Requirements for IPv6 Routers
draft-ali-ipv6rtr-reqs-00

Abstract

An example.

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 March 24, 2017.

Copyright Notice

Copyright (c) 2016 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.
1. Introduction

This memo defines and discusses requirements for devices that perform forwarding for Internet Protocol version 6 (IPv6). This can include (but is not limited to) the devices described below.

- Devices which are primarily designed to forward traffic between multiple interfaces. These are normally referred to by the Internet community as routers or, in some cases, intermediate systems.

- Devices which are designed to modify packets rather than "just" forwarding them. These are often referred to by the Internet community as middle boxes.
Readers should recognize that while this memo applies to IPv6, devices processing IPv6 packets will often also process IPv4 packets, forward based on MPLS labels, and potentially process many other protocols. This memo will only interact with IPv4, MPLS, and other protocols as they impact the behavior of an IPv6 forwarding device; no attempt is made to specify requirements for protocols other than IPv6.

The reader should, therefore, not count on this document as a "sole source of truth," but rather use this document as a guide.

This document is broken into the following sections: a review of Internet architecture and principles, requirements relating to device management, requirements related to telemetry, requirements related to IPv6 forwarding and addressing, and future considerations. Following these sections, a short conclusion is provided for review.

2. Review of the Internet Architecture

The Internet relies or interacts with a number of basic concepts and considerations. These concepts are not explicitly called out in any specification, nor do they necessarily impact protocol design or packet forwarding directly. This section provides an overview of these concepts and considerations to help the reader understand the larger context of this document.

2.1. Robustness Principle

Every point where multiple protocols, or multiple implementations of the same protocol, interact, is an interaction surface that can threaten the robustness of the overall system. While it may seem the global Internet has achieved a level of stability that makes it immune to such considerations, the reality is every network is a complex system, and is therefore subject to massive non repeatable unanticipated failures. Postel's Robustness Principle countered this problem with a simple statement: "Be conservative in what you do, and liberal in what you accept from others." [RFC7922]

However, since this time, it has been noted that following this law allows errors in protocols to accumulate over time, with overall negative effects on the system as a whole. RFC1918 [RFC1918] describes several points in conjunction with this principle that bear updating based on further experience with large scale protocol and network deployments within the Internet community, including:

- Software should be written to deal with error states gracefully. Software should not degrade in a way that will cause the failure of adjacent systems when possible. For instance, when a routing
protocol implementation fails, it should not do so in a way that
will cause the spreading of or continued existence of false
reachability information, nor should it fail in a way that
overloads adjacent routers or interacting protocols and causing a
cascading failure.

- It is best to assume the network is filled with poor
implementations and malevolent actors, both of which will find
every possible failure mode over time.

- It is best to assume every technology will be used to the limits
of its technical capabilities, rather than assuming a particular
protocol's scope of use will align (in any way) with the intent of
the original designer(s). Successful implementations attract more
functionality, much like a few nodes in a scale free graph
eventually become connectivity hubs.

- Protocols and implementations change over time. A corollary of
the assumption that protocols will be used until they reach their
technical limits rather than staying within a tightly scoped
purpose, is that protocols will change over time as they gain new
functionality. Protocol and implementation design should take
into account use cases that have not yet been thought of by
building flexibility into protocols.

- Obsolete, but legal, protocol features are often ignored or left
unimplemented. It is important to work within the bounds of what
is actually implemented in any given protocol, and to leave corner
cases for another day. It is never helpful to boil the ocean
whether in a design, an implementation, or a protocol.

2.2. Complexity Principle

The simplest and most obvious solution to any problem is often easy
to design, deploy, and manage. It's also often wrong and/or broken.
As much as developers, designers, and operators might like to make
things as simple as possible, hard problems require complex
solutions. This leads to the following observations.

Elegance is the ultimate goal. Rather than seeking out simple
solutions because they are simple, seek out solutions that will solve
the problem in the simplest way possible. Often this will require
seeing the problem from different angles, trying to break the problem
up in multiple ways, and trying, abandoning, and rebuilding ideas and
implementations until a better way is found.

