IPv6 Operations Working Group (v6ops)
Internet-Draft
Intended status: Informational
Expires: June 14, 2021

F. Gont
G. Gont
SI6 Networks

December 11, 2020

IPv6 Addressing Considerations
draft-gont-v6ops-ipv6-addressing-considerations-00

Abstract

IPv6 addresses can differ in a number of properties, such as scope, stability, and intended usage type. This document analyzes the impact of these properties on aspects such as security, privacy, interoperability, and network operations. Additionally, it identifies challenges and gaps that currently prevent systems and applications from leveraging the increased flexibility and availability of IPv6 addresses.

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 June 14, 2021.

Copyright Notice

Table of Contents

1. Introduction

IPv6 addresses can differ in a number of properties, such as address scope (e.g. link-local vs. global), stability (e.g. stable addresses vs. temporary addresses), and intended usage type (outgoing communications vs. incoming communications). While often overlooked, these properties have direct impact on areas such as security, privacy, interoperability, and network operations.

IPv6 hosts typically configure addresses based on local system policy, which tends to be static and irrespective of the specific network the host attaches to. For example, most IPv6 host implementations configure one link-local address for each network interface, and one stable and one (or more) temporary addresses per each autoconfiguration prefix advertised via Stateless Address Auto-configuration (SLAAC) [RFC4862] for each network interface. Additionally, in scenarios where the network provides address configuration via both SLAAC and DHCPv6, host typically configure addresses using both protocols; as a result, hosts will typically add one DHPv6-leased address per local prefix to the set of configured addresses.

Each application on a given system could have its own set of requirements or expectations for the properties of the underlying IPv6 addresses. For example, an application meaning to offer a public service might expect to employ addresses that are both globally-reachable [RFC8190] and stable [RFC7721] [RFC8064], while a privacy-sensible client application might prefer short-lived temporary addresses [I-D.ietf-6man-rfc4941bis], or might even expect to employ single-use ("throw-away") IPv6 addresses when connecting to

public servers. However, the subtleties associated with IPv6 addresses are often ignored or overlooked by application programmers and network administrators, and thus hosts could end up making suboptimal choices when configuring addresses, while applications could fail to signal their requirements and preferences to the underlying system. Additionally, limitations in the current Application Programming Interfaces (APIs) along with implementation constraints in some network devices can also limit the ability of applications to leverage the increased flexibility of IPv6 addressing. This not only results in a failure to leverage the increased addressing flexibility provided by IPv6 but, at times, also in unintended consequences.

This document identifies a set of properties that can be associated with IPv6 addresses (such as scope and stability), and analyzes the impact of these properties on areas ranging from security and privacy to network operations, with the goal of providing guidance about IPv6 address usage, and identifying challenges and gaps that currently prevent systems and applications from leveraging the increased flexibility and availability of IPv6 addresses.

2. Terminology

This document employs the definitions of "public address", "stable address", and "temporary address" from Section 2 of [RFC7721].

This document employs the definition of "globally reachable" from Section 2.1 of [RFC8190].

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.

3. Conventions

3.1. Legacy Specifications and Schemes

IPv6 SLAAC has traditionally employed schemes for generating Interface Identifiers (IIDs) that have negatively affected the security and privacy properties of IPv6 addresses. For example, IPv6

SLAAC originally generated stable addresses by embedding the underlying link-layer address in the IPv6 Interface Identifier (IID), thus negatively affecting the security and privacy properties of IPv6 addresses [RFC7721] [RFC7707]. Similarly, IPv6 temporary addresses [RFC4941] reused the same randomized IID for multiple auto-configuration prefixes [RFC4941], thus allowing for network activity correlation across different addresses of the same host.

These schemes have become formally superseded by other schemes, such as [RFC7217] and [I-D.ietf-6man-rfc4941bis], that mitigate the issues present in the legacy schemes. Therefore, does not discuss any implications arising from legacy IID generation algorithms.

NOTE:
The security and privacy implications of such schemes are discussed in [RFC7721], [RFC7707], and [RFC7217].

3.2. Unique Local IPv6 Unicast Addresses (ULAs)

Unique Local IPv6 Unicast Addresses (ULAs) are considered to have global scope, but non-global reachability. However, the only reasons for which these addresses are considered to have "global scope" are:

- given two different networks that employ ULAs, it's unlikely that they will employ the same ULA prefix
- the ULA address block formally belongs to the Global Unicast Address (GUA) address block

The ULA prefixes of two different networks that e.g. need to be merged will be (statistically) different *if* sites generate their ULA prefixes following the recommendations in [RFC4193] -- i.e., randomize the ULA prefix. However, this is not enforceable, and there exists anecdotal evidence that some sites employ non-randomized ULA prefixes, possibly in the hope of employing prefixes that are easier to express via the IPv6 address notation (e.g. fd00::/48).

The ULA address block has been carved out of the GUA address block, without updating the IPv6 Addressing Architecture ([RFC4291]) to

define the ULA address block as a different address type (e.g. "IPv6 private address space"). However, [RFC4193] notes that ULAs are not expected to be globally-routable, and the ULA prefix is commonly included in "bogon filters" typically enforced by network operators and administrators, thus limiting the reachability of these addresses.

