Workgroup:
IPv6 Operations Working Group (v6ops)
Internet-Draft:
draft-gont-v6ops-ipv6-addressing-considerations-02
Published: 1 June 2022
Intended Status: Informational
Expires: 3 December 2022
Authors: F. Gont
SI6 Networks
G.G. Gont
SI6 Networks

IPv6 Addressing Considerations

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, with the goal of providing guidance about IPv6 address usage. 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 3 December 2022.

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 Stateless Address Auto-configuration (SLAAC) [RFC4862] prefix for each network interface. However, this static policy for address configuration might be inappropriate. For example, mobile nodes might benefit from employing only temporary addresses, which generally offer better privacy properties than stable addresses. On the other hand, an enterprise network might prefer that local hosts employ only stable addresses, which might be more convenient when enforcing access control, performing network trouble-shooting, or performing legitimate network activity correlation when e.g. hosts become infected by malware.

Additionally, each application on a given host could have its own set of requirements or expectations for the underlying IPv6 addresses. For example, an application meaning to offer a public service might expect to employ addresses that are both stable [RFC7721] [RFC8064] and globally-reachable [RFC8190], while a privacy-sensible client application might prefer short-lived temporary addresses [RFC8981], or might even expect to employ single-use ("ephemeral") IPv6 addresses when connecting to public servers. However, the subtleties associated with IPv6 address usage and with IPv6 addresses themselves are often ignored or overlooked by application programmers. This means that applications could fail to signal their requirements and preferences to the underlying host, or that the addresses configured by the underlying host might be inappropriate to satisfy the requirements of the corresponding applications.

Finally, a number of limitations in components that range from network devices to Application Programming Interfaces (APIs) could also prevent hosts and applications from leveraging the increased flexibility of IPv6 addressing.

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. Additionally, it identifies 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 different 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 [RFC8981], that mitigate the aforementioned issues. Therefore, this document does not discuss issues arising from legacy IID generation algorithms.

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

3.2. Address Scope

[RFC4007] defines the scope of an address as:

> "[the] topological span within which the address may be used as a unique identifier for an interface or set of interfaces"

And defines the "global scope" to be used for:

> "uniquely identifying interfaces anywhere in the Internet"

However, the term "scope" is employed in conflicting ways in different specifications (see [I-D.gont-6man-ipv6-ula-scope]). Throughout this document, we employ the notion of "scope" defined in [RFC4007]. As a result, addresses that do not uniquely identify interfaces Internet-wide are considered to have "non-global" or "limited" scope. Grouping addresses in such a way is simply useful for the purpose of discussing address properties.

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 topological span where an address can be expected to uniquely identify an interface; i.e., the topological span 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 topological span where an address can be expected to be used for receiving and transmitting packets. Reachability is implicitly constrained by the address scope, and may also 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 depends on the local policy of a node (e.g. stable vs. temporary addresses), it may also be constrained by other properties and external factors, such as provider dependency: if a network employs a prefix that is assigned/leased by an upstream provider, then the overall stability an address will also depend on the stability corresponding network prefix.

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 an upstream provider and carved out from the provider's address space, and thus is topologically-related to the upstream provider's address space; on the other hand, "provider independent" address space is "owned" by the subnet in question and thus is not necessarily topologically-related to the upstream provider.

4.1. Address Scope Considerations

The IPv6 address scope [RFC4007] has a direct implication on address reachability: the address scope essentially constrains address reachability. For example, addresses that have a non-global/limited 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 with the "external realm" via globally-reachable addresses (i.e., the middle-box provides an interface between two topological spans).

The IPv6 address scope can, in some scenarios, limit the attack exposure of a node as a result of the implicit isolation provided by non-global/limited 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 usable by only a limited number of applications/protocols that operate on a limited scope (e.g., mDNS), or deployments where the intended participants may be known to operate in a limited domain [RFC8799] (e.g., OpenSSH client and server attached to the same link and employing link-local addresses, or mDNS hosts 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 locally-generated ULA prefix will be, by definition, provider independent. However, a router might also employ a ULA prefix leased by an upstream router, in which case this prefix would be "provider dependent". The possible implications of the address scope on "provider dependency" may also affect address stability: for example, a locally-generated ULA prefix is "provider independent", and will not be subject to renumbering events triggered by the upstream provider. However, a router (e.g. CE Router) might, in some circumstances, be unable to guarantee prefix stability -- as in the case where the locally-generated ULA prefix is not recorded on stable storage, and thus cannot be guaranteed to remain stable across power outages (see [RFC9096] for more details).

