Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 14 April 2022

H. Song
Futurewei Technologies

11 October 2021

Short Hierarchical IP Addresses at Edge Networks
draft-song-ship-edge-02

Abstract

To mitigate the IPv6 header overhead in edge networks, this draft proposes to use short hierarchical addresses excluding the network prefix within edge networks. An edge network can be further organized into a hierarchical architecture containing one or more levels of networks. The border routers for each hierarchical level are responsible for address augmenting and pruning when a packet leaves or enter a lower level network. Specifically, the top-level border routers convert the internal IP header to and from the standard IPv6 header. This draft presents an incrementally deployable scheme allowing packet header to be effectively compressed in edge networks without affecting the network interoperability.

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.

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 14 April 2022.

Table of Contents

1. Introduction

Internet of Things (IoT) and 5G introduce to the Internet a huge number of addressable entities (e.g., sensors, machines, vehicles, and robots). The transition to IPv6 is inevitable. While the 128-bit address of IPv6 was considered large enough and future-proof, the long IP addresses inflate the packet header size. 80% of a basic IPv6 header is consumed by addresses.

In IoT networks, thing-to-thing communication through wireless connections is dominant, which presents several distinct characteristics. (1) The communication pattern is often frequent short-message exchanges (e.g., industry robots and networked vehicles). (2) The communication is usually energy sensitive (e.g.,

battery-powered sensors). (3) The communication often requires low latency (e.g., industry control). (4) The precious wireless channels demand high bandwidth utilization (e.g., ZigBee, Bluetooth, Wi-Fi, and 5G). These characteristics render a large header overhead unfavorable and even prohibitive.

The address overhead also takes its toll on Data Center Networks (DCN), especially when large scale containers are deployed, the east-west traffic is dominant, and the prevailing communications are comprised of short messages (e.g., key-value pairs) and conducted through virtual switches.

In IoT and DCN, since most communications happen between adjacent and related entities, it is a good practice to locally confine communication, computing, and storage due to performance, efficiency, and security considerations, as advocated by Edge Computing. Such a communication pattern provides an opportunity to mitigate the IPv6 header overhead problem due to the long addresses.

When an IPv6 address block is allocated to an edge network, all the entities in the edge network share the same address prefix. When these entities communicate with each other, they can ignore the common prefix. In fact, they don not even need to know the common prefix. Only when they need to communicate with entities outside of the edge network, the full addresses are needed. Even in this case, the entities in the edge network still do not need to know the prefix. It is sufficient for the gateway routers at the network border to manipulate the addresses (i.e., augmenting or pruning the address) to meet the addressing requirement.

Following this line of thought, an edge network can be further partitioned into multiple hierarchical levels, which support flexible sub-networking. The result is that an end entity needs to maintain an even shorter address as its identifier. For communication crossing network levels, the address manipulation is done at each gateway router on the path recursively.

2. Short Hierarchical Address in Edge Networks

2.1. Edge Network Hierarchy

In this draft, we define an edge network as a stub network which does not support traffic transit service. The stub network is assigned an IPv6 address block. In this sense, a data center network in cloud can also be considered as an edge network. An edge network usually falls under a single network administration domain.

The address block assigned to an edge network is identified by a prefix P with the length of L < 128 bits. The remaining S = 128-L bits can be used to assign addresses to the entities in this network. A key observation is: the entities in this network do not need to be aware of P's length and value at all. We can further partition the edge network into multiple hierarchical levels, making a tree architecture. The root represents the entire edge network. Each other node represents a lower level network occupying a sub address space owned by its parent node. A leaf node represents a lowest level network. We name the root level network the L_0 network. Its children are all L_1 networks, and so on so forth. In other words, the network level is the depth of the corresponding node in the tree.

The network hierarchy partitions the S-bit address into multiple sections. Assume an entity is in an L_n network. The S-bit address is partitioned into n+1 sections. The entity only needs to keep the last section of the S-bit address as its ID. The gateway routers for each level of network maintain one section of the S-bit address. Specifically, the gateway routers of L_i (i>0) keep the i-th section of the S-bit address, and the gateway routers of L_0 keep the assigned IPv6 address block prefix P.

