Internet Engineering Task Force (IETF)
Request for Comments: 7695
Category: Standards Track
ISSN: 2070-1721

P. Pfister
B. Paterson
Cisco Systems
J. Arkko
Ericsson

November 2015

Distributed Prefix Assignment Algorithm

Abstract

This document specifies a distributed algorithm for dividing a set of prefixes in a manner that allows for automatic assignment of sub-prefixes that are unique and non-overlapping. Used in conjunction with a protocol that provides flooding of information among a set of participating nodes, prefix configuration within a network may be automated.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies a distributed algorithm for automatic prefix assignment. The algorithm provides a generic alternative to centralized (human- or software-based) approaches for network prefix and address assignment. Although it does not have to be configured to operate properly, it supports custom configuration by means of variable priority assignments, and can therefore be used in fully autonomic as well as configured networks. This document focuses on the algorithm itself and therefore context-specific considerations (such as the process of selecting a prefix value and length when making a new assignment) are out of scope.

The algorithm makes use of a flooding mechanism allowing participating nodes to advertise prefixes assigned to the links to which they are directly connected or for other purposes, e.g., for private assignment or prefix delegation. Advertising a prefix therefore serves two purposes. It is a claim that a prefix is in use, meaning that no other node may advertise an overlapping prefix (unless it has a greater priority). And, it is a way for other nodes to know which prefixes have been assigned to the links to which they are directly connected.

The algorithm is given a set of delegated prefixes and ensures that the following assertions are satisfied after a finite convergence period:

1. At most one prefix from each delegated prefix is assigned to each link.

2. Assigned prefixes are non-overlapping (i.e., an assigned prefix never includes another assigned prefix).

3. Assigned prefixes do not change in the absence of topology or configuration changes.

In the rest of this document, the two first conditions are referred to as the correctness conditions of the algorithm, while the third condition is referred to as its convergence condition.

Each assignment has a priority specified by the node making the assignment, allowing for custom assignment policies. When multiple nodes assign different prefixes from the same delegated prefix to the same link, or when multiple nodes assign overlapping prefixes (to the same link or to different links), the assignment with the greatest priority is kept and other assignments are removed.

The prefix assignment algorithm requires that participating nodes share information through a flooding mechanism. If the flooding mechanism ensures that all messages are propagated to all nodes within a given time window, the algorithm also ensures that all assigned prefixes used for networking operations (e.g., host configuration) remain unchanged, unless another node assigns an overlapping prefix with a higher assignment priority, or the topology changes and renumbering cannot be avoided.

2. Definitions

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC2119].

This document makes use of the following terminology. The terms defined here are ordered in such a way as to try to avoid forward references, and therefore are not sorted alphabetically.

Node: An entity executing the algorithm specified in this document and able to communicate with other Nodes using the Flooding Mechanism.

Flooding Mechanism: A mechanism allowing participating Nodes to reliably share information with all other participating Nodes.

Link: An object to which the distributed algorithm will assign prefixes. A Node may only assign prefixes to Links to which it is directly connected. A Link is either Shared or Private.

Shared Link: A Link to which multiple Nodes may be connected. Most of the time, a Shared Link is a multi-access link or point-to-point link, virtual or physical, requiring prefixes to be assigned to it.

Private Link: A Private Link is an abstract concept defined for the sake of this document. It allows Nodes to make assignments for their private use or delegation. For instance, every DHCPv6-PD [RFC3633] requesting router may be considered as a different Private Link.

Delegated Prefix: A prefix provided to the algorithm and used as a prefix pool for Assigned Prefixes.

Node ID: A value identifying a given participating Node. The set of identifiers MUST be strictly and totally ordered (e.g., using the alphanumeric order). The mechanism used to assign Node IDs, whether manual or automated, is out of scope for this document.

Flooding Delay: A value that MUST be provided by the Flooding Mechanism and SHOULD be a deterministic or likely upper bound on the information propagation delay among participating Nodes.

Advertised Prefix: A prefix advertised by another Node and delivered to the local Node by the Flooding Mechanism. It has an Advertised Prefix Priority and, when assigned to a directly connected Shared Link, is associated with that Shared Link.

Advertised Prefix Priority: A value that defines the priority of an Advertised Prefix received from the Flooding Mechanism or a published Assigned Prefix. Whenever multiple Advertised Prefixes are conflicting (i.e., overlapping or from the same Delegated Prefix and assigned to the same link), all Advertised Prefixes but the one with the greatest priority will eventually be removed. In case of a tie, the assignment advertised by the Node with the greatest Node ID is kept, and others are removed. In order to ensure convergence, the range of priority values MUST have an upper bound.

