

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

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

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

## Abstract

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

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

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

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

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

Internet-Draft

fa-inas

October 2021

This Internet-Draft will expire on 28 April 2022.

## Copyright Notice

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

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

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Overview . . . . .	<a href="#">3</a>
<a href="#">1.2.</a>	Disclaimer . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Challenges . . . . .	<a href="#">4</a>
<a href="#">2.1.</a>	High level observations . . . . .	<a href="#">4</a>
<a href="#">2.2.</a>	Internetworking limited domain networks with IP addressing . . . . .	<a href="#">5</a>
<a href="#">2.3.</a>	Shorter addresses . . . . .	<a href="#">9</a>
<a href="#">2.4.</a>	Additional semantics . . . . .	<a href="#">9</a>
<a href="#">2.5.</a>	Programmability . . . . .	<a href="#">10</a>
<a href="#">3.</a>	FA-IINAS: Functional Addressing (FA) for Internetworking with Independent Network Address Spaces (IINAS) . . . . .	<a href="#">10</a>
<a href="#">3.1.</a>	Addressing for unicast . . . . .	<a href="#">11</a>
<a href="#">3.2.</a>	Forwarding . . . . .	<a href="#">12</a>
<a href="#">3.2.1.</a>	Dispose Function . . . . .	<a href="#">12</a>
<a href="#">3.2.2.</a>	Steering Function . . . . .	<a href="#">12</a>
<a href="#">3.2.3.</a>	Multiple semantics . . . . .	<a href="#">12</a>
<a href="#">3.2.4.</a>	Internetworking Function . . . . .	<a href="#">14</a>
<a href="#">3.3.</a>	Control Plane . . . . .	<a href="#">16</a>
<a href="#">3.3.1.</a>	Unicast routing . . . . .	<a href="#">16</a>
<a href="#">3.3.2.</a>	Naming . . . . .	<a href="#">17</a>
<a href="#">3.3.3.</a>	Routing . . . . .	<a href="#">18</a>
<a href="#">3.3.4.</a>	Routing policies . . . . .	<a href="#">19</a>
<a href="#">3.4.</a>	Hardware considerations . . . . .	<a href="#">20</a>
<a href="#">3.4.1.</a>	Forwarding plane simplicity . . . . .	<a href="#">20</a>

<a href="#">3.4.2.</a>	Optimizing for smaller networks . . . . .	<a href="#">21</a>
<a href="#">3.4.3.</a>	Maximum address sizes . . . . .	<a href="#">21</a>
<a href="#">3.5.</a>	Example packet header encoding . . . . .	<a href="#">21</a>
<a href="#">4.</a>	Inspirations . . . . .	<a href="#">22</a>
<a href="#">4.1.</a>	E.164 . . . . .	<a href="#">23</a>

Internet-Draft

fa-inas

October 2021

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

## [1.](#) Introduction

### [1.1.](#) Overview

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

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

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

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

This memo calls the addressing model functional, because addresses

are constructed as a structure of  
func1{parameter(s),func2{parameter(s),...i.funcN{parameter(s)}}}.

## [1.2.](#) Disclaimer

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

## [2.](#) Challenges

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

### [2.1.](#) High level observations

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

- \* Fixed size address space: IPv4/IPv6 address space is fixed length, not allowing to adopt address length to shorter or longer demands. While it is possible to add more addressing via extension headers, there is no option to not send, or shorten the IPv4/IPv6 base header addresses, when they are not required. While the reasons for fixed size addressing in IPv4/IPv6 can be understood for the feasible high-speed, low-cost forwarders of the 1900th, when IPv6 was conceived, these reasons are today (in the opinion of the author) as obsolete as ATM cells where by the end of the 1990th when both hardware forwarding and mathematical models allowed to provide all ATM type QoS with variable sized packets.
- \* The Internet as the primary, if not only use-case driving the design: The address space semantics provided especially by IPv6 is very much focused on the one use-case that drove the development of IPv6: The Internet. While it was and will continue to be the core and sufficient reason for maintaining IPv6, it is not

sufficient in the opinion of the author for the much broader use of IPv6. As of today, a likely overwhelming number of hosts using TCP/IP(v6) protocol stacks are not "on the Internet" and the majority likely is not even "connected to the Internet", but instead, they are part of limited domains. This even includes many routers in large service providers that are used to service Internet traffic. Routers in these networks are only in networks that may be called an "underlay" limited domain networks using MPLS, SR-MPLS or SRv6 and Internet traffic is tunneled across them. When the network design is secure, those routers are neither "on" the internet nor "connect to" the Internet.