Therefore, ULAs are not globally meaningful and thus, for most (if not all) practical purposes, ULAs can be considered to have non-global scope. For this reason, ULAs are treated as non-global scope addresses, even when from a specifications point of view they have global scope.

4. IPv6 Address Properties

There are, at least, four properties that can be associated with every IPv6 address:

- Scope
- Reachability
- Stability
- Provider Dependency

The address scope essentially represents the area of the network where an address can be expected to uniquely identify a system or set of systems, or conceptually, the area of the network where an given address is meaningful. For example, link-local addresses are only meaningful within a given network link, and are expected to be unique only within such network link.

Address reachability represents the area of the network, where an address can be expected to be used for receiving and transmitting packets. Besides the semantics of specific address types, reachability can be affected by network devices: for example, Customer Edge Routers (CE Routers) that enforce a filtering policy of "only allowing outgoing communications" can render otherwise globally reachable addresses as "unreachable from the public Internet, unless communication is initiated from the customer's network".

The stability of an address is associated with the invariance of an address over time. For example, a manually-configured address will typically remain stable while the node remains attached to the same subnet, while a temporary address will, by definition, change over time. While address stability does depend on the inherent properties of a given address (e.g. stable vs. temporary), it also depends on other factors, such as provider dependency: if a network employs a prefix that is assigned/leased by an upstream provider, then the overall stability of the addresses will also depend on the stability of the network prefix leased/assigned by the upstream.

Provider-dependency is typically discussed in the context of Global Unicast Addresses, where the address space may be allocated by an Internet Service Provider (ISP) (and hence "provider aggregatable") or by a Regional Internet Registry (RIR) (and hence "provider independent"). However, this document considers "provider dependency" in a more general way: "provider aggregatable" address space is assigned or leased by the upstream (which may or may not be an ISP) from the provider's address space, and thus has a topological relationship with the upstream's address space, whereas "provider independent" address space is "owned" by the network in question and does not necessarily have a topological relationship with the upstream.

4.1. Address Scope Considerations

The IPv6 address scope [RFC4007] has a direct implication on address reachability: the address scope essentially limits reachability. For example, addresses that have a non-global scope are not, in principle, globally reachable.

NOTE:

This assumption becomes invalid if technologies such as Network Prefix Translation (NPT) [RFC6296] are employed, though. However, strictly speaking, in these scenarios the non-global addresses are still not globally reachable, but rather the middle-box acts as an interface to the "external realm" via globally-reachable addresses (i.e., the middle-box has an interface within the scope of the non-global addresses, and globally-reachables address that are used to communicate with the "external realm").

The IPv6 address scope can, in some scenarios, limit the attack exposure of a node as a result of the implicit isolation provided by a non-global address scopes. For example, a node that only employs link-local addresses will, in principle, only be exposed to attacks from other nodes on the same local link.

The potential protection provided by a non-global-scope addresses should not be regarded as a complete security strategy, but rather as a form of "prophylactic" security (see [I-D.gont-opsawg-firewalls-analysis]).

We note that non-global scope addresses are normally only of use for limited number of applications/protocols that operate on a limited scope (e.g., mDNS), or deployments where the intended participants have been deployed in a limited area of the network topology (e.g., OpenSSH client and server attached to the same link, both employing link-local addresses).

The address scope can at times be somewhat related with the provider dependency property. For example, link-local addresses are, by definition, provider independent. In the same light, a ULA prefix generated by a local router will be, by definition, provider independent. However, a router might also be leased an ULA sub-prefix from its upstream, in which case this prefix would be "provider dependent".

4.2. Provider Dependency

Provider-dependency is typically discussed in the context of Global Unicast Addresses, where the address space may be allocated by an Internet Service Provider (ISP) (and hence "provider agregatable") or by a Regional Internet Registry (RIR) (and hence "provider independent"). However, this document considers "provider dependency" in a more general way: "provider agregatable" address space is assigned or leased by the upstream (which may or may not be an ISP) from the provider's address space, and thus has a topological relationship with the upstream's address space, whereas "provider independent" address space is "owned" by the network in question and

does not necessarily have a topological relationship with the

upstream.

An implicit consequence of PA address space is that its use is tied to the specific provider/upstream that has assigned/leased the address space. This means that multi-homing (employing the address space with multiple providers/upstreams) is not possible, and that a renumbering event at the upstream could lead to a renumbering event of the local network.

As a result, provider-dependency can affect address stability, with PI address space generally having better stability properties. For example, a home network could internally employ both ULAs and GUAs, where a ULA prefix is locally generated by the Customer Edge Router (CE Router), and global prefix is leased by the ISP via DHCPv6 Prefix Delegation (DHCPv6-PD) [RFC8415]. If for some reason there was an outage involving the connection with the upstream ISP, the prefix lease time would eventually expire, and therefore addresses configured for such prefix would need to be invalidated. Similarly, if upon prefix lease expiration the ISP were to lease a new IPv6 prefix rather than renew the previously employed prefix, the network would need to be renumbered. On the other hand, ULA prefixes locally generated and advertised by the CE Router will not require renewal from the ISP, since they are locally generated by the CE Router.