Assigned Prefix: A prefix included in a Delegated Prefix and assigned to a Shared or Private Link. It represents a local decision to assign a given prefix from a given Delegated Prefix to a given Link. The algorithm ensures that there is never more than one Assigned Prefix per Delegated Prefix and Link pair. When destroyed, an Assigned Prefix is set as not applied, ceases to be advertised, and is removed from the set of Assigned Prefixes.

Applied (Assigned Prefix): When an Assigned Prefix is applied, it MAY be used (e.g., for host configuration, routing protocol configuration, prefix delegation). When not applied, it MUST NOT be used for any purpose outside of the prefix assignment algorithm. Each Assigned Prefix is associated with a timer (Apply Timer) used to apply the Assigned Prefix. An Assigned Prefix is unapplied when destroyed.

Published (Assigned Prefix): The Assigned Prefix is advertised through the Flooding Mechanism as assigned to its associated Link. A published Assigned Prefix MUST have an Advertised Prefix Priority. It will appear as an Advertised Prefix to other Nodes, once received from the Flooding Mechanism.

Destroy (an Assigned Prefix): Local action of removing an Assigned Prefix from the set of Assigned Prefixes. If applied, the prefix is unapplied. If published, the prefix stops being advertised through the Flooding Mechanism.

Prefix Adoption: When an Advertised Prefix that does not conflict with any other Advertised Prefix or published Assigned Prefix stops being advertised, any other Node connected to the same Link may, after some random delay, start advertising the same prefix. This procedure is called adoption and provides seamless assignment transfer from a Node to another, e.g., in case of Node failure.

Backoff Timer: Every Delegated Prefix and Link pair is associated with a timer counting down to zero. By delaying the creation of new Assigned Prefixes or the advertisement of adopted Assigned Prefixes by a random amount of time, it reduces the probability of colliding assignments made by multiple Nodes.

Renumbering: Event occurring when an Assigned Prefix that was applied is destroyed. Renumbering is undesirable as it usually implies reconfiguring routers or hosts.

2.1. Subroutine-Specific Terminology

In addition to the terms defined in Section 2, the subroutine specified in Section 4 makes use of the following terms.

Current Assignment: For a given Delegated Prefix and Link, the Current Assignment is the Assigned Prefix (if any) included in the Delegated Prefix and assigned to the given Link by the Node executing the algorithm. At some point in time, the Current Assignment from different Nodes may differ, but the algorithm ensures that, eventually, all Nodes directly connected to a Shared Link have the same Current Assignment for any given Delegated Prefix.

Precedence: An Advertised Prefix takes precedence over an Assigned Prefix if and only if one of the following conditions is met:

* The Assigned Prefix is not published.

* The Assigned Prefix is published, and the Advertised Prefix Priority from the Advertised Prefix is strictly greater than the Advertised Prefix Priority from the Assigned Prefix.

* The Assigned Prefix is published, the priorities are identical, and the Node ID from the Node advertising the Advertised Prefix is strictly greater than the local Node ID.

Best Assignment: For a given Delegated Prefix and Link, the Best Assignment is computed as the unique Advertised Prefix (if any) that:

* Includes or is included in the Delegated Prefix (i.e., the Advertised Prefix is a sub-prefix of the Delegated Prefix, or the Delegated Prefix is a sub-prefix of the Advertised Prefix).

* Is assigned on the given Link.

* Has the greatest Advertised Prefix Priority among Advertised Prefixes fulfilling the two preceding conditions (and, in case of a tie, the prefix advertised by the Node with the greatest Node ID among all prefixes with greatest priority).

* Takes precedence over the Current Assignment associated with the same Link and Delegated Prefix (if any).

Valid (Assigned Prefix): An Assigned Prefix is valid if and only if the following two conditions are met:

* No Advertised Prefix including or included in the Assigned Prefix takes precedence over the Assigned Prefix.

* No Advertised Prefix including or included in the same Delegated Prefix as the Assigned Prefix and assigned to the same Link takes precedence over the Assigned Prefix.