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 agreggatable") 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 an upstream provider and carved out from the provider's address space, and thus is topologically-related to the upstream provider's address space; on the other hand, "provider independent" address space is "owned" by the network in question and thus is not necessarily topologically-related to the upstream provider.

An implicit consequence of PA address space is that its use is tied to the specific provider/upstream provider that provides the address space. This has a number of consequences, including:

* Multi-homing (employing local address space with multiple upstream providers) may not be possible.

* A renumbering event at the upstream provider will typically cause the local network to be renumbered.

Some organizations have opted to employ NPT [RFC6296] such that:

* The local network is isolated of renumbering events caused by the upstream provider.

* The local network employs the same address space regardless of the upstream provider employed to communicate with the external realm.

While PA space may impact address stability, PI address space generally has better stability properties. For example, a home network could internally employ both ULAs and GUAs, where a ULA prefix is locally generated by the CE Router (and hence resulting in PI space), and a global prefix is leased by the ISP via DHCPv6 Prefix Delegation [RFC8415] (hence PA space). If for some reason there was an outage involving the connection with the upstream ISP, the lease time for the GUA prefix 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 GUA prefix (rather than renew the existing prefix), the network would need to be renumbered. On the other hand, locally-generated ULA prefixes can be employed independently from the upstream ISP.

Similarly, an organizational network that employs PI global address space obtained from a RIR would be able to employ the same address space irrespective of renumbering events or outages involving the upstream provider. However, if dynamic sub-prefixes were delegated via DHCPv6-PD within the corresponding organization, such sub-address-space would be considered "provider dependent" from the perspective of such leaf networks.

4.3. Address Reachability

Address reachability represents the area of the network (and the associated conditions), where an address can be used for receiving and transmitting packets. As noted in Section 4.1, the address scope has a direct implication on address reachability, since it constrains the network span where the address is reachable.

In addition to the inherent 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 successfully 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 protocols 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 protocols 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 can be specifically devoted to administer address reachability.

4.4. Address Stability Considerations

Address stability typically depends on two factors:

- *Stability of the network prefix
- *Stability of the associated interface identifier (IID)

Depending on whether the local prefix is PI or PA (see Section 4.2) and whether the prefix is stable or dynamic (see [RFC8978]), the resulting addresses will have different stability properties. Additionally, even in the presence of stable prefixes, a host may use stable and/or temporary IIDs, thus resulting in stable addresses [RFC8064] and/or temporary addresses [RFC8981].

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 remove) 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, the longer an address is employed the longer the window of exposure of a host (via an address that becomes revealed as a result of active communication). While such exposure is typically 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" [RFC8981]. Temporary addresses are typically employed along with stable addresses, with the temporary addresses employed for outgoing communications, and the stable addresses employed for incoming communications.

NOTE:
That latest revision of the "temporary addresses" RFC ([RFC8981]) allows the configuration and use of only temporary addresses (i.e., removes the requirement 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 of interest. However, temporary addresses do limit the window of exposure to network-based attacks (including that of network activity correlation).

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 temporary addresses or, even better, generate one new IPv6 address for each application or new transport protocol instance (sometimes referred to as "ephemeral addresses"). However, reduced address lifetimes and the use of multiple IPv6 addresses may have a negative impact on the network (please see Section 6.3).

Enforcing a maximum lifetime (versus "preferred 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. Traditionally, many application protocols have assumed or expected address stability. However, in the light of mobile roaming nodes that may frequently switch among different connections (e.g. Wi-Fi, 4G, etc.) or that may be subject to renumbering events (see [RFC8978]), robust applications should assume and expect "ephemeral" IPv6 addresses (i.e., gracefully handle the case where the underlying IPv6 addresses change over short periods of time).

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

Finally, we note that if a different single-use (i.e., "ephemeral") IPv6 address is employed for each transport protocol instance, the possibility of an attacker successfully performing off-path attacks (such as the TCP reset attacks discussed in [RFC4953]) is reduced, since the ephemeral IPv6 address will typically be unknown and unpredictable to the off-path attacker.

5. IPv6 Address Usage

5.1. Default IPv6 Address Selection

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

Default Address Selection for IPv6 is specified in [RFC6724], and only applies for outgoing connections, such as those made by clients trying to use services offered by other hosts. The selection starts with a set of potential destination addresses, such as returned by getaddrinfo(3), 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.

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.

NOTE:

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 (see Section 4.4 for further details).