NOTE:
As noted above, while strictly speaking ULAs belong to the global unicast address space, they can be considered non-global addresses for all practical purposes.

Similarly, an organizational network that employs PI address space obtained from a RIR would be able to keep the same address space, even if the upstream network is renumbered.

4.3. Address Reachability

Address reachability represents the area of the network (and the associated conditions), where an address can be expected to be used for receiving and transmitting packets. As noted in Section 4.1, the address scope has a direct implication on address reachability, since address reachability is limited to a subset of the address scope. For example, only global-scope addresses can be globally reachable.

However, besides the reachability semantics of each address type, network filtering policies may also affect address reachability. For example, there is widespread deployment of Customer Edge Routers that implement a (stateful) filtering policy of "only allowing outgoing communications" -- mimicking the filtering policy enforced (as a

side-effect) by IPv4 NATs. In such scenarios, even otherwise globally-reachable addresses become unreachable, unless:

- communication is initiated from the internal network, or,
- the CE Router is manually configured override the default filtering policy, or,
- a technology to dynamically override the filtering policy (such as UPnP [UPnP] or PCP [RFC6887]) is employed.

Address reachability is what ultimately determines the application architecture that may be employed by an IPv6 node.

NOTE:
Ironically, an IPv6-only host (with global-scope addresses) attached to a home network where the CE Router "only allows outgoing communications" and does not implement a protocol such as UPnP [UPnP] or PCP [RFC6887], will normally have a harder time using peer-to-peer (P2P) applications than an IPv4-only host (with a private address) attached to a home network where the CE Router employs NAT but implements a protocol such as UPnP or PCP.

Address reachability has a direct impact on security, since the ability to attack a system normally relies on the ability of the attacker to reach the system in the first place. Firewalls [I-D.gont-opsawg-firewalls-analysis] are, indeed, devices that are specifically devoted to administer address reachability.

4.4. Address Stability Considerations

The stability of an address has two associated security/privacy implications:

- Ability of an attacker to correlate network activity
- Exposure to attack

For obvious reasons, an address that is employed for multiple communication instances allows the aforementioned network activities to be correlated. The longer an address is employed (i.e., the more stable it is), the longer such correlation will be possible. In the worst-case scenario, a stable address that is employed for multiple communication instances over time will allow all such activities to be correlated. On the other hand, if a host were to generate (and eventually "throw away") one new address for each communication

instance (e.g., TCP connection), network activity correlation would be mitigated.

NOTE:
The security and privacy implications of predictable addresses are discussed in [RFC7721] and [RFC7707].

Typically, when it comes to attack exposure, the longer an address is employed the longer an attacker is exposed to attacks. While such exposure is traditionally associated with the stability of the address, the usage type of the address may also have an impact on attack exposure (see Section 5.2).

A popular approach to mitigate network activity correlation is the use of "temporary addresses" [RFC4941]. Temporary addresses are typically auto-configured and employed along with stable addresses, with the temporary addresses employed for outgoing communications, and the stable addresses employed for incoming communications.

NOTE:
Ongoing work [I-D.ietf-6man-rfc4941bis] aims at updating [RFC4941] such that temporary addresses can be employed without the need to configure stable addresses.

We note that the extent to which temporary addresses provide improved mitigation of network activity correlation and/or reduced attack exposure may be questionable and/or limited in some scenarios. For example, a temporary address that is reachable for, say, a few hours has a questionable "reduced exposure" (particularly when automated attack tools do not typically require such a long period of time to complete their task). Similarly, if network activity can be correlated for the life of such address (e.g., on the order of several hours), such period of time might be long enough for the attacker to correlate all the network activity he is meaning to correlate. However, they do introduce a limit to the attack window, and the amount of time during which address-based network activity correlation can be performed.

In order to better mitigate network activity correlation and/or possibly reduce host exposure, an implementation might want to either reduce the preferred lifetime of a temporary address, or even better, generate one new temporary address for each new transport protocol

instance. However, the associated lifetime/stability of an address may have a negative impact on the network (please see Section 6.3.

Additionally, enforcing a maximum lifetime on IPv6 addresses may cause long-lived TCP connections to fail. For example, an address becoming "Invalid" (after transitioning through the "Preferred" and "Deprecated" states) would cause the TCP connections employing them to break, which would in turn cause e.g. long-lived SSH sessions to break/fail.

In some scenarios, attack exposure may be mitigated by limiting the usage of temporary addresses to outgoing connections, and prevent such addresses from being used for incoming connections (please see Section 5.2).

5. IPv6 Address Usage

5.1. Default Address Selection

Applications use system API's to select the IPv6 addresses that will be used for incoming and outgoing connections. These choices have consequences in terms of privacy, security, stability and performance.