3. Applicability Statement

Although the algorithm was primarily designed as an autonomic prefix assignment tool for home networks, it is applicable to other areas. In particular, it can operate without any kind of configuration as well as use advanced prefix assignment rules. Additionally, it can be applied to any address space and can be used to manage multiple address spaces simultaneously. For instance, an implementation can make use of IPv4-mapped IPv6 addresses [RFC4291] in order to manage both IPv4 and IPv6 prefix assignment using a single prefix space.

Each Node MUST have a set of non-overlapping Delegated Prefixes (i.e., that do not include each other). This set MAY change over time and be different from one Node to another at some point, but Nodes MUST eventually have the same set of non-overlapping Delegated Prefixes.

Given this set of non-overlapping Delegated Prefixes, Nodes may assign available prefixes from each Delegated Prefix to the Links they are directly connected to. The algorithm ensures that at most one prefix from a given Delegated Prefix is assigned to any given Link. Prefixes may also be assigned for private use. For example, an assigned prefix may be delegated to some other entity that does not implement this algorithm [RFC3633], or associated with a high priority in order to prevent other nodes from assigning any overlapping prefix [RFC6603].

The algorithm supports dynamically changing topologies and therefore will converge if the topology remains unmodified for a long enough period of time. (That time depends on the Flooding Mechanism properties.) Nevertheless, some topology changes may induce renumbering, while others do not. In particular, Nodes joining the set of participating Nodes do not cause renumbering. Similarly, Nodes leaving the network may be handled without renumbering by using the prefix adoption procedure. On the other hand, Links that merge or split may break correctness conditions, and therefore cause renumbering.

All Nodes MUST run a common Flooding Mechanism in order to share published Assigned Prefixes. The set of participating Nodes is defined as the set of Nodes participating in the Flooding Mechanism.

The Flooding Mechanism MUST:

o Provide a way to flood Assigned Prefixes assigned to a directly connected Link along with their respective Advertised Prefix Priority and the Node ID of the Node that is advertising them.

o Specify whether an Advertised Prefix is assigned to a directly connected Shared Link, and if so, which one. This information also needs to be updated in case of Links that merge or split.

o Provide a Flooding Delay value, which SHOULD represent a deterministic or likely upper bound on the information propagation delay among participating Nodes. Whenever the Flooding Mechanism is unable to adhere to the provided Flooding Delay, renumbering may happen. As such, a delay often depends on the size of the network, it MAY change over time and MAY be different from one Node to another. Furthermore, the process of selecting this value is subject to a tradeoff between convergence speed and lower renumbering probability (e.g., the value 0 may be used when renumbering is harmless), and is therefore out of scope for this document.

The algorithm ensures that whenever the Flooding Delay is provided and held, and in the absence of any topology change or Delegated Prefix removal, renumbering only happens when a Node deliberately overrides an existing assignment. In the absence of such deliberate override, the algorithm converges within an absolute worst-case timespan of (2 * Flooding Delay * L) seconds, where L is the number of links.

Each Node MUST have a Node ID. In the situation where multiple nodes have the same Node ID, the algorithm will not suffer, assuming there are no colliding assignments. However, in order for collisions to be resolved, that situation MUST be transient.

Finally, leaving the Flooding Mechanism or Node ID assignment process unsecured makes the network vulnerable to denial-of-service attacks, as detailed in Section 8. Additionally, as this algorithm requires all Nodes to know which Node has made which assignment, it may be unsuitable depending on privacy requirements among participating Nodes.

4. Algorithm Specification

This section specifies the behavior of Nodes implementing the prefix assignment algorithm. The terms ’Current Assignment’, ’Precedence’, ’Best Assignment’, and ’Valid’ are used as defined in Section 2.1.

4.1. Prefix Assignment Algorithm Subroutine

This section specifies the prefix assignment algorithm subroutine. It is defined for a given Delegated Prefix and Link pair and takes a BackoffTriggered boolean as parameter (indicating whether the subroutine execution was triggered by the Backoff Timer or by another event). The subroutine also makes use of the two following configuration parameters: ADOPT_MAX_DELAY and BACKOFF_MAX_DELAY, which are defined in Section 7.

For a given Delegated Prefix and Link pair, the subroutine MUST be run with the BackoffTriggered boolean set to false whenever:

- An Advertised Prefix including or included in the considered Delegated Prefix is added or removed.
- An Assigned Prefix included in the considered Delegated Prefix and associated with a different Link than the considered Link was destroyed, while there is no Current Assignment associated with the given pair. This case MAY be ignored if the creation of a new Assigned Prefix associated with the considered pair is not desired.
- The considered Delegated Prefix is added.
- The considered Link is added.
- The Node ID is modified.
- An Assigned Prefix included in the considered Delegated Prefix and associated with the considered Link is destroyed outside of the context of the subroutine, as specified in Section 4.2.