- \* Transparent end-to-end addressing at the core of the IP/IPv6 protocol design, but an ever more diverse reality breaking that design for good reasons: The current core principle of IPv4 and IPv6 is that forwarders have to be passing network layer (IPv4/IPv6) addresses transparently and are not allowed to touch/modifying them. This is the core behavior to support primarily the Internet use case. Yet, the IPv4 Internet today would not

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

## 2.2. Internetworking limited domain networks with IP addressing

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

One reference example is that of networking inside a particular product/solution/installation, and then compositing this product with



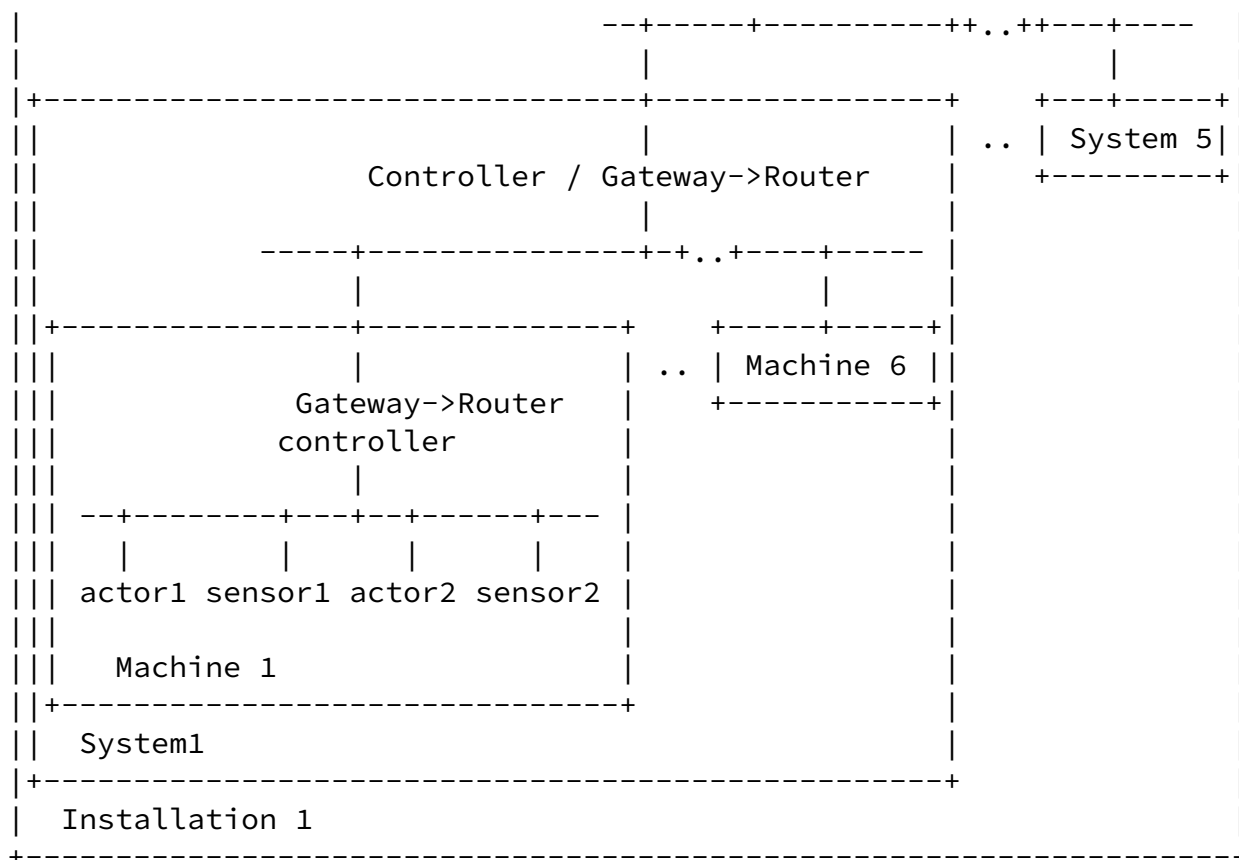


Figure 1: Example hierarchical composed internetwork

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

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

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

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

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

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

In result, ULA is not very persuasive for many such deployments,



especially when the alternative with IPv4 is address prefix mapping as required for NAT, when NAT an an almost free provisioning side effect of setting up the required connectivity via permit lists via network/transport filters. The need to automate such in-network filtering to secure such deployments can also be seen in the advent of MUD, [[RFC8520](#)].

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

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

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

### [2.3.](#) Shorter addresses

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

### [2.4.](#) Additional semantics

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

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

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

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

Internet-Draft

fa-inas

October 2021

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

### [2.5.](#) Programmability

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

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

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

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

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

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

optionally to show where in an address one semantic part ends and another starts.

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

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

### [3.1](#). Addressing for unicast

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

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

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

extensibility.

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

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

## [3.2.](#) Forwarding

### [3.2.1.](#) Dispose Function

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

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

### [3.2.2.](#) Steering Function

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

receives it.

### [3.2.3.](#) Multiple semantics

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

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

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

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

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

Figure 2

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

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

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

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

Figure 3

In other words, because IP multicast uses addresses 224.0.0.0/4, its non-overlapping semantic prefix is E, and IPv4 unicast addresses use the non-overlapping prefixes 0...D. Assume further that a node in

the network had assigned prefix 10.0.0.0/24, then this would translate in our scheme into:

0.A0000.XX

Figure 4

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

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

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

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

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

#### [3.2.4.](#) Internetworking Function

.....