Default Address Selection for IPv6 is specified in [RFC6724]. The selection starts with a set of potential destination addresses, such as returned by getaddrinfo(), and the set of potential source addresses currently configured for the selected interfaces. For each potential destination address, the algorithm will select the source address that provides the best route to the destination, while choosing the appropriate scope and preferring temporary addresses. The algorithm will then select the destination address, while giving a preference to reachable addresses with the smallest scope. The selection may be affected by system settings. We note that [RFC6724] only applies for outgoing connections, such as those made by clients trying to use services offered by other hosts.

We note that [RFC6724] selects IPv6 addresses from all the currently available addresses on the host, and there is currently no way for an application to indicate expected or desirable properties for the IPv6 source addresses employed for such outgoing communications. For example, a privacy-sensitive application might want that each

outgoing communication instance employs a new, single-use IPv6 address, or to employ a new reusable address that is not employed or reusable by any other application on the host. Reuse of an IPv6 address by an application would allow the correlation of all network activities corresponding to such application as being performed by the same host, while reuse of an IPv6 address by multiple different applications would allow the correlation of all such network activities as being performed by the host with such IPv6 address.

When devices provide a service, the common pattern is to just wait for connections over all addresses configured on the device. For example, applications using the BSD Sockets API will commonly bind() the listening socket to the undefined address. This long-established behavior is appropriate for devices providing public services, but can have unexpected results for devices providing semi-private

services, such as various forms of peer-to-peer or local-only applications.

This behavior leads to three problems: device tracking, discussed in Section 6.1.2; unexpected address discovery, discussed in Section 6.1.3; and availability outside the expected scope, discussed in Section 6.1.4. These problems are caused in part by the limitations of available address selection API, discussed in Section 7.1.

5.2. Usage Type Considerations

IPv6 hosts can both stable [RFC8064] and/or temporary [RFC4941] addresses, in which case stable addresses are typically employed for incoming (server-like) communications, while temporary addresses are employed for outgoing (client-like) communications. That is, the stability properties of an address have an implicitly associated usage type.

A node that employs one of its addresses to communicate with an external server (i.e., to perform an "outgoing connection") will expose that address to the other communicating system. For example, once the external server receives an incoming connection, the corresponding server might launch an attack against the aforementioned address. A real-world instance of this type of

scenario has been documented in [Hein].

However, we note that employing an IPv6 address for outgoing communications need not increase the exposure of local services to other parties. For example, nodes could employ temporary addresses only for outgoing communications, and disallow their use for incoming communications. Thus, external nodes that learn about a client's addresses could not really leverage such addresses for actively contacting clients.

Section 5 discusses current IPv6 address usage, along with possible improvements.

5.3. Current Alternatives for IPv6 Address Usage

5.3.1. Incoming communications

There are a number of ways in which a system or network may affect which addresses (and how) may be employed for different services and cases. Namely,

- TCP/IP stack address filtering

- Application-based address filtering
- Firewall-based address filtering

Clearly, the most elegant approach for address selection would be for applications to be able to specify the properties of the addresses they are willing to employ by means of an API, such the TCP/IP stack itself could "filter" which addresses are allowed for the given service/application. For example, an application could specify the stability and scope properties of the addresses on which incoming communications should be accepted, such that the application can be relieved from dealing with low-level networking details, portability is improved, and duplicate code in applications is avoided. However, constraints in the current APIs (see Section 7.1) limit the ability of application programmers for leveraging this technique. Alternatively, services could be bound to specific (explicit) addresses, rather than to all locally-configured addresses. However, there are a number of short-comings associated with this approach.

Firstly, an application would need to be able to learn all of its addresses and associated properties, something that tends to be non-trivial and non-portable, and that also makes applications protocol-dependent, unnecessarily. Secondly, the BSD Sockets API does not allow a socket to be bound to a subset of the node's addresses. That is, sockets can be bound to a single address or to all available addresses (wildcard), but not to a subset of all the configured addresses.

Another possible approach would be for applications to e.g. bind services to all available addresses, and perform the associated selection/filtering at the application level. While possible, this would have a number of drawbacks. Firstly, it would require applications to deal with low-level networking details, lead to duplicated code in all applications, and also negatively affect portability. Secondly, performing address/selection filtering at the application level may not mitigate some possible attacks. For example, port scanning will still be possible, since the aforementioned filtering would only be performed once e.g. UDP packets are received or TCP connections are established.

A client could simply run a host-based firewall that only allows incoming connections on the stable addresses. This would be clearly more of an operational approach for achieving the desired functionality, and would require good firewall/host integration (e.g., the firewall should be able to tell stable vs. temporary addresses), would require the client to run additional firewall software for this specific purpose, etc. In other scenarios, a network-based firewall could be configured to allow outgoing communications from all internal addresses, but only allow incoming

communications to stable addresses (i.e., allow such addresses either manually, or via a helper protocol such as [UPnP] or PCP [RFC6887]). For obvious reasons, this is generally only applicable to networks where incoming communications are allowed to a limited number of hosts/servers.

5.3.2. Outgoing communications

An application might be able to obtain the list of currently-configured addresses, and subsequently select an address with desired properties, and explicitly "bind" the address to the socket, to

override the default source address selection.

