Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 22, 2017

Z. Kahn
LinkedIn
J. Brzozowski
Comcast
R. White
LinkedIn

February 18, 2017

Requirements for IPv6 Routers
draft-ali-ipv6rtr-reqs-02

Abstract

The Internet is not one network, but rather a collection of networks. The interconnected nature of these networks, and the nature of the interconneted systems that make up these networks, is often more fragile than it appears. Perhaps "robust but fragile" is an overstatement, but the actions of each vendor, implementor, and operator in such an interconneted environment can have a major impact on the stability of the overall Internet (as a system). The widespread adoption of IPv6 could, particularly, disrupt network operations, in a way that impacts the entire system.

This time of transition is an opportune time to take stock of lessons learned through the operation of large scale networks on IPv4, and consider how to apply these lessons to IPv6. This document provides an overview of the design and architectural decisions that attend IPv6 deployment, and a set of IPv6 requirements for routers, switches, and middleboxes deployed in IPv6 networks. The hope of the editors and contributors is to provide the neccessary background to guide equipment manufacturers, protocol implemenetors, and network operators in effective IPv6 deployment.

This Internet-Draft will expire on August 22, 2017.

Table of Contents

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 middleboxes. See [RFC7663] for a fuller definition of middleboxes.

Readers should recognize that while this memo applies to IPv6, routers and middleboxes IPv6 packets will often also process IPv4 packets, forward based on MPLS labels, and potentially process many other protocols. This memo will only discuss 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.

For IPv4 router requirements, readers are referred to [RFC1812]. For simplicity, the term "devices" is used interchangeably with the phrase "routers and middleboxes" and the term "routers" throughout this document. These three terms represent stylistic differences, rather than substantive differences.

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.

1.1. Contributors

Shawn Zandi, Pete Lumbis, Fred Baker, and Lee Howard contributed significant text and ideas to this draft.

1.2. Acknowledgements

The editors and contributors would like to thank....

2. Review of the Internet Architecture

The Internet relies on 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 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, explicated in [RFC7922]: "Be conservative in what you do, and liberal in what you accept from others."

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] 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:

- Applications should deal with error states gracefully; an application 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). [RFC5218] defines a wildly sucessful protocol as one that "far exceeds its original goals, in terms of purpose (being used in scenarios far beyond the initial design), in terms of scale (being deployed on a scale much greater than originally envisaged), or both." 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 is that protocols will change over time as they gain new functionality. [RFC5218] points out several problems with "wild success" in a protocol: undesirable side effects, performance problems, and becoming a high value attack target. Protocol and implementation design should take into account use cases that have not yet been thought of by building flexibility into protocols. Protocols should also remained focused on a narrow range of use cases; it is often wise to invent a new protocol than to extend a single protocol into a broad set of use cases.

- Protocols are sometimes replaced or updated to new versions in order to add new capabilities or features. Updating a protocol requires great care in providing for a transition mechanism between older and newer versions. draft-iab-protocol-transitions [I-D.iab-protocol-transitions] provides sound advice on protocol transition planning and mehanisms.

- Obscure, but legal, protocol features are often ignored or left unimplemented. Protocols must handle receiving unexpected information gracefully so they do not fail because of incomplete or partial implementations. Protocols should avoid specifying contradictory states, or features that will cause interoperability issues if multiple implementations choose to implement different feature sets.

- Monocultures are almost always bad. While multiple implementations can represent an interaction surface which increases complexity, particularly if a brad set of protocol capabilities and/or implementation features are used, using the same implementation at every point in a deployment results in a

monoculture. In a monoculture, a single event can trigger a defect in every router, causing a network failure. Monocultures must be carefully balanced against interaction surfaces; often this is best accomplished by using multiple implementations and minimal, widely implemented, and well understood protocol features.

A summary of the points above might be this: 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 often easy to assume "virtual oceans" are easier to boil than physical ones, or for an ocean to appear much smaller because it is being implemented in software. This is often deceptive. It is never helpful to boil the ocean whether in a design, an implementation, or a protocol.