Furthermore, for a given Delegated Prefix and Link pair, the subroutine MUST be run with the BackoffTriggered boolean set to true whenever:

- The Backoff Timer associated with the considered Delegated Prefix and Link pair fires while there is no Current Assignment associated with the given pair.

When such an event occurs, a Node MAY delay the execution of the subroutine instead of executing it immediately, e.g., while receiving an update from the Flooding Mechanism, or for security reasons (see Section 8). Even if other events occur in the meantime, the subroutine MUST be run only once. It is also assumed that if one of these events is the firing of the Backoff Timer while there is no Current Assignment associated with the given pair, the subroutine is executed with the BackoffTriggered boolean set to true.

In order to execute the subroutine for a given Delegated Prefix and Link pair, first get the Current Assignment and compute the Best Assignment associated with the Delegated Prefix and Link pair, then execute the steps depending on the following cases:

1. If there is no Best Assignment and no Current Assignment: Decide whether the creation of a new assignment for the given Delegated Prefix and Link pair is desired. (As any result would be valid, the process of making this decision is out of scope for this document.) And, do the following:

    * If it is not desired, stop the execution of the subroutine.

    * Else if the Backoff Timer is running, stop the execution of the subroutine.

    * Else if the BackoffTriggered boolean is set to false, set the Backoff Timer to some random delay between ADOPT_MAX_DELAY and BACKOFF_MAX_DELAY (see Section 7) and stop the execution of the subroutine.

    * Else, continue the execution of the subroutine.

    Select a prefix for the new assignment (see Section 5 for guidance regarding prefix selection). This prefix MUST be included in or be equal to the considered Delegated Prefix and MUST NOT include or be included in any Advertised Prefix. If a suitable prefix is found, use it to create a new Assigned Prefix:

    * Assigned to the considered Link.

    * Set as not applied.

    * The Apply Timer set to (2 * Flooding Delay).

    * Published with some selected Advertised Prefix Priority.

2. If there is a Best Assignment but no Current Assignment: First, check if the Best Assignment is equal to or included in the Delegated Prefix. If not, stop the execution of the subroutine. Otherwise, cancel the Backoff Timer and use the prefix from the Best Assignment to create a new Assigned Prefix:

   * Assigned to the considered Link.

   * Set as not applied.

   * With the Apply Timer set to (2 * Flooding Delay).

   * Set as not published.

3. If there is a Current Assignment but no Best Assignment:

   * If the Current Assignment is not valid, destroy it, and execute the subroutine again with the BackoffTriggered boolean set to false.

   * If the Current Assignment is valid and published, stop the execution of the subroutine.

   * If the Current Assignment is valid and not published, the Node MUST either:

     + Adopt the prefix by canceling the Apply Timer and set the Backoff Timer to some random delay between 0 and ADOPT_MAX_DELAY (see Section 7). This procedure is used to avoid renumbering when the Node advertising the prefix left the Shared Link, and it SHOULD therefore be preferred.

     + Destroy it and go to case 1, allowing a different prefix to be assigned, or the prefix to be removed. When the Current Assignment is applied, this causes renumbering.

4. If there is a Current Assignment and a Best Assignment:

   * Cancel the Backoff Timer.

   * If the two prefixes are identical, set the Current Assignment as not published. If the Current Assignment is not applied and the Apply Timer is not set, set the Apply Timer to (2 * Flooding Delay).

   * If the two prefixes are not identical, destroy the Current Assignment and go to case 2.

When the prefix assignment algorithm subroutine requires an assignment to be created or adopted, any Advertised Prefix Priority value can be used. Other documents MAY provide restrictions over this value depending on the context in which the algorithm is operating or leave it as implementation specific.

4.2. Overriding and Destroying Existing Assignments

In addition to the behaviors specified in Section 4.1, the following procedures MAY be used in order to provide additional behavior options (Section 6).

Overriding Existing Assignments: For any given Link and Delegated Prefix, a Node MAY create a new Assigned Prefix using a chosen prefix and Advertised Prefix Priority such that:

* The chosen prefix is included in or is equal to the considered Delegated Prefix.

* The Current Assignment, if any, as well as all existing Assigned Prefixes that include or are included inside the chosen prefix are destroyed.

