky.uk

Mikael Abrahamsson
T-Systems

Email: mikael.abrahamsson@t-systems.se