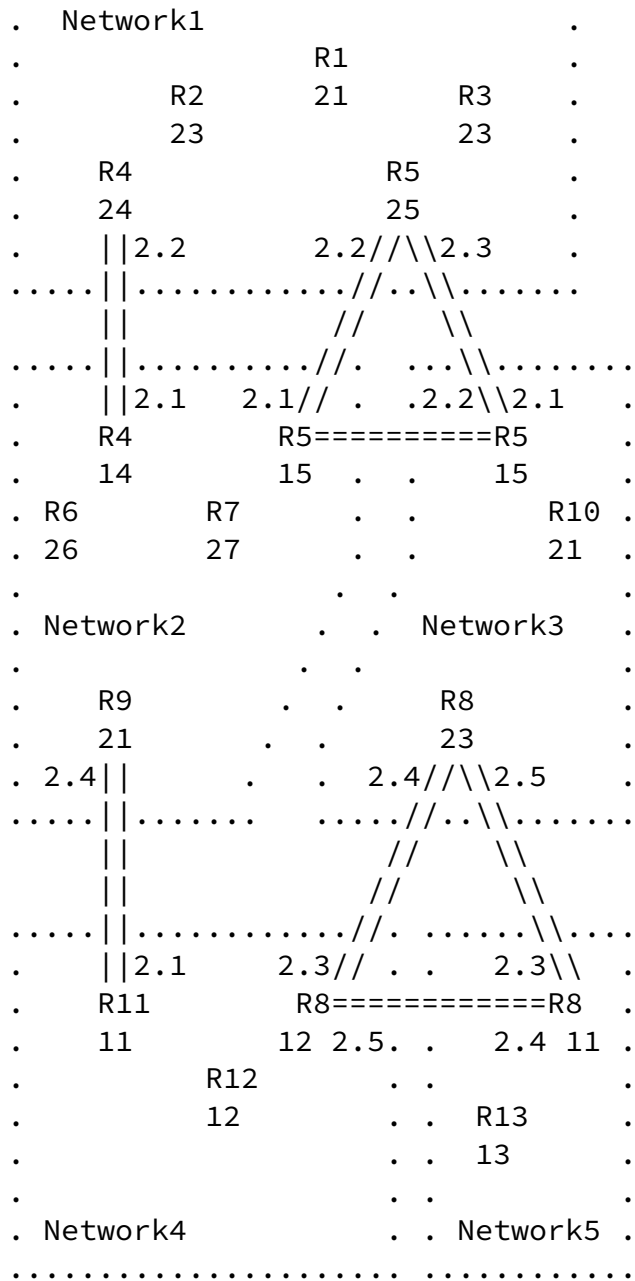


Figure 5: Internetworking example

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



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

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

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

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

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

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

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

### [3.3.](#) Control Plane

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

#### [3.3.1.](#) Unicast routing

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

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

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

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

Internet-Draft

fa-inas

October 2021

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

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

### [3.3.2.](#) Naming

#### [3.3.2.1.](#) Intra network naming

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

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

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

#### [3.3.2.2.](#) Simple inter network naming

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

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

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

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

### [3.3.3.](#) Routing

#### [3.3.3.1.](#) With internetwork topology knowledge

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

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

#### [3.3.3.2.](#) With internetwork naming knowledge

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

Every network (name space) is assigned a globally unique identifier.

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

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