2.2. Complexity

Complexity, as articulated by Mike O'Dell (see [RFC3439]), is "the primary mechanism which impede efficient scaling, and as a result is the primary driver of increases in both capital expenditures (CAPEX) and operational expenditures (OPEX)." At the same time, complexity cannot be "solved," but rather must be "managed." 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. See Alderson and Doyle [COMPLEXHARD] for a discussion of the relationship between hard problems and complex solutions.

The following sections contain observations which apply to the management of complexity in both protocol and network design.

2.2.1. Elegance

Elegance should be the goal of protocol and network design. Rather than seeking out simple solutions because they are simple, seek out solutions that will solve the problem in the simplest way possible (and no simpler). Often this will require:

- Ensuring the goal is actually the goal. Many times the goal is taken from the operational realm into the protocol design realm before enough thought has been applied to ensure the correct problem is being addressed.

- 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.

- Sometimes the complexity of the solution will overwhelm the use case; sometimes it is better to leave the apparent problem unsolved, or allow the community time and space to find a simpler solution.

2.2.2. Tradeoffs

There are always tradeoffs. For any protocol, network, or operational design decision, there will always be a tradeoff between at least two competing goals. If some problem appears to have a single solution without tradeoffs, this doesn't mean the tradeoffs don't exist. Rather, it means the tradeoffs haven't been discovered yet. In the area of protocol and network design, these tradeoffs often take the form of common "choose two or three" situations, such as "quick, cheap, high quality." In network and protocol design, the tradeoffs are often:

- The amount of state carried in the system and the speed at which it changes, or simply the state. The amount of state required to operate a system as it scales tends to be nonlinear. Some instances of this are described in [RFC3439] section 2.2.1, the Amplification Principle.

- The number of interaction surfaces between the components that make up the complete system, and the depth of those interaction surfaces. Some examples of surfaces are described in [RFC3439]section 2.2.2, the Coupling Principle. Layering is essentially a form of abstraction; all abstractions are subject to the law of leaky abstractions, [LEAKYABS] which states: "all nontrivial absractions leak."

- The desired optimization, including efficient use of network resources, optimal support for business objectives, and optimal support for a specific set of applications.

These three make up a "triangle problem." For instance, to increase the optimization of traffic flow through a network generally requires adding more state to the control plane, leading to problems in complexity due to amplification. To reduce amplification, the control plane (or perhaps the various functions the control plane serves) can be broken up into subsystems, or modules. Breaking the control plane up into subsystems, however, introduces interaction surfaces between the components, which is another form of complexity. [RFC7980] provides a good overview of network complexity; in particular, section 3 of that document provides some examples of complexity tradeoffs.

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.

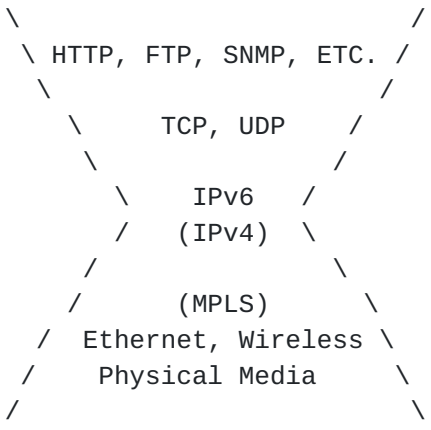

Figure 1

This layering emulates or mirrors many naturally occurring systems; it is a common strategy for managing complexity (see Meyer's presentation on complexity). [COMPLEXLAYER] 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 two 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 is often used as a "middle" subtransport layer, and at other times as "middle" sub data link layer; hence MPLS is difficult to classify within the strictly hierarchical model depicted here. 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 (see above on the law of leaky abstractions). 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, physical header, or tunnel encapsulation off the packet
- Examining the packet for errors, and determining if this packet needs to be punted to another process on the router
- 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 a match based on:

- The longest prefix containing the destination address (this is the most common matching element)
- A label, such as a flow label or MPLS label
- The source address or other header fields (not common)

The router then examines the information in the matching entry 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.