Figure 1 shows an edge network example, in which are three network levels. The edge network A is assigned a 96-bit IPv6 address prefix (2001:0db8:ac10:fe01::0001), which means it owns a 32-bit address space. In this space, two L_1 networks are created: B with a 16-bit prefix (0xaaaa) and C with a 24-bit prefix (0xcccccc). Note that the prefixes at the same level must not overlap in order to guarantee entities in the edge network are uniquely addressable. Network B contains two entities x and y, and Network C contains one entity z. In network B, an L_2 network C is further created with a 8-bit prefix (0xbb). In this example, an entity in C or D (e.g., m and z) only need to own a 8-bit address, an entity in B but not in D (e.g., x and y) needs to own a 16-bit address, and an entity in A but not in B and C needs to own a 32-bit address. In this way, each entity in A still logically owns a unique IPv6 address (e.g., the IPv6 address of the entity m in D with ID of 5 is 2001:0db8:ac10:fe01::0001:aaaa:bb05), although the entity m is only aware of its local ID (0x05).

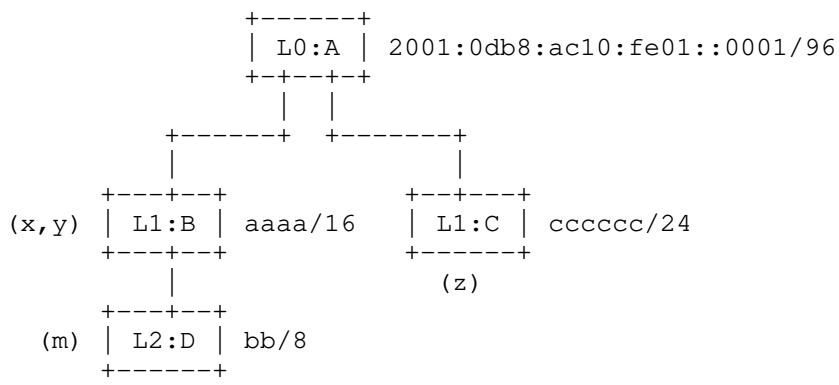

Figure 1: A Hierarchical Edge Network Example

2.2. Address Fields

The edge networks adopting the short and variable size address scheme need a new type of IP header, which is referred as IPvn in this draft. Apart from the IP version, the major difference between IPvn and IPv6 headers is the address fields. IPvn replaces IPv6's 128-bit source address field and 128-bit destination address field with the four fields shown in Figure 2.

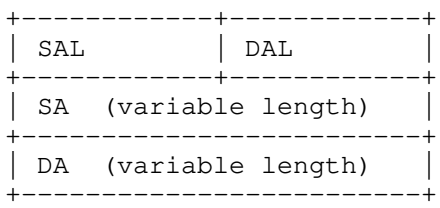

Figure 2: IPvn Address Fields

The Source Address Length (SAL) and the Destination Address Length (DAL) fields have fixed length, while the Source Address (SA) and the Destination Address (DA) fields are of variable length. To simplify the implementation, SA and DA are preferred to be byte-aligned. It is possible to define the length of address in the unit of byte, nibble, or bit. Each has its own pros and cons. The unit of byte can help reduce the size of the SAL/DAL but results in coarse network granularity which might be inefficient in address allocation. For example, a 3-bit SAL/DAL is enough to encode 8 possible address lengths (one to eight bytes) for networks. In this design, each higher level network's address space expands 256 times. On the other

extreme, the unit of bit allows fine network granularity but requires more space for SAL/DAL. For example, 6-bit SAL and DAL can support an address length up to 64 bits (8 bytes) and each higher level network is only twice larger.

With a few bits, it is also possible to design a more sophisticated encoding scheme that supports variable address length steps and adapts to the ideal network sizes at different levels.

Assuming SA and DA are 2 bytes each, and SAL and DAL are 4 bits each, the address fields are only 5 bytes in total. Comparing to IPv6, the size of the address fields is reduced by 84%.

2.3. Router Roles and Function

In the edge network hierarchy, each network has one or more Level Gateway Routers (LGR) which are responsible for forwarding packets in or out of this network. The LGRs are the only interface between a network and its parent network.

A network can be in a single L2 domain, which means all the entities in this network (excluding those in its child networks) and all the network devices (including the LGRs to the parent network and the child networks) are L2 reachable. A network can also be a pure L3 network in which no L2 device is allowed. Each entity in a network is directly connected to either an LGR or some internal routers named Intra-Level Router (ILR) which is solely responsible for packet forwarding within the network. In this case, the entities need to partially participate in the routing process (e.g., advertising its address).