There are always tradeoffs. For any protocol, network, or
operational design decision, there will a tradeoff between at least
two competing goals. If some problem appears to have a single solution with not tradeoffs, this doesn’t mean the tradeoffs don’t exist. Rather, it means the tradeoffs haven’t been discovered yet.

2.3. Layered Structure

The Internet data plane is organized around broad top and bottom layers, and much thinner middle layer. This is illustrated in the figure below.

```
  /          /
 / HTTP, FTP, SNMP, ETC. /
 /   TCP, UDP   /
 /   IPv6     /
 /   (MPLS)  \
 /   \\   \
 /   Ethernet, Wireless \
 /   Physical Media   \
```

Figure 1

This layering emulates or mirrors many naturally occurring systems, and is a common strategy for managing complexity. The single protocol in the center, IPv6, serves to separate the complexity of the lower layers from the complexity of the upper layers. This center layer of the Internet ecosystem has traditionally been called the Network Layer, in reference to the Department of Defense (DoD) [DoD] and OSI models. [OSI] The Internet ecosystem includes three different protocols in this central location.

- IPv4, an older network protocol that, it is anticipated, will be replaced over time as the Internet ecosystem standardizes on IPv6
- IPv6, a newer network protocol that is being adopted
- MPLS, or MultiProtocol Label Switching, which is often used as a data plane within an autonomous system

These protocols are often treated as if they exist in strict hierarchical layers with a well defined and followed Application Programming Interface (API), data models, Remote Procedure Calls (RPCs), sockets, etc. The reality, however, is there are often solid reasons for violating these layers, creating interaction surfaces that are often deeper than intended or understood without some
experience. Beyond this, such layering mechanisms act as information abstractions. It is well known that all such abstractions leak. Because of these intentional and unintentional leakages of information, the interactions between protocols is often subtle.

2.4. Routers

A router connects to two or more logical interfaces and at least one physical interface. A router processes packets by:

- Receiving a packet through an interface
- Stripping the data link and/or physical header off the packet
- Examining the packet for errors, information that must be handled locally, etc.
- Looking up the destination in a local forwarding table
- Rewriting the data link and/or physical layer header
- Transmitting the packet out an interface

When consulting the forwarding table, the router searches for the longest prefix containing the destination address, and uses the information in the table to determine the next hop, or rather the next logically connected device to forward the packet to. The next hop will either be another router, which will presumably carry the packet closer to the final destination, or it will be the destination host itself. The following figure provides a conceptual model of a router; not all routers actually have this set of tables and interactions, and some have many more moving parts. This model is simply used as a common reference to promote understanding.
Figure 2
3. Requirements Related to Device Management

Network engineering began in the era of Command Line Interfaces (CLIs), and has generally stayed with these CLIs even as the Graphical User Interface (GUI) has become the standard way of interacting with almost every other computing device. Direct human interaction with networking devices in large scale and complex environments, however, tends to result in an unacceptably low Mean Time Between Mistakes (MTBM), directly impacting the overall availability of the network. In reaction to this, operators have increased their reliance on automation in deploying and configuring devices. This section considers the various components of device management.

3.1. Configuration

Configuration primarily relates to the startup, candidate, and running configurations in the router model shown above. In order to deploy networks at any scale, operators rely on automated management of network device configuration. This effort has traditional focused on Simple Network Management Protocol (SNMP) Management Information Base (MIBs). In the future, operators expect to move towards open source/open standards YANG models.

Network devices should place a priority on supporting machine readable Application Programming Interfaces (APIs), rather than human interaction, particularly interfaces that understand and accept configuration and other information carried in YANG models.

To support automated network device configuration, IPv6 routers and network devices SHOULD support YANG and SNMP configuration, including (but not limited to):

- Openconfig models [OPENCONF] related to the protocols configured on the device, interface state, and device state
- [RFC7223]: A YANG Data Model for Interface Management
- [RFC7224]: IANA Interface Type YANG Module
- [RFC7277]: A YANG Data Model for IP Management
- [RFC7317]: A YANG Data Model for System Management
- Simple Network Management Protocol (SNMP) MIBs as appropriate
3.2. Device Access

To operate a network at scale, operators rely on the ability to access a device to troubleshoot and gather state manually and programmatically through a number of different interfaces. These interfaces should provide current device configuration, current device state (such as interface state, packets drops, etc.), and current control plane contents (such as the RIB in the figure above). In other words, manual and programmable interfaces should provide information about the network device (the whole device stack).