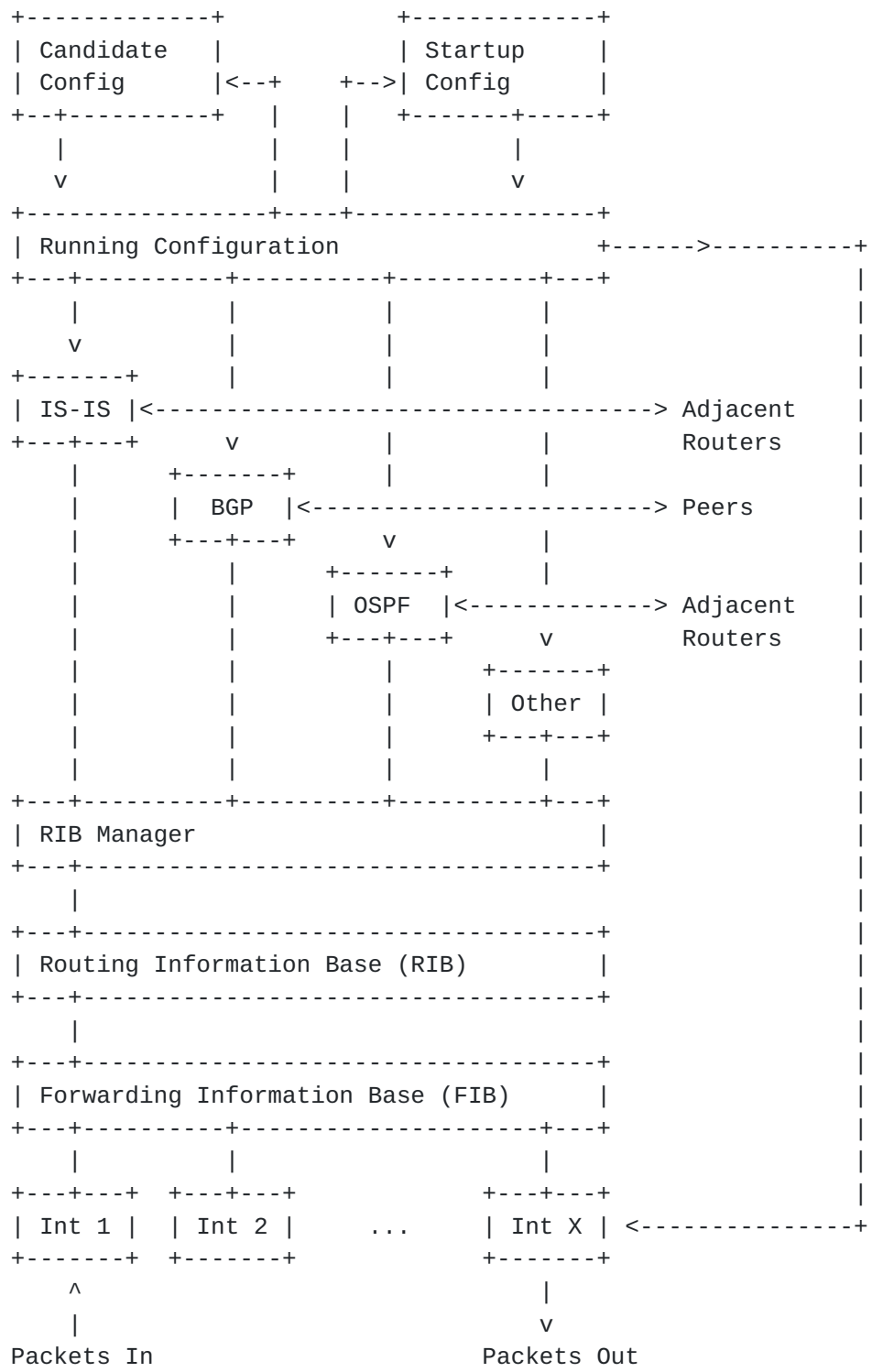
Candidate Config
Startup Config
Running Configuration
IS-IS
Adjacent Routers
BGP
Peers
OSPF
Adjacent Routers
Other
RIB Manager
Routing Information Base (RIB)
Forwarding Information Base (FIB)
Int 1
Int 2
...
Int X
Packets In
Packets Out