The scale of an intra-level network can be used to guide the L2/L3 selection. Small networks prefer the L2-based solution and large networks prefer the L3-based solution. In the higher level networks (e.g., closer to the top level network or the tree root), since the number of entities is usually small, it is free to choose between L2 or L3-based solution. The leaf level networks are usually L2-based for simplicity.

Unlike in IPv4 and IPv6 networks, the address related fields in IPvn header can be modified by LGRs. An LGR of a network keeps a prefix that can augment the SAs from this network to an address outside of this network. If an LGR needs to forward an internal packet outside (i.e., DAL > SAL), it augments the packet's SA and updates its SAL accordingly. Reversely, if an LGR receives a packet from the parent network destined for the child network for which it serves as a gateway (i.e., the parent network prefix matches the DA's prefix), it strips off the parent network prefix from the packet's DA and updates its DAL accordingly.

In contrast, within an L3-based level network, ILRs do not modify the address fields. An ILR can decide the packet forwarding direction by examining the DAL. If DAL > SAL, the packet needs to be forwarded to an LGR of this network; otherwise, the packet needs to be forwarded within the current network, and possibly into a lower-level child network.

An LGR of the top-level network (i.e., the L0 network) is special. In addition to the address manipulation, it is also responsible for converting the IPvn header to and from the standard IPv6 header to support the Internet interoperability. We name such a router IP Translator (IPT).

We use the edge network shown in Figure 1 to illustrate some packet forwarding examples. The details for the involved entities are summarized in Figure 3. In the IPvn packet header, we use 4 bits to encode the address length. In particular, 0b0000 is used to indicate the address is 16 bytes long (i.e., a complete IPv6 address).

Entity	ID	Length	Level	Network	Prefix
x	0x0001	2bytes	1	B	0xaaaa/16
y	0x0002				
z	0x01	1byte	1	C	0xcccccc/24
m	0x08	1byte	2	D	0xbb/8

Figure 3: Entity Address Configuration

The first example in Figure 4 shows how packets are forwarded from x to y within the same network B. In this case, the source address and destination address have the same length. The packets only pass through an ILR which does not change the address fields.

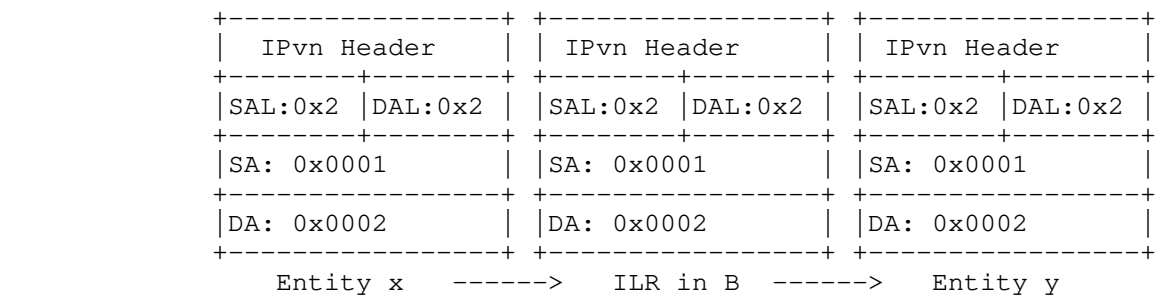

Figure 4: Forward within a network level in the edge

The second example in Figure 5 shows how packets are forwarded from x in B to z in C. At LGR of B, the source address is augmented, and at the LGR of C, the destination address is pruned. Since x and z's nearest common ancestor network is A, so the packets never need to leave network A, so A's prefix is oblivious throughout the communication.

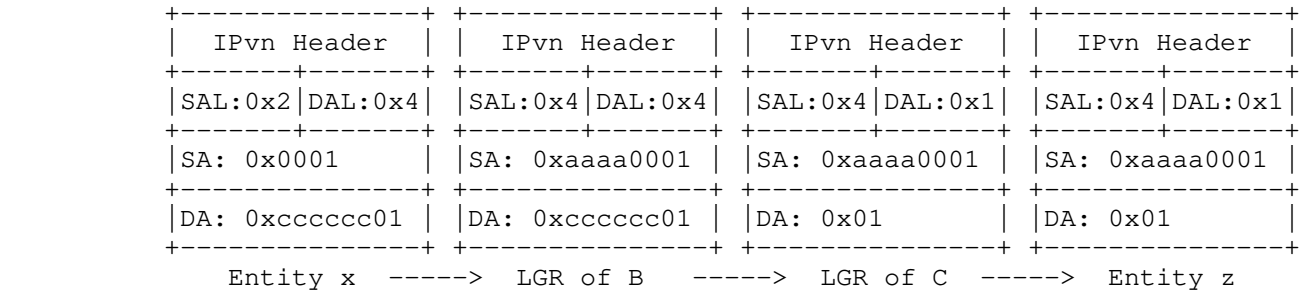

Figure 5: Forward to another network in the edge

The last example in Figure 6 shows how packets are forwarded from x in B to a host in IPv6 domain. In the IPT of A, the IPvn header is converted to an IPv6 header.