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

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

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

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

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

#### [3.3.4.](#) Routing policies

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

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

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

### [3.4.](#) Hardware considerations

#### [3.4.1.](#) Forwarding plane simplicity

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

The steering function should have a lookup complexity whose complexity is in the order of SR-MPLS or even simpler. It can

constitute of a prefix lookup in the same forwarding table as non-steered forwarding, but the adjacency would then have to strip the looked up prefix from the destination address (comparable to MPLS label pop) and forward the packet again based on the remainder of the destination address - unless additional on-node service functions have to be invoked.

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

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

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

#### [3.4.2.](#) Optimizing for smaller networks

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

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

2<sup>16</sup> nodes".

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

### [3.4.3.](#) Maximum address sizes

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

### [3.5.](#) Example packet header encoding

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

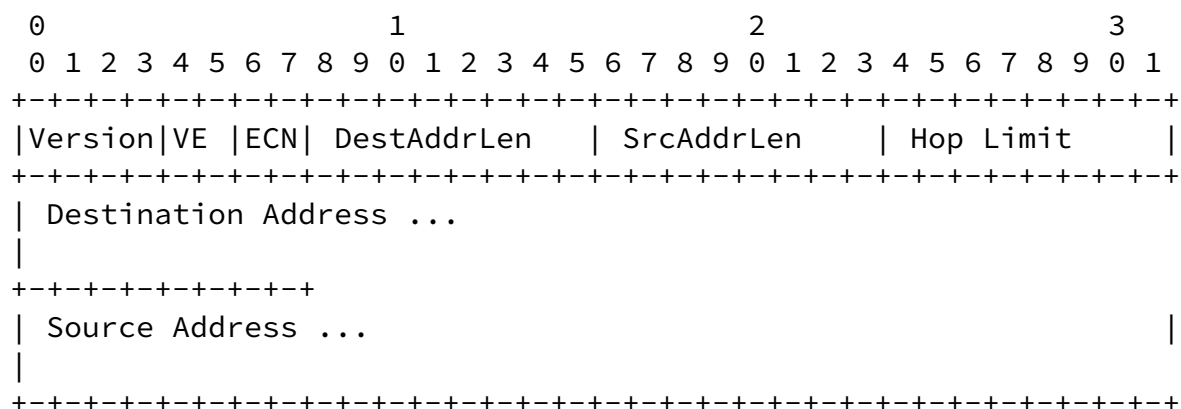


Figure 6: Example packet header encoding

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

VE: Version Extension. 00. Reserved for future variations of the header, such as new extension header formats if desired, so as to not



use up any more than one Version code point.