Figure 2

3. Requirements Related to Device Management and Security

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 routers and middleboxes 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, specifically targetting machine to machine intefaces, such as Remote Procesdure Calls (RPCs) and Application Programming Interface (API) solutions, to manage and configure routers and middleboxes. This section considers the various components of device management.

3.1. Programmable Device Access

Configuration primarily relates to the startup, candidate, and running configurations in the router model shown above. In order to deploy networks at scale, operators rely on automated management of router configuration. This effort has traditionally 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.

Vendors and implementors should implement machine readable interfaces with overlays to support human interaction, rather than human readable interfaces with overlays to support machine to machine interaction. Emphasis should be placed on machine to machine interaction for day to day operations, rather than on human readable interfaces, which are largely used in the process of troubleshooting. Within the realm of machine to machine interfaces, emphasis should be placed on marshaling information in YANG models.

To support automated router configuration, IPv6 routers and routers 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. Human Readable Device Access

To operate a network at scale, operators rely on the ability to access routers and middleboxes to troubleshoot and gather state manually 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 interfaces should provide information about the router (the whole device stack).

To support manual state gathering and troubleshooting, IPv6 routers and middleboxes 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 routers

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 router in and it just works without any manual configuration. The closer an operator can come to this ideal, the more MTBM and Operational Expenses (OPEX) can be reduced -- an important goals in the real world. IPv6 routers 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 router interfaces

SLAAC SHOULD be able to be disabled by operators who prefer to use some other mechanism for address management and assignment.

3.4. Authentication, Authorization, and Accounting

(Need some text here)

3.5. 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. To provide for effective counters to DoS and DDoS attacks directly on routers:

- Manufacturers and system integrators should test and clearly report the packet/traffic load handling capabilities of devices with and without various encryption methods enabled

- Routers should be able to police traffic destined to the control plane based on the rate of traffic received, including the ability ot police individual flows, targeted services, etc., at individual rates as described in [RFC6192]

- Ideally, devices should be able to statefully filter traffic destined to the control plane

There are other useful techniques for dealing with DDoS attacks at the network level, including: transferring sessions to a new address and abandoning the address under attack, using BGP communities to spread the attack over multiple ingress ports and "consume" it, and requiring mutual authentication before allocating larger resource pools to a connection. These techniques are not "device level," and hence are not considered further here.

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. Two major challenges face operators in the area of telemetry:

- Information that is laid out primarily for human, rather than machine, consumption. While human consumption of telemetry is important in some situations, this information should be supplied in a form that focuses on machine readability with an overlay or interpretor that allows human consumption.

- Software systems that require information to be queried (or polled or even pushed) on a per-item basis. This form of organization can produce a lot of information, and a lot of individual packets, very quickly, overwhelming monitoring systems and consuming a large amount of available network resources. Instead, telemetry should be focused on bulk collection.

There are three broad categories of telemetry: device state and traceability, topology state and traceability, and flow traceability. 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 State and Traceablity

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. In real deployments, it is often impossible to reach this ideal; however, reducing the languages and methods used, while focusing on machine readibility, can greatly ease the deployment and management of a large scale network. Specifically, IPv6 routers SHOULD support:

- [RFC6241]: NETCONF/RESTCONF transporting telemetry formatted according to YANG (see above)
- [RFC5424]: Syslog
- gRPC based telemetry interfaces [GRPC]
- SNMP as appropriate

Syslog and SNMP access for telemetry should be considered "legacy," and should not be the focus of new telemetry access development efforts.