However, this approach is problematic for a number of reasons. Firstly, there is no portable way of obtaining the list of currently-configured addresses on the local node, and even less to check for address properties such "valid lifetime". Secondly, as discussed in Section 5.3.1, it would require application programmers to understand all the subtleties associated with IPv6 addressing, and would also lead to duplicate code on all applications. Finally, applications would be limited to use already-configured addresses and unable to trigger the generation of new addresses where desirable (e.g. the generation of a new single-use address for this application instance or communication instance).

6. Current Issues

6.1. Sub-optimal IPv6 Address Usage

6.1.1. Correlation of Network Activity

As discussed in [RFC7721], a node that reuses an IPv6 address for multiple communication instances will enable the correlation of such network activities. This could be the case when the same IPv6 address is employed by several instances of the same application (e.g., a browser in "privacy" mode and a browser in "normal" mode), or when the same IPv6 address is employed by two different applications on the same node (e.g., a browser in "privacy" mode, and an email client).

Particularly for privacy-sensitive applications, an application or system might want to limit the usage of a given IPv6 address to a single communication instance, a single application, a single user on the system, etc. However, given current APIs, this is practically impossible.

6.1.2. Passive Host Tracking

The stable addresses recommended in [RFC8064] use stable IIDs defined in [RFC7217]. One key part of that algorithm is that if a device

connects to a given network at different times, it will always configure the same IPv6 addresses on that network. If the device hosts a service ready to accept connections on that stable address, adversaries can test the presence of the device on the network by attempting connections to that stable address. Stable addresses used by listening services will thus enable testing whether a specific device is returning to a particular network, which in a number of cases might be considered a privacy issue.

6.1.3. Unintended Service Disclosure

Systems like DNS-Based Service Discovery [RFC6763] allow clients to discover services within a limited scope, that can be defined by a domain name. These services are not advertised outside of that scope, and thus do not expect to be discovered by random parties on the Internet. However, such services may be easily discoverable if they listen for connections to IPv6 addresses that a client process also uses as source address when connecting to remote servers.

NOTE:
An example of such service disclosure is described in [Hein]. A network manager observed scanning traffic directed at the temporary addresses of local devices. The analysis in [Hein] shows that the scanners learned the addresses by observing the device contact an NTP service ([RFC5905]). The remote scanning was possible because the local services were accepting connections on all configured addresses, including temporary addresses.

It is obvious from this example that local services are disclosed because they are bond to the same IPv6 addresses that are also used by clients for outgoing communications with remote systems. But the overlap between "client" and "server" addresses is only one part of the problem. Suppose that a device hosts both a video game and a home automation application. The video game users will be able to discover the IPv6 address of the game server. If the home automation server listens to the same IPv6 addresses, it is now exposed to connection attempts by all these users. That, too, increases the exposure of the home automation server.

We note that a host or network that wants to limit access to local services should filter incoming connection attempts by affecting address reachability (see Section 4.3) via firewalls [I-D.gont-opsawg-firewalls-analysis] and/or the use of IPv6 addresses of appropriate scope (see Section 4.1). However, it is also prudent

to avoid unintended service disclosure by suboptimal reuse of IPv6 addresses, as discussed in this section.

6.1.4. Availability Outside the Expected Scope

The IPv6 addressing architecture [RFC4291] defines multiple address scopes, with devices often configured with globally reachable unicast addresses, link local addresses, and Unique Local IPv6 Unicast Addresses (ULA) [RFC4193]. Availability outside the expected scope happens when a service is expected to be available only in some local scope, but inadvertently becomes available from outside of that scope. That could happen, for example, if a service is meant to be available only on a given link, but becomes reachable through ULA or through globally reachable addresses, or if a service is meant to be available only inside some organization's perimeter and becomes reachable through globally reachable addresses. This will commonly happen if a service intended for some local scope is programmed to bind() to the "unspecified" addresses, which in practice means every address configured for the device (please see Section 7.1).

6.2. Sub-optimal Address Configuration

6.2.1. Number of Addresses

Two mechanisms exist for automatic network configuration: SLAAC and DHCPv6. DHCPv6 centralizes network configuration and address assignment, and may thus prevent hosts from leveraging the increased flexibility and availability of IPv6 addresses. On the other hand, SLAAC may also result in network configuration anarchy, where hosts may e.g. configure and use addresses in a way that may render the network virtually unusable (please see Section 6.3.1).

Most of the challenges associated with the use of multiple addresses can be addressed by allocating one /64 per host via mechanisms such as DHCPv6-PD. However, support for such mechanisms in host implementations and e.g. the LAN-side of CE Routers is not widespread, and thus is currently unfeasible. On the other hand, SLAAC lacks the means for the network to convey information about e.g., the number of addresses per host the network is able or willing to support.

NOTE:
Use of a /64 prefix per host could render techniques such as temporary addresses [RFC4941] ineffective, since hosts would become identified by corresponding /64 prefix.