To support automated state gathering and troubleshooting, routers supporting IPv6 SHOULD support:

- TELNET ([RFC0854]): TELNET SHOULD be disabled by default, but should be available for operational purposes as required or as configured by the operator
- SSH ([RFC4253]): SSH SHOULD be the default access for IPv6 capable network devices
- NETCONF ([RFC6241])
- RESTCONF ([I-D.ietf-netconf-restconf])

3.3. Zero Touch Provisioning

To operate a network at scale, operators rely on protocols and mechanisms that reduce provisioning time to a minimum. The preferred state is zero touch provisioning; plug a new network device in and it just works without any manual configuration. This is likely to be unattainable for some time yet to come, but coming closer to this ideal reduces MTBM and Operational Expenses (OPEX), both important goals in the real world.

To reach this goal, IPv6 network devices should support several standards, including, but not limited to:

- [I-D.ietf-dhc-rfc3315bis]: Dynamic Configuration Protocol for IPv6
- SLAAC ([RFC7217] and [RFC7527]): SLAAC SHOULD be enabled by default on all network device interfaces

(not certain SLAAC should be default, but I put it there for now)
3.4. Device Protection against Denial of Service Attacks

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are unfortunately common in the Internet globally; these types of attacks cost network operators a great deal in opportunity and operational costs in prevention and responses. The network control plane, as well as protocols used to access the device. To provide for effective counters to DoS and DDoS attacks directly on network devices:

- Manufacturers and system integrators should test and clearly report the packet/traffic load handling capabilities of devices with and without various encryption methods enabled
- Network devices should be able to police traffic destined to the control plane based on the rate of traffic received
- Ideally, network devices should be able to police traffic destined to the control plane based on best practice host based firewall techniques

4. Requirements Related to Telemetry

Telemetry relates to information devices push to systems used to monitor and track the state of the network. This applies to individual devices as well as the network as a system. There are three broad categories of telemetry: device state, topology state, and flow state. These three roughly correspond to the management plane, the control plane, and the forwarding plane of the network. Each of the sections below considers one of these three telemetry types.

4.1. Device Status and Error Logging

IPv6 network devices should be able to report errors and status changes in a number of different formats. The supported formats should focus on machine readability, rather than human readability. Specifically, IPv6 network devices SHOULD support:

- NETCONF/RESTCONF transporting telemetry formatted according to YANG (see above)
- Syslog ([RFC5424])
- SNMP as appropriate

NETCONF/RESTCONF transporting telemetry formatted according to YANG should be the current focus of development for vendor provided and
open source software; ideally, the entire network could be monitored using a single modeling language to ease implementation of telemetry systems and increase the pace at which new software can be deployed in production environments.

4.2. Network Topology and Traceability

IPv6 network devices are part of a system of devices that, combined, make up the entire network. Viewing the network as a system is often crucial for operational purposes, such as understanding flows, tracing changes in the topology, utilization, and other factors over time. To support systemic monitoring of the network topology, IPv6 devices SHOULD support at least:

- [RFC5424]: North-Bound Distribution of Link-State and Traffic Engineer (TE) Information using BGP
- [I-D.ietf-i2rs-yang-l2-network-topology]: An I2RS model for layer 2 topologies
- [I-D.ietf-i2rs-yang-l3-topology]: An I2RS model for layer 3 topologies
- [I-D.ietf-i2rs-yang-network-topo]: A Data Model for Network Topologies

(More to be added)

4.3. Traffic Flow and Traceability

(To be added)

5. Requirements Related to IPv6 Forwarding and Addressing

There are a number of capabilities that a device should have to be deployed into an IPv6 network, and several forwarding plane considerations operators and vendors need to bear in mind. The sections below explain these considerations.