4.2. Topology State and Traceability

IPv6 routers 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. For instance, being able to understand changes in the topology and utlization of a network can lead to insights about traffic flow and network growth that lead to a greater understanding of how the network is operating, where problems are developing, and how to improve the network's performance. To support systemic monitoring of the network topology, IPv6 devices SHOULD support at least:

- [RFC5424]: North-Bound Distribution of Link-State and Traffic Engineering (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

4.3. Flow 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. Specifically:

- The IPv6 address will change when a device changes where it connects to the network.

- A single host can have multiple addresses. For instance, a host may have one address per interface, or multiple addresses assigned through different mechanisms, or through multiple connection points.

- A single IPv6 address may represent many hosts, as in the case of a group of hosts reachable through multicast address, or a set of services reachable through an anycast address.

Because the host address may change at any time, 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] 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 SHOULD be addressed with a longer prefix length to prevent certain kinds of denial of service attacks ([RFC3627] originally mandated 64 bit prefix lengths on point-to-point links; [RFC6164] explains possible security issues with assigning a 64 bit prefix length to a point-to-point, and recommends a /127 instead)

- Although aggregation may only 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 multi-access interface should require additional configuration steps to prevent manual configuration errors

- Routers SHOULD NOT assume IPv6 prefix lengths only on nibble boundaries

- Routers SHOULD support any prefix length shorter or greater than /64

- 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 calcualte 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 (see the section on ICMP filtering below)
- 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 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 1.2ms of serialization delay. Each additional 1500 byte packet added to the queue in front of the 64 byte packet adds and addtional 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 megligable 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, described in [RFC7922], SHOULD be enabled to ensure correct point-to-point MTU configuration. Devices SHOULD also support:

- [RFC1191]: Path MTU Discovery
- [RFC1981]: Path MTU Discovery for IP version 6
- [RFC4821]: Packetization Layer Path MTU Discovery

5.4. ICMP Considerations

Internet Control Message Protocool (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 middleboxes of various kinds and/or ICMP filters configured on the ingress edge of a provider network, most often to prevent the discovery of reachable hosts and network topology. Routers implementing IPv6:

- SHOULD NOT filter ICMP unreachables by default, as this has negative impacts on many aspects of IPv6 operation, particularly path MTU
- SHOULD filter ICMP echo and echo response by default, to prevent the discovery of reachable hosts and topology.
- SHOULD 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
- SHOULD implement the filtering suggestions in [I-D.gont-opsec-icmp-ingress-filtering]

There are implications for path MTU discovery and other useful mechanisms in filtering and rate limiting ICMP. The tradeoff here is

between allowing unlimited ICMP, which would allow path MTU detection to work, or limiting ICMP in a way that prevents negative side effects for individual devices, and hence the operational capabilities of the network as a whole. Operators rightly limit ICMP to reduce the attack surface against their network, as well as the opportinity for "perfect storm" events that inadvertently reduce the capability of routers and middleboxes. Hence ICMP can be treated as "quasi-reliable" in many situations; existence of an ICMP message can prove, for instance, that a particular host is unreachable. The non-existence of an ICMP message, however, does not prove a particular host exists or does not.

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 router 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 routers. 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 Operation

While the transition to IPv6 only networks may take years (or perhaps decades), a number of operators are moving to deploy IPv6 on internal networks supporting transport and data center fabric applications more quickly. Routers and middleboxes that support IPv6 SHOULD support IPv6 only operation, including:

- Link Local addressing SHOULD be configurable and useable as the primary address on all interfaces on a device.

- IPv4 and/or MPLS SHOULD NOT be required for proper device operation. For instance, an IPv4 address should not be required to determine the router ID for any protocol. See [RFC6540] section 2.