* It is not applied.

* The Apply Timer is set to (2 * Flooding Delay).

* It is published.

* The Advertised Prefix Priority is greater than the Advertised Prefix Priority from all Advertised Prefixes that include or are included in the chosen prefix.

* The Advertised Prefix Priority is greater than the Advertised Prefix Priority from all Advertised Prefixes that include or are included in the considered Delegated Prefix and are assigned to the considered Link.

In order to ensure algorithm convergence:

* Such overriding assignments MUST NOT be created unless there was a change in the Node configuration, a Link was added, or an Advertised Prefix was added or removed.

* The chosen Advertised Prefix Priority for the new Assigned Prefix SHOULD be greater than all priorities from the destroyed Assigned Prefixes. If not, simple topologies with only two Nodes may not converge. Nodes that do not adhere to this rule

MUST implement a mechanism that detects if the distributed algorithm does not converge and, when this occurs, they MUST stop creating overriding Assigned Prefixes that do not adhere to this rule. The specifications for such safety procedures are out of scope for this document.

Removing an Assigned Prefix: A Node MAY destroy any Assigned Prefix that is published. Such an event reflects the desire of a Node to not assign a prefix from a given Delegated Prefix to a given Link anymore. In order to ensure algorithm convergence, such a procedure MUST NOT be executed unless there was a change in the Node configuration. Furthermore, whenever an Assigned Prefix is destroyed in this way, the prefix assignment algorithm subroutine MUST be run for the Delegated Prefix and Link pair associated with the destroyed Assigned Prefix.

The two procedures specified in this section are OPTIONAL. They could be used for various purposes, e.g., for providing custom prefix assignment configuration or reacting to prefix space exhaustion (by overriding short Assigned Prefixes and assigning longer ones).

4.3. Other Events

When the Apply Timer fires, the associated Assigned Prefix MUST be applied.

When the Backoff Timer associated with a given Delegated Prefix and Link pair fires while there is a Current Assignment associated with the same pair, the Current Assignment MUST be published with some associated Advertised Prefix Priority and, if the prefix is not applied, the Apply Timer MUST be set to (2 * Flooding Delay).

When a Delegated Prefix is removed from the set of Delegated Prefixes (e.g., when the Delegated Prefix expires), all Assigned Prefixes included in the removed Delegated Prefix MUST be destroyed.

When one Delegated Prefix is replaced by another one that includes or is included in the deleted Delegated Prefix, all Assigned Prefixes that were included in the deleted Delegated Prefix but are not included in the added Delegated Prefix MUST be destroyed. Others MAY be kept.

When a Link is removed, all Assigned Prefixes assigned to that Link MUST be destroyed.

5. Prefix Selection Considerations

When the prefix assignment algorithm subroutine specified in Section 4.1 requires a new prefix to be selected, the prefix MUST be selected either:

- o Among prefixes included in the considered Delegated Prefix that were previously assigned and applied on the considered Link. For that purpose, Applied Prefixes may be stored in stable storage along with their associated Link.

- o Randomly, picked from a set of prefixes, where the set is of at least RANDOM_SET_SIZE (see Section 7). The prefixes are those included in the considered Delegated Prefix and not including or included in any Assigned or Advertised Prefix. If less than RANDOM_SET_SIZE candidates are found, the prefix MUST be picked among all candidates.

- o Based on some custom selection process specified in the configuration.

A simple implementation MAY randomly pick the prefix among all available prefixes, but this strategy is inefficient in terms of address space use as a few long prefixes may exhaust the pool of available short prefixes.

The rest of this section describes a more efficient approach that MAY be applied any time a Node needs to pick a prefix for a new assignment. The two following definitions are used:

Available prefix: The prefix of the form Prefix/PrefixLength is available if and only if it satisfies the three following conditions:

  * It is included in the considered Delegated Prefix.

  * It does not include and is not included in any Assigned or Advertised Prefix.

  * It is equal to the considered Delegated Prefix or Prefix/(PrefixLength-1) includes an Assigned or Advertised Prefix.

Candidate prefix: A prefix of desired length that is included in or is equal to an available prefix.

The procedure described in this section takes the three following criteria into account:

Prefix Stability: In some cases, it is desirable that the selected prefix should remain the same across executions and reboots. For this purpose, prefixes previously applied on the Link or pseudorandom prefixes generated based on Node- and Link-specific values may be considered.