When a host provides a service, the common pattern is to just wait for incoming connections over all configured addresses. For example, applications using the BSD Sockets API will commonly bind(2) the listening socket to the undefined address. This long-established behavior is appropriate for hosts providing public services, but can have unexpected results for hosts providing semi-private services, such as various forms of peer-to-peer or local-only applications (e.g. mDNS).

This behavior leads to three problems: host tracking, discussed in Section 6.2.2; unexpected address discovery, discussed in Section 6.2.3; and availability outside the expected scope, discussed in Section 6.2.4. These problems are caused in part by the limitations of available address selection APIs, as discussed in Section 7.4.

5.2. Usage Type Considerations

IPv6 hosts may configure stable [RFC8064] and/or temporary [RFC8981] addresses, where stable addresses are typically employed for incoming (server-like) communications, and temporary addresses are employed for outgoing (client-like) communications. That is, the stability properties of an address have an implicitly associated usage type.

A host that employs one of its addresses to communicate with a remote server (i.e., that performs an "outgoing connection") will expose that address to the target server (and to on-path nodes). Once the remote server receives an incoming connection, it could readily launch an attack against the host via the exposed 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, nodes that learn about a client's addresses could not really leverage such addresses for actively contacting clients. Unfortunately, current APIs represent a challenge when trying to leverage IPv6 addresses in this way (please see Section 5.2.1 and Section 7.4 for further details).

The following subsections discuss possible techniques that could be employed by applications to better leverage IPv6 addresses for both incoming and outgoing communications

5.2.1. Incoming communications

There are a number of ways in which a system or network may affect which addresses may be employed (and how) 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 7.4) prevent application programmers from 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 the underlying addresses and their 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 would still be possible, since the aforementioned filtering would be performed once UDP packets have been received or TCP connections have been established.

A client could simply run a host-based firewall that only allows incoming connections on the stable addresses. This would be 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 some scenarios, a network-based firewall could be configured to allow outgoing communications from all internal addresses, but only allow incoming communications to stable addresses (either via manual configuration 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.

5.2.2. 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, let alone checking the properties of such addresses. Secondly, as discussed in Section 5.2.1, 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 Associated with IPv6 Addressing

The following subsections discuss current problems associated with IPv6 addresses, namely:

- *Sub-optimal Address Configuration (Section 6.1)
- *Sub-optimal IPv6 Address Usage (Section 6.2)
- *Operational Problems (Section 6.3)

6.1. Sub-optimal Address Configuration

6.1.1. Number of Addresses

Two mechanisms exist for automatic network configuration: SLAAC [RFC4862] and DHCPv6 [RFC8415]. 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 result in network configuration anarchy, where hosts may e.g. configure and use addresses in a way that may negatively affect the network (please see Section 6.3.2).

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 [RFC8415]. However, support for such mechanisms in host implementations and in the LAN-side of CE Routers is rather uncommon. On the other hand, SLAAC lacks the means for conveying information about e.g., the number of addresses per host that the network is able or willing to support.

NOTE:
Use of a /64 prefix per host could also render techniques such as temporary addresses [RFC8981] ineffective, since hosts would become identified by corresponding /64 prefix.

6.1.2. SLAAC/DHCPv6 Interaction

Many CE Routers offer address configuration via both SLAAC and DHCPv6, by including Prefix Information Options (PIOs) with the "A" flag set in Router Advertisement messages, and also setting the "M" flag in such RA 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 may end up employing predictable addresses resulting from DHCPv6, thus thwarting the security and privacy improvements of SLAAC-configured addresses (i.e., [RFC7217] and [RFC8981]).

6.2. Sub-optimal IPv6 Address Usage

6.2.1. 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 in the case of 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, as discussed in Section 5, this is practically impossible achieve with currently-available APIs.

6.2.2. 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 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.

6.2.3. Unintended Service Disclosure

Systems like DNS-Based Service Discovery [RFC6763] allow clients to discover services within a limited domain (e.g. a local link). These services are not advertised outside of that domain, and thus are not expected to be discovered by random parties on the Internet. However, such services may be easily discoverable if they allow

incoming connections on IPv6 addresses that client processes also use when connecting to remote servers.

NOTE:

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

Local services may be disclosed if 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 host operates both a video game server and a home automation application server. 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, its address will be revealed to all these users, thus increasing 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 avoiding the scenarios discussed in this section altogether.