6.2.2. SLAAC/DHCPv6 Interaction

Many CE Routers offer address configuration via both SLAAC and DHCPv6, by including Prefix Information Options (PIOs) in Router Advertisement messages, and also setting the "M" such messages. This has a number of implications:

- The outcome of the configuration process is non-deterministic, difficulting network troubleshooting (see [I-D.ietf-v6ops-dhcpv6-slaac-problem]).

- Nodes end up configuring more addresses than needed (or even used), normally configuring multiple stable addresses for each autoconfiguration prefix, with at least one address for each configuration mechanism (SLAAC and DHCPv6).

- A host can end up employing stable and predictable addresses resulting configured via DHCPv6, even when effort has been made to mitigate security and privacy issues associated with IPv6 addresses for the SLAAC-configured addresses (i.e., [RFC7217] and [RFC4941]).

6.3. Operation of Multi-Prefix/Multi-Address/Multi-Router Networks

6.3.1. Implications of Addresses

Network deployments are currently recommended to provide multiple IPv6 addresses to general-purpose hosts [RFC7934]. However, in some scenarios, use of a large number of IPv6 addresses may have negative implications on network devices that need to maintain entries for each IPv6 address in network data structures (e.g., [RFC7039]). Additionally, concurrent active use of multiple IPv6 addresses will normally increase neighbour discovery traffic if Neighbour Caches in network devices are not large enough to store all addresses on the link. This can impact performance and energy efficiency on networks on which multicast is expensive (e.g. [I-D.ietf-mboned-ieee802-mcast-problems]). Finally, network devices may interpret the use of a number of addresses above a certain threshold as a security event, and block the offending device from using the network.

6.3.2. Legitimate Network Activity Correlation

The desires of protecting individual privacy versus the desire to effectively maintain and debug a network can conflict with each other. For example, having clients use addresses that change over time will make it more difficult to track down and isolate operational problems. For example, when looking at packet traces, it

could become more difficult to determine whether one is seeing behavior caused by a single errant machine, or by a number of them.

6.3.3. Routing in Multi-Prefix/Multi-Router Networks

If the network is provided with multiple upstreams via different routers, each of the upstream will provide its PA address space (see Section 4.2) and local hosts will typically configure addresses for each of such prefixes. In this scenarios, packets sourced from a given prefix should only employ the local router that announced that prefix, since otherwise the packets might be dropped as a result of ingress/egress filtering [RFC2827]. Unfortunately, the traditional Neighbor Discovery [RFC4861] can advertise routes only with a per-destination granularity, irrespective of the source address/prefix.

[RFC8028] finally addressed the most important challenges associated with these scenarios. However, [RFC8028] is not widely implemented. As a result, operating a multi-prefix/multi-router IPv6 network represents a major challenge -- if at all possible.

6.3.4. Renumbering

The challenges posed by network renumbering have been known for a very long time [RFC5887], with renumbering being analyzed with the assumption that the network topology remains stable and the network is renumbered.

However, in scenarios where a host is moved to a different network without the host detecting the network disattach/re-attach event, or where the network a host attaches to is moved to a different point of the network topology, the aforementioned host will also perceive a renumbering event. In an era in which moving virtual machines, containers, and networks around a network topology is commonplace,

and where mobile systems changing network connectivity to and from e.g. WiFi and 4G is also commonplace, renumbering events are anything but rare.

One of the challenges represented by network renumbering is how hosts can infer that an existing network prefix and associated address(es) have become stale and should be removed and replaced by new prefixes and addresses. In scenarios where the network topology does not change and the network is renumbered, network elements may be aware about the renumbering event and signal this condition to attached systems (i.e., signal that existing network configuration information should be removed and replaced). However, in scenarios where it is the host, virtual machine or container (or the network they are attached to) that move around the network topology, the network will not be able to signal the "renumbering event", and the renumbered

host, virtual machine, or container might or might not be able to detect the event (e.g., via link-down/link-up events).

Unfortunately, both SLAAC and DHCPv6 assume network configuration information to be somewhat stable. SLAAC has traditionally employed long lifetimes for network configuration information, meaning that stale information could be employed for an unacceptably long period of time. DCHPv6 has traditionally suffered from the same problem but, in addition, there is no widespread support for RECONFIGURE messages, so even if the network were in a position to signal a renumbering event, in practice hosts would normally have to rely on expiration of lease times for stale information to be cleared up.

Some of these problems have been discussed in detail in [I-D.ietf-v6ops-slaac-renum], and there is ongoing work [I-D.ietf-6man-slaac-renum] [I-D.ietf-v6ops-cpe-slaac-renum] to mitigate their effects.

7. Current Gaps that Prevent Leveraging IPv6 Addressing

7.1. Better Address Selection APIs

Application developers using the BSD Sockets API can "bind()" a listening socket to a specific address, and ensure that the application is only reachable through that address. In theory, careful selection of the binding address could mitigate the problems

described in Section 6.1. Binding services to temporary addresses could mitigate the ability of an attacker from testing for the presence of the node in the network. Binding different services to different addresses could mitigate unexpected discovery. Binding services to non-global addresses (e.g. link-local addresses or ULAs) could mitigate availability outside the expected scope. However, explicitly managing addresses adds significant complexity to the application development. It requires that application developers master addressing architecture subtleties, and implement logic that reacts adequately to connectivity events and address changes. Experience shows that application developers would probably prefer some much simpler solution.