Randomness: When no stored or pseudorandom prefix is chosen, a prefix may be randomly picked among RANDOM_SET_SIZE candidates of desired length. If less than RANDOM_SET_SIZE candidates can be found, the prefix is picked among all candidates.

Addressing-space usage efficiency: In the process of assigning prefixes, a small set of badly chosen long prefixes may prevent any shorter prefix from being assigned. For this reason, the set of RANDOM_SET_SIZE candidates is created from available prefixes with longest prefix lengths, and, in case of a tie, numerically small prefix values are preferred.

When executing the procedure, do as follows:

1. For each prefix stored in stable storage, check if the prefix is included in or equal to an available prefix. If so, pick that prefix and stop.

2. For each prefix length, count the number of available prefixes of the given length.

3. If the desired prefix length was not specified, select one. The available prefixes count computed previously may be used to help pick a prefix length such that:

   * There is at least one candidate prefix.

   * The prefix length is chosen large enough to not exhaust the address space.

   Let N be the chosen prefix length.

4. Iterate over available prefixes starting with prefixes of length N down to length 0 and create a set of RANDOM_SET_SIZE candidate prefixes of length exactly N included in or equal to available prefixes. The end goal here is to create a set of RANDOM_SET_SIZE candidate prefixes of length N included in a set of available prefixes of maximized prefix length. In case of a tie, smaller prefix values (as defined by the bit-wise lexicographical order) are preferred.

5. Generate a set of prefixes of desired length, which are pseudorandomly chosen based on Node- and Link-specific values. For each pseudorandom prefix, check if the prefix is equal to a candidate prefix. If so, pick that prefix and stop.

6. Choose a random prefix from the set of selected candidates.

The complexity of this procedure is equivalent to the complexity of iterating over available prefixes. Such operation may be accomplished in linear time, e.g., by storing Advertised and Assigned Prefixes in a binary tree.

6. Implementation Capabilities and Node Behavior

Implementations of the prefix assignment algorithm may vary from very basic to highly customizable, enabling different types of fully interoperable behaviors. The three following behaviors are given as examples:

Listener: The Node only acts upon assignments made by other Nodes, i.e, it never creates new assignments nor adopts existing ones. Such behavior does not require the implementation of the considerations specified in Section 4.2 or 5. The Node never checks the validity of existing assignments, which makes this behavior particularly suited to lightweight devices that can rely on more capable neighbors to make assignments on directly connected Shared Links.

Basic: The Node is capable of assigning new prefixes or adopting prefixes that do not conflict with any other existing assignment. Such behavior does not require the implementation of the considerations specified in Section 4.2. It is suited to situations where there is no preference over which prefix should be assigned to which Link, and there is no priority between different Links.

Advanced: The Node is capable of assigning new prefixes, adopting existing ones, making overriding assignments, and destroying existing ones. Such behavior requires the implementation of the considerations specified in Sections 4.2 and 5. It is suitable when the administrator desires some particular prefix to be assigned on a given Link, or some Link to be assigned prefixes with a greater priority when there are not enough prefixes available for all Links.

Note that if all Nodes directly connected to some Link are listener Nodes or none of these Nodes are willing to make an assignment from a given Delegated Prefix to the given Link, no prefix from the given

Delegated Prefix will ever be assigned to the Link. This situation may be detected by monitoring whether any prefix from a given Delegated Prefix has been assigned to the Link for longer than BACKOFF_MAX_DELAY plus the Flooding Delay.

7. Algorithm Parameters

This document does not provide values for ADOPT_MAX_DELAY, BACKOFF_MAX_DELAY, and RANDOM_SET_SIZE. The algorithm ensures convergence and correctness for any chosen values, even when these are different from Node to Node. They MAY be adjusted depending on the context, providing a tradeoff between convergence time, efficient addressing, control traffic (generated by the Flooding Mechanism), and collision probability.

ADOPT_MAX_DELAY represents the maximum backoff time a Node may wait before adopting an assignment; BACKOFF_MAX_DELAY represents the maximum backoff time a Node may wait before making a new assignment. BACKOFF_MAX_DELAY MUST be greater than or equal to ADOPT_MAX_DELAY. The greater ADOPT_MAX_DELAY and (BACKOFF_MAX_DELAY - ADOPT_MAX_DELAY), the lower the collision probability and the lesser the amount of control traffic, but the greater the convergence time.