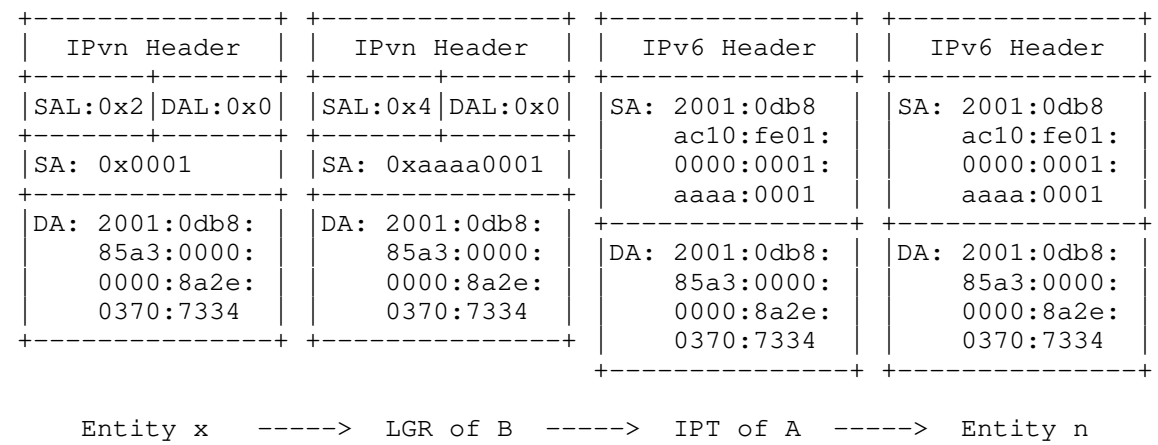

Figure 6: Forward out of the edge network

3. Deployment and Interoperability Consideration

3.1. Control Plane

Within the edge networks where IPvn is applied, all the control plane functions and protocols need to be modified or redesigned due to the hierarchical network architecture of IPvn. Fortunately, the updates are often incremental and the results are usually simpler than their counterparts in IPv4 and IPv6. We briefly discuss a few essential protocols that enable the operation of IPvn.

DHCP: An entity can be manually configured or dynamically acquire its address when booting up. Each network in the edge network hierarchy may contain a Dynamic Host Configuration Protocol (DHCP) server responsible for assigning addresses (i.e., IDs) to the entities in the same network. The protocol is almost identical to that for IPv4 and IPv6, except that the assigned address length is adaptive to the allocated network size.

DNS: For an entity to acquire the address of a peer entity in order to initiate a communication, Domain Name System (DNS) is the prominent approach but with a new service model. Any network in the hierarchy can provide name service. Each entity is configured with the address of the closest DNS server on the path to the root network. The hierarchical network architecture allows a scoped domain name service. That is, a name registered in a network is only valid in this network and its child networks. It is possible that a same name is registered in two networks and one network is the other’s ancestor. Such name conflict is not a bug but a

feature for name reuse, which is transparent to the name query process. In this case, the name resolved from the closer DNS server is used.

Each network may contain a DNS server (the notation is only logical. The actual implementation may follow the same hierarchical and distributed architecture of today's DNS). Each DNS server knows the nearest DNS server in a higher level network and the nearest DNS servers in lower level networks. This essentially organizes the DNS servers in the same tree structure as the hierarchical network. Each named entity in a network is registered with the DNS server that covers its scope, which is basically a subtree.