In addition, we note that many application developers use high level APIs that listen to TLS, HTTP, or some other application protocol. These high level APIs seldomly provide detailed access to specific IP addresses, and typically default to listening to all available addresses.

A more advanced API could allow application programmers to select desired properties in an address (scope, stability, etc.), such that the best-suitable addresses are selected, while relieving the

application from low-level IPv6 addressing details. Such API could also trigger the generation of new IPv6 addresses if/when the specified properties required so.

7.2. Universal Support of Multi-prefix/Multi-router Networks

To put it bluntly, multi-prefix/multi-router networks cannot possibly work properly without implementation of [RFC8028]. Unfortunately, [RFC8028] is not widely implemented. On the protocol standardization side, the IETF should consider elevating the requirement to support RFC8028 in the IPv6 Node Requirements RFC [RFC8504] from "SHOULD" to "MUST".

7.3. Profile-based IPv6 Address Configuration

Most operating systems configure the same type of addresses regardless of the current "operating mode" or "profile" of the device (e.g., device connected to an enterprise network vs roaming across untrusted networks). For example, many operating systems configure

both stable [RFC8064] and temporary [RFC4941] addresses for all network interfaces. However, this "one size fits all" approach tends to be sub-optimal or inappropriate for some scenarios. For example, enterprise networks typically prefer usage of only stable addresses, thus meaning that a network administrator needs to find the means for disabling the generation of temporary addresses on all those systems that would otherwise generate them. On the other hand, some mobile devices normally configure both stable and temporary addresses, even when their usage type (client-like operation) would allow for the more privacy-sensible option of configuring only temporary addresses.

The lack of better-tuned address configuration policies has helped establish the "one size fits all" approach that, as noted, usually leads to suboptimal results. Advice in this area might help achieve more optional address configuration policies such that IPv6 addressing capabilities are fully leveraged.

NOTE:
One might envision a document that provides advice regarding the address generation for different typical scenarios (e.g., when to configure stable-only, temporary-only, or stable+temporary). In the most simple analysis, one might expect nodes in a typical enterprise network to employ only stable addresses. General-purpose nodes in a home or "trusted" network might want to employ both stable and temporary addresses. Finally, mobile nodes (e.g. when roaming across non-trusted networks) might want to employ only temporary addresses).

7.4. Protocol Improvements to Deal with Many Addresses

Possible improvements to IPv6 SLAAC should be evaluated, including:

- Enabling IPv6 routers to convey information about network constraints such as maximum number of addressees per node
- Enabling hosts to register/de-register configured addresses, such that e.g. routers need not tie resources to addresses that are no longer used

If a /64 prefix is to be assigned for each host in order to leverage

IPv6 address availability while mitigating the possible effects on network elements of employing large numbers of addresses, widespread support for DHCPv6-PD (or some proposed alternative mechanism) should be considered.

7.5. Support for Firewall Traversal in CE Routers

Customer Edge Routers that implement a default filtering policy of "only allowing outgoing communications" need to support helper protocols such as [UPnP] or PCP [RFC6887], so that applications can punch holes in the CE Router firewall for applications that need to receive incoming communications. Otherwise, P2P applications that currently work in IPv4 will not function properly in IPv6-only networks.

Support for these protocols is particularly important for IPv6 deployments since, as hosts will normally employ "provider aggregatable" addresses (see Section 4.2), renumbering events will result in host address changes, and thus static firewall rules will become invalid/outdated. Additionally, use of temporary addresses [RFC4941] will also lead to changing IPv6 addresses, which will require that the associated firewall rules be updated.

7.6. Advice on IPv6 Address Usage

An application programmer, left with the question of which are the most appropriate addresses for a given usage type and application, typically resorts to the Default IPv6 Address Selection for IPv6 (see Section 5.1) for outgoing communications, and to accepting incoming communications on all available addresses for incoming communications. As discussed throughout this document, this leads to sub-optimal results. Besides, all applications on a node share the same pool of configured addresses, and applications are also prevented from triggering the generation of new addresses (e.g. to be employed for a particular application or communication instance).

Guidance in this area is warranted such that applications and systems fully-leverage IPv6 addressing.

NOTE:
Such guidance would elaborate, among other things, on the usage of

IPv6 addresses when offering network services and when performing client-like communications. For example, for incoming communications, hosts might want to employ only the smallest-scope applicable addresses (if available) and, if stable addresses are available, they might want to accept incoming connections only on such addresses (but *not* on temporary addresses). For client-like communications, hosts might prefer temporary addresses, unless the corresponding communication instances are expected to be long-lived (e.g., SSH sessions).

8. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

9. Security Considerations

The security and privacy implications associated with the predictability and lifetime of IPv6 addresses has been analyzed in [RFC7217] [RFC7721], and [RFC7707]. This document complements and extends the aforementioned analysis by considering other IPv6 properties such as the address scope and address reachability, and the associated trade-offs.