RANDOM_SET_SIZE represents the desired size of the set from which a random prefix will be picked. The greater RANDOM_SET_SIZE, the better the convergence time and the lower the collision probability, but the worse the addressing-space usage efficiency.

8. Security Considerations

The prefix assignment algorithm functions on top of two distinct mechanisms, the Flooding Mechanism and the Node ID assignment mechanism.

An attacker able to publish Advertised Prefixes through the Flooding Mechanism may perform the following attacks:

* Publish a single overriding assignment for a whole Delegated Prefix or for the whole address space, thus preventing any Node from assigning prefixes to Links.

* Quickly publish and remove Advertised Prefixes, generating traffic at the Flooding Mechanism layer and causing multiple executions of the prefix assignment algorithm in all participating Nodes.

* Publish and remove Advertised Prefixes in order to prevent the convergence of the algorithm.

An attacker able to prevent other Nodes from accessing a portion or the whole set of Advertised Prefixes may compromise the correctness of the algorithm.

An attacker able to cause repetitive Node ID changes may cause traffic to be generated in the Flooding Mechanism and multiple executions of the prefix assignment algorithm in all participating Nodes.

An attacker able to publish Advertised Prefixes using a Node ID used by another Node may impede the ability to resolve prefix assignment collisions.

Whenever the security of the Flooding Mechanism and Node ID assignment mechanism cannot be ensured, the convergence of the algorithm may be prevented. In environments where such attacks may be performed, the execution of the prefix assignment algorithm subroutine SHOULD be rate limited, as specified in Section 4.1.

9. References

9.1. Normative References

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

9.2. Informative References

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <http://www.rfc-editor.org/info/rfc3633>.

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

[RFC6603] Korhonen, J., Ed., Savolainen, T., Krishnan, S., and O. Troan, "Prefix Exclude Option for DHCPv6-based Prefix Delegation", RFC 6603, DOI 10.17487/RFC6603, May 2012, <http://www.rfc-editor.org/info/rfc6603>.

Appendix A. Static Configuration Example

This section describes an example of how custom configuration of the prefix assignment algorithm may be implemented.

The Node configuration is specified as a finite set of rules. A rule is defined as:

- A prefix to be used.
- A Link on which the prefix may be assigned.
- An Assigned Prefix Priority (the smallest possible Assigned Prefix Priority if the rule may not override other Assigned Prefixes).
- A rule priority (0 if the rule may not override existing Advertised Prefixes).

In order to ensure the convergence of the algorithm, the Assigned Prefix Priority MUST be an increasing function (not necessarily strictly) of the configuration rule priority (i.e., the greater the configuration rule priority is, the greater the Assigned Prefix Priority must be).

Each Assigned Prefix is associated with a rule priority. Assigned Prefixes that are created as specified in Section 4.1 are given a rule priority of 0.

Whenever the configuration is changed or the prefix assignment algorithm subroutine is run, for each Link/Delegated Prefix pair, look for the configuration rule with the greatest configuration rule priority such that:

- The prefix specified in the configuration rule is included in the considered Delegated Prefix.
- The Link specified in the configuration rule is the considered Link.
- All the Assigned Prefixes that would need to be destroyed in case a new Assigned Prefix is created from that configuration rule (as specified in Section 4.2) have an associated rule priority that is strictly lower than the one of the considered configuration rule.
- The assignment would be valid when published with an Advertised Prefix Priority equal to the one specified in the configuration rule.

If a rule is found, a new Assigned Prefix is created based on that rule as specified in Section 4.2. The new Assigned Prefix is associated with the Advertised Prefix Priority and the rule priority specified in the considered configuration rule.

Note that the use of rule priorities ensures the convergence of the algorithm.

Acknowledgments

The authors would like to thank those who participated in the development of draft versions of this document as well as the present document. In particular, the authors would like to thank Tim Chown, Fred Baker, Mark Townsley, Lorenzo Colitti, Ole Troan, Ray Bellis, Markus Stenberg, Wassim Haddad, Joel Halpern, Samita Chakrabarti, Michael Richardson, Anders Brandt, Erik Nordmark, Laurent Toutain, Ralph Droms, Acee Lindem, Steven Barth, and Juliusz Chroboczek for interesting discussions and document review.

Authors’ Addresses

Pierre Pfister
Cisco Systems
Paris
France

Email: pierre.pfister@darou.fr

Benjamin Paterson
Cisco Systems
Paris
France

Email: paterson.b@gmail.com

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net