6.2.4. Availability of Service Outside the Expected Domain

IPv6 defines multiple address scopes [RFC4291] [RFC4007], with hosts typically configuring Global Unicast Addresses (GUAs), link local addresses, and Unique Local IPv6 Unicast Addresses (ULAs) [RFC4193]. Availability of a service outside the expected scope happens when a service is expected to be available only in some limited domain, but it inadvertently becomes available from outside of that domain. This could happen, for example, if a service is meant to be accessible only within a given link, but becomes reachable from outside that link via ULAs or GUAs, or if a service is meant to be accessible only within some organization's perimeter but becomes accessible from the public Internet via GUAs. This will commonly happen if a service intended for a limited domain is implemented by bind()ing the listening socket to the "unspecified" addresses (please see Section 7.4).

6.3. Operational Problems

6.3.1. Implications on Firewall Rules and Access Control Lists (ACLs)

Simple firewall rules have traditionally been specified in terms of the associated IP addresses and transport protocol port numbers, generally implying that the associated IP addresses are stable. In the IPv4 world, IP addresses may be considered rather stable. However, this is generally not the case with IPv6 addresses, which tend to be less stale than IPv4 addresses. This may prevent the enforcement of filtering policies based on specific IPv6 addresses, or may lead to filtering based on a more coarse granularity (e.g. on specific address prefixes, as opposed to specific IPv6 addresses). In some scenarios, it may also encourage disabling features such as IPv6 temporary addresses [RFC8981].

NOTE:

In some scenarios, from the point of view of enforcing filtering policies, it might be desirable to disable temporary addresses altogether, whether at the system level or at the application level (if possible). For example, an administrator might prefer that a secondary DNS server performing DNS zone transfers, or an MTA, always employ the same source IPv6 address, as opposed to the different temporary addresses over times [I-D.gont-opsawg-firewalls-analysis].

6.3.2. Implications on Network Infrastructure

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. [RFC9119]). 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.

6.3.3. 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. 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.

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

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

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

6.3.5. Renumbering

The challenges posed by network renumbering have been known for a very long time [RFC5887], with network renumbering typically being assumed to be performed in a planned manner.

However, in scenarios where a host is moved to a different network without the host detecting the network re-attachment event, or where the network a host attaches to is moved to a different point of the network topology (i.e., the network itself is migrated/"moved"), the aforementioned host will also experience a renumbering event [RFC8978]. In an era in which migrating 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 (such that stale prefixes and addresses can 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 of 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, container or network that move around the

network topology, the network might not be able to signal the "renumbering event", and these events might be harder to infer and react to.

Unfortunately, both SLAAC and DHCPv6 assume that network configuration information is 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 operates on the same premise, and lacks widespread support for RECONFIGURE messages -- so even if the network were in a position to signal a renumbering event, hosts will normally rely on expiration of lease times for stale information to be cleared up.

Some of these problems have been discussed in detail in [RFC8978], and there have been a number of efforts (see [I-D.ietf-6man-slaac-renum] and [RFC9096]) to mitigate this issue.

7. Current Gaps that Prevent Leveraging IPv6 Addressing

The following subsections identify and discuss areas where further work is needed. Namely,

- *Profile-based IPv6 Address Configuration (see Section 7.1)
- *Advice on IPv6 Address Usage (see Section 7.2)
- *Protocol Improvements to Deal with Many Addresses (see Section 7.3)
- *Improved Address Selection APIs (see Section 7.4)
- *Universal Support of RFC 8028 (see Section 7.5)
- *Support for Firewall Traversal in CE Routers (see Section 7.6)

7.1. 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 [RFC8981] addresses for all network types. However, this "one size fits all" approach tends to be sub-optimal or even inappropriate for some scenarios. For example, enterprise networks typically prefer the use of only stable addresses, thus requiring the network administrator to configure each host to disable the use of temporary addresses. On the other hand, mobile devices typically configure both stable and temporary addresses, even when their operating mode (client-like

operation) would allow for the more privacy-sensible option of configuring only temporary addresses.

The lack of fine-grained address configuration policies forces nodes to rely on a "one size fits all" approach that, as noted, usually leads to suboptimal results. Advice in this area might help achieve profile-based address configuration policies such that IPv6 addressing capabilities are fully leveraged.

NOTE:

One might envision a document that provides advice regarding IPv6 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).

7.2. Advice on IPv6 Address Usage

Application programmers typically rely on the Default Source IPv6 Address Selection for IPv6 (see Section 5.1) for selected source addresses for outgoing communications, and on accepting incoming communications on any of the configured addresses. As discussed throughout this document, this leads to sub-optimal or undesirable results. All applications on a node share the same pool of configured addresses, and currently available APIs prevent applications from requesting 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 can fully leverage IPv6 addressing.