5.1. The IPv6 Address is not a Host Identifier

The IPv6 address is commonly treated as a host identifier; it is not. Rather, it is an interface identifier that describes the topological point where a particular host connects to the Internet. It is generally harmful to embed IPv6 addresses inside upper layer headers to identify a particular host.
5.2. Router Handling of IPv6 Addresses

Internet Routing Registries may allocate a network operator a wide range of prefix lengths (see RFC6177 [RFC6177] for further information). Within this allocation, network operators will often suballocate address space along nibble boundaries (/48, /52, /56, /60, and /64) for ease of configuration and management. Several common practices are:

- Each multiaccess interface is allocated a /64
- Point-to-point links are allocated a /64, but may be addressed with a longer prefix length to prevent certain kinds of denial of service attacks
- Although aggregation may only be performed to the nibble boundaries noted above, variances are possible
- Loopback addresses are assigned a /128

Given these common practices, routers designed to run IPv6 SHOULD support the following addressing conventions:

- The default prefix length on any interface other than a loopback SHOULD be a /64
- Configuring a prefix length longer than a /64 on any interface not configured as a point-to-point should require additional configuration steps to prevent manual configuration errors
- Network devices SHOULD NOT assume IPv6 prefix lengths only on nibble boundaries, but rather should support any prefix length shorter than the /64, /128, and longer prefixes used for point-to-point interfaces
- Loopback interfaces SHOULD default to a /128 prefix length unless some additional configuration is undertaken to override this default setting

5.3. Maximum Transmission Unit and Jumbo Frames

The long history of the Maximum Transmission Unit (MTU) in networks is not a happy one. Specific problems with MTU sizing include:

- Many different default sizes on different media types, from very small (576 bytes on X.25) to very large (17914 bytes on 16Mbps Token Ring)
Many different ways to calculate the MTU on any given link; for instance a 9000 byte MTU can be calculated as 8184 bytes on one operating system, 8972 on another, and 9000 on a third.

The increasing use of tunnel encapsulations in the network; for instance MPLS over GRE over IP over...

The wide variety of default MTUs across many different end hosts and operating systems.

The general ineffectiveness of path MTU discovery to operate correctly in the face of packet filters and rate limiters.

Lower speed links at the network edge which require a lot of time to serialize a packet with a large MTU.

Increased jitter caused by increasing the disparity between large and small packet size across a lower bandwidth links.

The final point requires some further elucidation. The time required to serialize various packets at various speeds are:

- 64 byte packet onto a 10Mb/s link: .5ms
- 1500 byte packet onto a 10Mb/s link: 1.2ms
- 9000 byte packet onto a 10Mb/s link: 7.2ms
- 64 byte packet onto a 100Mb/s link: .05ms
- 1500 byte packet onto a 100Mb/s link: .12ms
- 9000 byte packet onto a 100Mb/s link: .72ms

A 64 byte packet trapped behind a single 1500 byte packet on a 10Mb/s link suffers 1.7ms of serialization delay. Each additional 1500 byte packet added to the queue in front of the 64 byte packet adds an additional 1.2ms of delay. In contrast, a 64 byte packet trapped behind a single 9000 byte packet on a 10Mb/s link suffers 7.7ms of serialization delay. Each additional 9000 byte packet added to the queue adds an additional 7.2ms of serialization delay. The practical result is that larger MTU sizes on lower speed links can add a significant amount of delay and jitter into a flow. On the other hand, increasing the MTU on higher speed links appears to add negligible additional delay and jitter.

The result is that it costs less in terms of overall systemic performance to use higher MTUs on higher speed links than on lower.
speed links. Based on this, increasing the MTU across any particular link may not increase overall end-to-end performance, but can greatly enhance the performance of local applications (such as a local BGP peering session, or a large/long standing elephant flow used to transfer data across a local fabric), while also providing room for tunnel encapsulations to be added with less impact on lower MTU end systems.

The general rule of thumb is to assume the largest size MTU should be used on higher speed transit only links in order to support a wide array of available link sizes, default MTUs, and tunnel encapsulations. Routers designed for a network or data center core SHOULD support at least 9000 byte MTUs on all interfaces. MTU detection mechanisms, such as IS-IS hello padding, [RFC7922] SHOULD be enabled to ensure correct point-to-point MTU configuration.