10. Acknowledgements

The authors would like to thank (in alphabetical order) Mikael Abrahamsson, Fred Baker, Owen DeLong, Francis Dupont, Tatuya Jinmei, and Dave Thaler for providing valuable comments on earlier versions of this document.

11. References

11.1. 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>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/rfc2827>.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <https://www.rfc-editor.org/info/rfc4007>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <https://www.rfc-editor.org/info/rfc4193>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <https://www.rfc-editor.org/info/rfc6724>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[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, <https://www.rfc-editor.org/info/rfc7217>.

[RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi, "Host Address Availability Recommendations", BCP 204, RFC 7934, DOI 10.17487/RFC7934, July 2016, <https://www.rfc-editor.org/info/rfc7934>.

[RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <https://www.rfc-editor.org/info/rfc8028>.

[RFC8064] Gont, F., Cooper, A., Thaler, D., and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers", RFC 8064, DOI 10.17487/RFC8064, February 2017, <https://www.rfc-editor.org/info/rfc8064>.

[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>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[RFC8504] Chown, T., Loughney, J., and T. Winters, "IPv6 Node Requirements", BCP 220, RFC 8504, DOI 10.17487/RFC8504, January 2019, <https://www.rfc-editor.org/info/rfc8504>.

11.2. Informative References

[Barnes2012]
Barnes, R., Altmann, R., and D. Kerr, "Mapping the Great Void Smarter scanning for IPv6", ISMA 2012 AIMS-4 - Workshop on Active Internet Measurements, February 2012, <https://www.caida.org/workshops/isma/1202/slides/aims1202_rbarnes.pdf>.

[Hein] Hein, B., "The Rising Sophistication of Network Scanning", January 2016, <http://netpatterns.blogspot.be/2016/01/the-rising-sophistication-of-network.html>.

[I-D.gont-opsawg-firewalls-analysis]
Gont, F. and F. Baker, "On Firewalls in Network Security", draft-gont-opsawg-firewalls-analysis-02 (work in progress), February 2016.

[I-D.ietf-6man-rfc4941bis]
Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", draft-ietf-6man-rfc4941bis-12 (work in progress), November 2020.

[I-D.ietf-6man-slaac-renum]
Gont, F., Zorz, J., and R. Patterson, "Improving the Robustness of Stateless Address Autoconfiguration (SLAAC) to Flash Renumbering Events", draft-ietf-6man-slaac-renum-01 (work in progress), August 2020.

[I-D.ietf-mboned-ieee802-mcast-problems]
Perkins, C., McBride, M., Stanley, D., Kumari, W., and J. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", draft-ietf-mboned-ieee802-mcast-problems-12 (work in progress), October 2020.

[I-D.ietf-v6ops-cpe-slaac-renum]
Gont, F., Zorz, J., Patterson, R., and B. Volz, "Improving the Reaction of Customer Edge Routers to Renumbering Events", draft-ietf-v6ops-cpe-slaac-renum-05 (work in progress), September 2020.

[I-D.ietf-v6ops-dhcpv6-slaac-problem]

Liu, B., Jiang, S., Gong, X., Wang, W., and E. Rey, "DHCPv6/SLAAC Interaction Problems on Address and DNS Configuration", draft-ietf-v6ops-dhcpv6-slaac-problem-07 (work in progress), August 2016.

[I-D.ietf-v6ops-slaac-renum]
Gont, F., Zorz, J., and R. Patterson, "Reaction of Stateless Address Autoconfiguration (SLAAC) to Flash-Renumbering Events", draft-ietf-v6ops-slaac-renum-05 (work in progress), November 2020.

[I-D.ietf-v6ops-ula-usage-considerations]
Liu, B. and S. Jiang, "Considerations For Using Unique Local Addresses", draft-ietf-v6ops-ula-usage-considerations-02 (work in progress), March 2017.

[RFC5887] Carpenter, B., Atkinson, R., and H. Flinck, "Renumbering Still Needs Work", RFC 5887, DOI 10.17487/RFC5887, May 2010, <https://www.rfc-editor.org/info/rfc5887>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <https://www.rfc-editor.org/info/rfc6296>.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <https://www.rfc-editor.org/info/rfc7039>.

[RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <https://www.rfc-editor.org/info/rfc7707>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <https://www.rfc-editor.org/info/rfc7721>.

[RFC8190] Bonica, R., Cotton, M., Haberman, B., and L. Vegoda,

"Updates to the Special-Purpose IP Address Registries", BCP 153, RFC 8190, DOI 10.17487/RFC8190, June 2017, <https://www.rfc-editor.org/info/rfc8190>.

[UPnP] UPnP, "UPnP Device Architecture 2.0", April 17, 2020, <https://openconnectivity.org/upnp-specs/UPnP-arch-DeviceArchitecture-v2.0-20200417.pdf>.

Authors' Addresses

Fernando Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Email: fgont@si6networks.com
URI: https://www.si6networks.com

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Email: ggont@si6networks.com
URI: https://www.si6networks.com