NOTE:

Such guidance would elaborate, among other things, on the usage of IPv6 addresses for incoming communications and for outgoing communications. For example, for incoming communications, hosts might want to employ only the smallest-scope applicable addresses (if available) and, if stable addresses were available, only accept incoming connections on such addresses. For outgoing communications, hosts might prefer temporary addresses, unless the corresponding communication instances are expected to be long-lived (e.g., SSH sessions).

7.3. 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.

On the other hand, in order for DHCPv6-PD (or some alternative protocol) to be employed to support the "one /64 per node" paradigm, widespread support for DHCPv6-PD (or an alternative protocol) would be necessary.

7.4. Improved 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 6.2. 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-globally-reachable addresses (e.g. link-local addresses or ULAs) could mitigate availability outside the expected domain. However, explicitly managing addresses adds significant complexity to application development. It requires that application developers master IPv6 addressing architecture subtleties, and implement logic that reacts adequately to connectivity events and address changes. Experience shows that application developers would probably prefer a 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 IPv6 addresses, and typically default to listening on 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 require so.

7.5. Universal Support of RFC 8028

To put it bluntly, multi-prefix/multi-router networks cannot possibly work properly without implementation of [RFC8028]. Unfortunately, [RFC8028] is not yet 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".

7.6. Support for Firewall Traversal in CE Routers

Customer Edge (CE) 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 open holes in the CE router firewall to be able to receive incoming communications. Otherwise, P2P applications that currently work in IPv4 networks might not function 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 be harder to implement than for the IPv4 networks. Similarly, use of only temporary addresses [RFC8981] would require that incoming connections be accepted on temporary addresses, thus requiring that the associated firewall rules be updated.

NOTE:

One might argue that if a node is to receive incoming connections, both stable and temporary addresses should be configured, though. Thus, firewall rules to allow incoming connections would be configured for the stable addresses rather than for the temporary addresses.

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

The security and privacy implications associated with the IPv6 addresses have been analyzed in [RFC7217] [RFC7721], and [RFC7707]. This document complements and extends the aforementioned analysis by also considering other IPv6 properties such as 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, Brian Carpenter, Owen DeLong, Francis

Dupont, Tatuya Jinmei, Ted Lemon, 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>.

[RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <https://www.rfc-editor.org/info/rfc8981>.

[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>.

[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>.

[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>.

[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

[RFC9119] Perkins, C., McBride, M., Stanley, D., Kumari, W., and JC. Zúñiga, "Multicast Considerations over IEEE 802 Wireless Media", RFC 9119, DOI 10.17487/RFC9119, October 2021, <https://www.rfc-editor.org/info/rfc9119>.

[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", Work in Progress, Internet-Draft, draft-ietf-6man-slaac-renum-02, 19 January 2021, <https://www.ietf.org/archive/id/draft-ietf-6man-slaac-renum-02.txt>.

[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>.

[RFC8978] Gont, F., Žorž, J., and R. Patterson, "Reaction of IPv6 Stateless Address Autoconfiguration (SLAAC) to Flash-Renumbering Events", RFC 8978, DOI 10.17487/RFC8978, March 2021, <https://www.rfc-editor.org/info/rfc8978>.

[RFC9096] Gont, F., Žorž, J., Patterson, R., and B. Volz, "Improving the Reaction of Customer Edge Routers to IPv6 Renumbering Events", BCP 234, RFC 9096, DOI 10.17487/RFC9096, August 2021, <https://www.rfc-editor.org/info/rfc9096>.

[I-D.gont-6man-ipv6-ula-scope] Gont, F., "Scope of Unique Local IPv6 Unicast Addresses", Work in Progress, Internet-Draft, draft-gont-6man-ipv6-ula-scope-00, 5 January 2021, <https://www.ietf.org/archive/id/draft-gont-6man-ipv6-ula-scope-00.txt>.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <https://www.rfc-editor.org/info/rfc4953>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <https://www.rfc-editor.org/info/rfc8799>.

[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>.

[I-D.gont-opsawg-firewalls-analysis] Gont, F. and F. Baker, "On Firewalls in Network Security", Work in Progress, Internet-Draft, draft-gont-opsawg-firewalls-analysis-02, 4 February 2016, <https://www.ietf.org/archive/id/draft-gont-opsawg-firewalls-analysis-02.txt>.

[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", Work in Progress, Internet-Draft, draft-ietf-v6ops-dhcpv6-slaac-problem-07, 17 August 2016, <https://www.ietf.org/archive/id/draft-ietf-v6ops-dhcpv6-slaac-problem-07.txt>.

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

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

Authors' Addresses

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

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

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

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