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

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

**ECN:** See [[RFC3168](#)] and the documents updating it.

**Rsv:** Reserved.

**Hop Limit:** As in IPv6

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

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

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

#### [4.](#) Inspirations

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

#### 4.1. E.164

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

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

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

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

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

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

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

Internet-Draft

fa-inas

October 2021

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

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

Soon:

```

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

```

Today:

```

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

```

Figure 7: Example internetwork for E.164 style address structure

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

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

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

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

#### [4.2.](#) MPLS

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

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

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

#### [4.3.](#) Segment Routing SR-MPLS / SRv6

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

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

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

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

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

#### [4.4.](#) Research

[Haoyu] proposes a hierarchical addressing scheme and provides

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

## 5. Summary and conclusions

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

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

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

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

## 6. Changelog

00: Initial version

01: Refresh, new co-author

## 7. Informative References

- [CiscoNAT] Akkiraju, P., Delgadillo, K., and Y. Rekhter, "Enabling Enterprise Multihoming with Cisco IOS Network Address Translation (NAT)", 2000, <[http://staff.ustc.edu.cn/~james/cisco/nat/emios\\_wp.htm](http://staff.ustc.edu.cn/~james/cisco/nat/emios_wp.htm)>.
- [Haoyu] Song, H., Zhang, Z., Qu, Y., and J. Guichard, "Adaptive Addresses for Next Generation IP Protocol in Hierarchical Networks", IEEE 2020 IEEE 28th International Conference on Network Protocols (ICNP), n.d..
- [I-D.[draft-farrel-soon](#)] Farrel, A., "A Definition of the Term "Soon" for Use in Discussions with Working Group Chairs and Area Directors", Work in Progress, Internet-Draft, [draft-farrel-soon-07](#), 8 March 2021, <<https://www.ietf.org/archive/id/draft-farrel-soon-07.txt>>.
- [I-D.jia-flex-ip-address-structure] Jia, Y., Chen, Z., and S. Jiang, "Flexible IP: An Adaptable IP Address Structure", Work in Progress, Internet-Draft, [draft-jia-flex-ip-address-structure-00](#), 31 October 2020, <<https://www.ietf.org/archive/id/draft-jia-flex-ip-address-structure-00.txt>>.

- [I-D.jia-intarea-scenarios-problems-addressing] Jia, Y., Trossen, D., Iannone, L., Shenoy, N., Mendes, P., 3rd, D. E. E., and P. Liu, "Challenging Scenarios and Problems in Internet Addressing", Work in Progress, Internet-Draft, [draft-jia-intarea-scenarios-problems-addressing-02](#), 23 October 2021, <<https://www.ietf.org/archive/id/draft-jia-intarea-scenarios-problems-addressing-02.txt>>.
- [I-D.king-irtf-challenges-in-routing] King, D. and A. Farrel, "Challenges for the Internet Routing Infrastructure Introduced by Changes in Address Semantics", Work in Progress, Internet-Draft, [draft-king-irtf-challenges-in-routing-03](#), 14 June 2021,

<<https://www.ietf.org/archive/id/draft-king-irtf-challenges-in-routing-03.txt>>.

[I-D.king-irtf-semantic-routing-survey]

King, D. and A. Farrel, "A Survey of Semantic Internet Routing Techniques", Work in Progress, Internet-Draft, [draft-king-irtf-semantic-routing-survey-02](https://www.ietf.org/archive/id/draft-king-irtf-semantic-routing-survey-02), 28 June 2021, <<https://www.ietf.org/archive/id/draft-king-irtf-semantic-routing-survey-02.txt>>.

[RFC1621] Francis, P., "Pip Near-term Architecture", [RFC 1621](#), DOI 10.17487/RFC1621, May 1994, <<https://www.rfc-editor.org/info/rfc1621>>.

[RFC1622] Francis, P., "Pip Header Processing", [RFC 1622](#), DOI 10.17487/RFC1622, May 1994, <<https://www.rfc-editor.org/info/rfc1622>>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", [BCP 5](#), [RFC 1918](#), DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", [RFC 3031](#), DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", [RFC 3168](#), DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", [RFC 3306](#), DOI 10.17487/RFC3306, August 2002, <<https://www.rfc-editor.org/info/rfc3306>>.

[RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", [RFC 3956](#), DOI 10.17487/RFC3956, November 2004,



<<https://www.rfc-editor.org/info/rfc3956>>.

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

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

#### Authors' Addresses

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

Email: [tte@cs.fau.de](mailto:tte@cs.fau.de)

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

Email: [nxsvks@rit.edu](mailto:nxsvks@rit.edu)