- Any control plane protocol implementations SHOULD support the recommendations in [RFC7404] for operation using link local addresses only.

6. Future Considerations

(To be added)

6.1. Segment Routing

(To be added)

7. Security Considerations

This document addresses several ways in which devices designed to support IPv6 forwarding. Some of the recommendations here are designed to increase device security; for instance, see the section on device access. Others may intersect with security, but are not specifically targeted at security, such as running IPv6 link local only on links. These are not discussed further here, as they improve the security stance of the network. Other areas discussed in this draft are more nuanced. This section gathers the intersection between operational concerns and security concerns into one place.

ICMP security is already considered in the section on ICMP; it will not be considered further here. Link local only addressing wil increase security by removing transit only links within the network as a reachable destination.

7.1. Robustness and Security

Robustness, particularly in the area of error handling, largely improves security if designed and implemented correctly. Many attacks take advantage of mistakes in implementations and variations in protocols. In particular, any feature that is unevenly implemented among a number of implementations often offers an attack surface. Hence, reducing protocol complexity helps reduce the breadth of attack surfaces.

Another point to consider at the intersection of robustness and security is the issue of monocultures. Monocultures are in and of themselves a potential attack surface, in that finding a single failure mode can be exploited to take an entire network (or operator) down. On the other hand, reducing the number of implementations for any particular protocol will decrease the set of "random" features deployed in the network. These two goals will often be opposed to one another. Network designers and operators need to consider these two sides of this tradeoff, and make an intelligent decision about how much diversity to implement versus how to control the attack surface represented by deploying a wide array of implementations.

7.2. Programmable Device Access and Security

Programmable interfaces, including programmable configuration, telemetry, and machine interface to the routing table, introduce a large attack surface; operators should be careful to ensure this attack surface is properly secured. Specifically:

- Prevent external access to any administrative access points used for device programmability
- Use AAA systems to ensure only valid devices and/or users access devices
- Rate limit the change rate and protect management interfaces from DoS and DDoS attacks

Such interfaces should be treated no differently than SSH, SFTP, and other interfaces available to manage routers and middleboxes.

7.3. Zero Touch Provisioning and Security

Zero touch provisioning opens a new attack surface; insider attackers can simply install a new device, and assume it will be autoconfigured into the network. A "simple" solution would be to install door locks, but this will likely not be enough; defenses need to be layered to be effective. It is recommended that devices installed in the network need to contain a hardware or software identification system that allows the operator to identify devices that are installed in the network.

8. Conclusion

The deployment of IPv6 throughout the Internet marks a point in time where it is good to review the overall Internet architecture, and assess the impact on operations of these changes. This document provides an overview of a lot of these changes and lessons learned,

as well as providing pointers to many of the relevant documents to understand each topic more deeply.

9. References

9.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[COMPLEXHARD] Alderson, D. and J. Doyle, "Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures", 2010, <http://ieeexplore.ieee.org/abstract/document/5477188/?reload=true>.

[COMPLEXLAYER] Meyer, D., "Macro Trends, Architecture, and the Hidden Nature of Complexity", 2010, <http://www.slideshare.net/dmm613/macro-trends-complexityandsdn-32951199>.

[DoD] Wikipedia, "The Internet Protocol Suite", 2016, <https://en.wikipedia.org/wiki/Internet_protocol_suite>.

[GRPC] gRPC, "gRPC", 2016, <http://www.grpc.io>.

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

[I-D.iab-protocol-transitions] Thaler, D., "Out With the Old and In With the New: Planning for Protocol Transitions", draft-iab-protocol-transitions-05 (work in progress), January 2017.

[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-06 (work in progress), October 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)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.

[I-D.ietf-i2rs-yang-l2-network-topology]
Dong, J. and X. Wei, "A YANG Data Model for Layer-2 Network Topologies", draft-ietf-i2rs-yang-l2-network-topology-03 (work in progress), July 2016.