We have several methods to populate the name to support the scoped name queries, each with different storage and performance trade-off: 1) register the name in all the DNS servers in its scope (i.e., all the subtree nodes); 2) recursively register the name in every parent DNS server until the scope root; and 3) register the name only in the DNS server in its scope root. The address for a name returned by a DNS server is on a "need-to-know" basis. In a network, if the address's prefix matches the query's address prefix, the prefix is removed. This can be easily done by the original or the relay DNS servers. If a query cannot be resolved by the DNS server in the L0 network, the query, after the IP protocol translation is done, exits the IPvn domain and enters into the IPv4/IPv6 domain to a public DNS server. When the response comes back and enters the edge network, the result can be cached by the DNS servers on the path.

ARP: In a L2-based network, the operation of Address Resolution Protocol (ARP) or Neighbor Discovery Protocol (NDP) is almost identical to that for IPv4 and IPv6. In an L2- based network, each immediate entity should be configured with a default gateway address to its parent network. If no default gateway is configured, a network LGR should be configured as an ARP proxy to respond to all internal ARP requests for addresses out of the network. Similarly, the LGRs to any child network of this network are also needed to be configured as ARP proxy to response all ARP requests for addresses in that network. Due to the multi-homing gateway routers, an ARP request may receive multiple responses. It is up to the requester to determine which one to cache.

Routing Protocol: The entire edge network may belong to a single AS, so the interior gateway routing protocols (IGP) such as OSPF and IS-IS can be used. Other child networks in this network can be considered OSPF stub areas or IS-IS levels. A simpler way is that each network run an independent instance of OSPF or IS-IS.

Specially, an LGR at a network border runs two OSPF/IS-IS instances: one for the upper-level network and the other for the lower-level network. The hierarchical architecture solves the routing protocol scalability issue, and simplifies the protocol implementation by removing unnecessary features. The clean routing scope helps to secure the infrastructure and troubleshoot the networks.

3.2. Data Plane

IPvn Socket for End Entities: To enable IPvn as a new network layer protocol in end entities, we need to add the protocol implementation in the OS Kernel and allow applications to invoke the socket API using the address family parameter AF_INETN. The L4-L7 protocol stack and the application logic remains the same, allowing direct communication between entities in IPvn domain and in IPv4/IPv6 domain.

Forwarding Table Lookups in Networks: The short hierarchical address simplifies the router forwarding table structure in L3-based networks. A forwarding table only contains the addresses to local entities and the prefixes to the child networks. Since there is no nested prefixes, the Longest Prefix Matching (LPM) is not necessary. The small number of unique prefix lengths allows the prefixes to be grouped on lengths and each group to be implemented as a hash table. A lookup can search all the hash tables in parallel, and at most one table can result a positive match. This design avoids the use of expensive TCAM or other complex trie-based algorithms.

An LGR between an L_i network and an L_(i+1) network has two types of interfaces: one faces the L_i network and the other faces the L_(i+1) network. One LGR may serve more than one L_(i+1) network. Hence, an LGR may contain multiple logical forwarding tables, with each for a network. For a packet in LGR, once its target network is determined and the address related fields are processed, the proper forwarding table can be searched.

3.3. Using NAT for the edge network

To expand the address space of the edge network, the IPT of the edge network can also support functions similar to NAT. In this case, the edge network is assigned one or more public IPv4/IPv6 addresses. The entities in IPvn domain use private addresses. The IPT maintains the mapping table between the private address and public address.

3.4. Extension Beyond IPv6

Although the motivation of this draft is to support shorter address (i.e., smaller L3 header overhead) in edge networks, it is worth noting that the scheme allows the addresses to be extend to arbitrary length, even longer than 128bits. In that case, the address space of the IPvn network can be greater than that of IPv6 and the entire IPv6 network can be considered an edge network of the IPvn network. This scenario should be considered when specifying the address fields of IPvn.

4. Security Considerations

The addressing scheme and architecture allow a securer edge network. The IPTs and LGRs naturally support the access control.

5. IANA Considerations

The proposal requires to use a new IP version and define a new IP header which can be converted to/from an equivalent IPv6 header.

6. Acknowledgments

We acknowledge the technical contributions, suggestions and comments from Yingzhe Qu, Zhaobo Zhang, James Guichard, Toerless Eckert, Stewart Bryant, and Michael McBride.

7. References

7.1. Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[RFC7775] Ginsberg, L., Litkowski, S., and S. Previdi, "IS-IS Route Preference for Extended IP and IPv6 Reachability", RFC 7775, DOI 10.17487/RFC7775, February 2016, <https://www.rfc-editor.org/info/rfc7775>.

Author's Address

Haoyu Song
Futurewei Technologies
Santa Clara,
United States of America

Email: haoyu.song@futurewei.com