5.4. ICMP Considerations

Internet Control Message Protocol (ICMP) is described in [RFC0792] and [RFC4443]. ICMP is often used to perform a traceroute through a network (normally by using a TTL expired ICMP message), for Path MTU discovery, and, in IPv6, for autoconfiguration and neighbor discovery. ICMP is often blocked by middle boxes of various kinds and/or ICMP filters configured on the ingress edge of a provider network. Routers implementing IPv6 SHOULD:

- NOT filter ICMP by default, as this has negative impacts on many aspects of IPv6 operation, particularly path MTU
- Rate limit the generation of ICMP messages relative to the ability of the device to generate packets and to block the use of ICMP packets being used as part of a distributed denial of service attack
- Implement the filtering suggestions in [I-D.gont-opsec-icmp-ingress-filtering]

5.5. Machine Access to the Forwarding Table

In order to support treating the "network as a whole" as a single programmable system, it is important for each network device have the ability to directly program forwarding information. This programmatic interface allows controllers, which are programmed to support specific business logic and applications, to modify and filter traffic flows without interfering with the distributed control plane. While there are several programmatic interfaces available, this document suggests that the I2RS interface to the RIB be
supported in all IPv6 network devices. Specifically, these drafts should be supported to enable network programmability:

- [I-D.ietf-i2rs-fb-rib-data-model]: Filter-Based RIB Data Model
- [I-D.ietf-i2rs-fb-rib-info-model]: Filter-Based RIB Information Model
- [I-D.ietf-i2rs-rib-data-model]: A YANG Data Model for Routing Information Base (RIB)
- [RFC7922]: I2RS Traceability

5.6. Processing IPv6 Extension Headers
(To be added)

5.7. IPv6 Only Forwarding
(To be added)

6. Future Considerations
(To be added)

6.1. Segment Routing
(To be added)

7. Security Considerations
(To be added)

8. Conclusion
(To be added)

9. References

9.1. Normative References

9.2. Informative References


[I-D.gont-opsec-icmp-ingress-filtering]  
Gont, F., Hunter, R., Massar, J., and S. LIU, "Defeating Attacks which employ Forged ICMPv6 Error Messages", 
draft-gont-opsec-icmp-ingress-filtering-02 (work in progress), March 2016.

[I-D.ietf-dhc-rfc3315bis]  
Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., 
Richardson, M., Jiang, S., Lemon, T., and T. Winters, 
"Dynamic Host Configuration Protocol for IPv6 (DHCPv6) bis", 
draft-ietf-dhc-rfc3315bis-05 (work in progress), June 2016.

[I-D.ietf-i2rs-fb-rib-data-model]  
Hares, S., Kini, S., Dunbar, L., Krishnan, R., Bogdanovic, D., and R. White, "Filter-Based RIB Data Model", 
draft-ietf-i2rs-fb-rib-data-model-00 (work in progress), June 2016.

[I-D.ietf-i2rs-fb-rib-info-model]  
Kini, S., Hares, S., Dunbar, L., Ghanwani, A., Krishnan, R., Bogdanovic, D., and R. White, "Filter-Based RIB Information Model", 
draft-ietf-i2rs-fb-rib-info-model-00 (work in progress), June 2016.

[I-D.ietf-i2rs-rib-data-model]  
Wang, L., Ananthakrishnan, H., Chen, M., 
amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", 

[I-D.ietf-i2rs-rib-data-model]  
Dong, J. and X. Wei, "A YANG Data Model for Layer-2 Network Topologies", 
draft-ietf-i2rs-yang-12-network-topology-03 (work in progress), July 2016.

[I-D.ietf-i2rs-yang-13-topology]  
Clemm, A., Medved, J., Varga, R., Tkacik, T., Liu, X., 
Bryskin, I., Guo, A., Ananthakrishnan, H., Bahadur, N., and V. Beeram, "A YANG Data Model for Layer 3 Topologies", 
draft-ietf-i2rs-yang-13-topology-04 (work in progress), September 2016.


Authors’ Addresses

Zaid Ali Kahn
LinkedIn
xxx
xxx, CA  xxx
USA

Email: zaid@linkedin.com

John Brzozowski
Comcast
xxx
xxx, xxx  xxx
USA

Email: John_Brzozowski@comcast.com

Russ White
LinkedIn
144 Warm Wood Lane
Apex, NC  27539
USA

Email: russ@riw.us