[I-D.ietf-i2rs-yang-l3-topology]
Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", draft-ietf-i2rs-yang-l3-topology-08 (work in progress), January 2017.

[I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-11 (work in progress), February 2017.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

[LEAKYABS] Spolsky, J., "The Law of Leaky Abstractions", 2002, <https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/>.

[OPENCONF] OpenConfig, "Openconfig release YANG models", 2016, <https://github.com/openconfig/public/tree/master/release>.

[OSI] Wikipedia, "OSI Model", 2016, <https://en.wikipedia.org/wiki/OSI_model>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC0854] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, DOI 10.17487/RFC0854, May 1983, <http://www.rfc-editor.org/info/rfc854>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://www.rfc-editor.org/info/rfc1191>.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <http://www.rfc-editor.org/info/rfc1812>.

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

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC3439] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, DOI 10.17487/RFC3439, December 2002, <http://www.rfc-editor.org/info/rfc3439>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC3627] Savola, P., "Use of /127 Prefix Length Between Routers Considered Harmful", RFC 3627, DOI 10.17487/RFC3627, September 2003, <http://www.rfc-editor.org/info/rfc3627>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <http://www.rfc-editor.org/info/rfc4253>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

[RFC5218] Thaler, D. and B. Aboba, "What Makes For a Successful Protocol?", RFC 5218, DOI 10.17487/RFC5218, July 2008, <http://www.rfc-editor.org/info/rfc5218>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[RFC6164] Kohno, M., Nitzan, B., Bush, R., Matsuzaki, Y., Colitti, L., and T. Narten, "Using 127-Bit IPv6 Prefixes on Inter-Router Links", RFC 6164, DOI 10.17487/RFC6164, April 2011, <http://www.rfc-editor.org/info/rfc6164>.

[RFC6177] Narten, T., Huston, G., and L. Roberts, "IPv6 Address Assignment to End Sites", BCP 157, RFC 6177, DOI 10.17487/RFC6177, March 2011, <http://www.rfc-editor.org/info/rfc6177>.

[RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <http://www.rfc-editor.org/info/rfc6192>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6540] George, W., Donley, C., Liljenstolpe, C., and L. Howard, "IPv6 Support Required for All IP-Capable Nodes", BCP 177, RFC 6540, DOI 10.17487/RFC6540, April 2012, <http://www.rfc-editor.org/info/rfc6540>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <http://www.rfc-editor.org/info/rfc7217>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <http://www.rfc-editor.org/info/rfc7224>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

[RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local Addressing inside an IPv6 Network", RFC 7404, DOI 10.17487/RFC7404, November 2014, <http://www.rfc-editor.org/info/rfc7404>.

[RFC7527] Asati, R., Singh, H., Beebee, W., Pignataro, C., Dart, E., and W. George, "Enhanced Duplicate Address Detection", RFC 7527, DOI 10.17487/RFC7527, April 2015, <http://www.rfc-editor.org/info/rfc7527>.

[RFC7663] Trammell, B., Ed. and M. Kuehlewind, Ed., "Report from the IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)", RFC 7663, DOI 10.17487/RFC7663, October 2015, <http://www.rfc-editor.org/info/rfc7663>.

[RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <http://www.rfc-editor.org/info/rfc7922>.

[RFC7980] Behringer, M., Retana, A., White, R., and G. Huston, "A Framework for Defining Network Complexity", RFC 7980, DOI 10.17487/RFC7980, October 2016, <http://www.rfc-editor.org/info/rfc7980>.

Authors' Addresses

Zaid Ali Kahn, Editor
LinkedIn
xxx
xxx, CA xxx
USA

Email: zaid@linkedin.com

John Brzozowski, Editor
Comcast
xxx
xxx, xxx xxx
USA

Email: John_Brzozowski@comcast.com

Russ White, Editor
LinkedIn
Oak Island, NC 28465
USA

Email: russ@riw.us