Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 31 May 2026

T. Przygienda
S. Hegde
Juniper Networks

27 November 2025

# Flooding Reduction Algorithms Framework
# draft-ietf-lsr-flood-reduction-arch-00

## Abstract

This document introduces a framework making it possible to deploy multiple flood reduction algorithms within the same IGP domain in an interoperable fashion.

## 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 31 May 2026.

Table of Contents



## 1. Introduction

Scenarios exist where multiple distributed (or centralized) flood reduction algorithms may be deployed simultaneously within an IGP domain. These scenarios necessitate certain agreed on cooperative behaviors between the involved algorithms to ensure the correctness of the overall solution. This is true in both permanent and transient (i.e., migration) deployment cases. Fortunately, existing graph theory concepts allow to provide guidance toward the design of algorithms with the necessary properties to ensure their interoperable coexistence.

This document presents the necessary requirements for the involved algorithms and the details of a framework for their interoperable deployment. Although running multiple algorithms simultaneously may not be a preferred operational choice, it is necessary if the migration from one algorithm to another with minimal network disruption is a priority. A migration itself may be caused by the discovery of defects in the deployed algorithms or the deployment of new algorithms that offer improvements.

Dealing with interoperability or lack thereof between this framework and other published frameworks such as e.g. [RFC9667] is explicitly outside the scope of this document.

## 2. Flooding Pruner Framework

## 2.1. Definitions and Axioms

This section outlines a framework that allows the operation of multiple different flood reduction algorithms (called _flooding pruners_ or _pruners_ from here on) in an interoperable fashion.

An important observation upfront, which will become clear later in this section, is that full, non-optimized flooding presents a special case of a pruner itself. Normal flooding includes all adjacencies without any pruning, and hence we name it the _non-pruner_ or _zero_ for short.

### 2.1.1. Maximum of One Flooding Pruner on a Node

This framework permits the use of at most one pruner on each node. It allows to change a specific pruner at any time on any subset of nodes in the network while limiting the impact to the node itself and possibly the re-convergence of a set of nodes within its connected component.

### 2.1.2. Connected Component

A _connected component_ (or component for short) is defined as a subset of all nodes in the network running the same pruner, e.g. A (such nodes are denoted by the set notation of A|) where each of the nodes has to be connected to all other nodes by a path that traverses only nodes that run A. Observe that there well may be in the network multiple components that are not connected, but that run the same pruner algorithm. We denote in such case a component for pruner A as A|, and if two _disjoint_ components running the same algorithm A are present in the network, we denote such second set as A|' and by extension we use A|'' notation for further such sets.

Non-pruners also build components denoted as Z| and its primes.

Another way to visualize components is to consider a network running multiple pruners as "islands running pruning algorithms" that are connected to each other by components running non-pruners (i.e. using normal flooding).

### 2.1.3. Flooding Connected Dominating Sets

A pruner may choose within its component a subset of links to flood while making sure that the component remains connected. In other words, after suppressing flooding on some links within the component there must still exist paths consisting of the remaining links that connect each pair of nodes in the component. We use for such remaining links the term _flooding connected dominating set_ or CDS

for short (more precisely, a not necessarily loop-free edge dominating set). Such a CDS is colloquially often called _flooding topology_ in context of flood reduction algorithms. A simple spanning tree is an easily visualized special case of a CDS. We denote such a CDS for a component A| as A|*. A|* is often not unique for a component and many different sets of links can be a CDS. Nor is it required that a CDS has to be loop-free since there may be many different paths on the CDS between two nodes in a component. Therefore, it is possible in a most extreme case that each LSP is flooded on a different CDS.

To summarize the section above in simple terms, a pruner must choose at least one set of flooding links that guarantees that all information can reach all the nodes in the component.

#### 2.1.4. Rules for Flooding Pruners

Any flood reduction algorithm expecting to interoperate with other algorithms within this framework but without having to understand their behavior MUST adhere to the following rules. Otherwise, the algorithm cannot be expected to accommodate other algorithms in the network at the same time or is in other words a ship in the night.

1. Each node of a pruner (except the non-pruner) MUST advertise in its flooded node information the currently active pruner. It MUST also understand such information as advertised by other nodes in the network. A node running a pruner MUST NOT assume implicitly that a node is a _non-pruner_ or supports or runs the same algorithm. However, any pruner can safely assume that any node that does not advertise any running pruner in its node information MUST be a non-pruner. Observe that a pruner does not need to understand how the algorithm of another pruner operates (or even whether it is centralized, centrally signalled or fully distributed). The only requirement is that every pruner uses the same signaling information provided in this framework which indicates the pruner currently running.

2. A pruner MUST NOT prune links in components other than the one it participates in or assume flooding behavior on links in other components (except in the case of a _non-pruner_ where the flooding is well understood). In other words, each pruner is allowed to prune some links from flooding, but only strictly within its own component.

3. A flooding pruner A MUST also include in its flooding CDS all links to adjacent components running a non _non-pruner_ different from A. A node running pruner P that is different from the _non-pruner_ SHOULD include in its flooding CDS all links to non-

pruners. It MAY use the known behavior the _non-pruner_ for further optimizations. Nevertheless, such optimizations MUST NOT assume that there is just a single Z| in the network. This is sufficient (but strictly speaking, more than necessary) to guarantee that the overall set of flooding CDSes within each component creates an overall flooding CDS over the whole network. In other words, the resulting set of links that still flood connects all nodes in the network.

This document does not consider other approaches that guarantee a pruner property on e.g. a clique, i.e. a subgraph where every vertex is neighbored to all other vertices in the clique. It assumes that such "ship in the night components" can be considered non-pruners due to their implicit guarantee of correct flooding to nodes that are part of their component where connected to other components.

## 2.2. Beneficial Properties of the Flooding Pruner Framework

Nodes are free to use any kind of pruner to calculate optimized flooding. Any mode of computation, distributed or centralized, will work fine as long as it adheres to Section 2.1.4. Per Section 2.1.2 a node will become part of one and exactly one component after choosing a pruner.

The framework allows but does not assume any centralized instance or election in a component. Computation and communication within each component is completely independent of other components.

A node is free to choose a different pruner or a _non-pruner_ at any point in time independent of all other nodes. It may end up in another component or become a _non-pruner_ with the maximum impact consisting of re-computation within two components that see such node leave or join. For a distributed algorithm, it is likely that only the adjoining nodes have to adjust their pruning decisions. That is to say, the framework allows for node-by-node deployment or migration of pruners without networkwide recomputation of optimized flooding. This is obviously critical to the stability of large networks that may not converge within reasonable time if the whole network were to revert to non-pruning due to networkwide impact. However, such behavior cannot be excluded, for example, in case of election problems due to misconfiguration or topological separation of nodes if the whole network runs a single pruner relying on centralized election. The network itself cannot ensure correctness of a pruner or prevent a pruner having a blast radius of the whole component depending upon the algorithm and further signaling used.

Although the framework provides extreme operational flexibility when deploying pruners, the most likely scenarios are a node-by-node deployment of a single pruner in addition to a non-pruner or, if the necessity arises, a node-by-node migration to another pruner.

## 2.3. Example

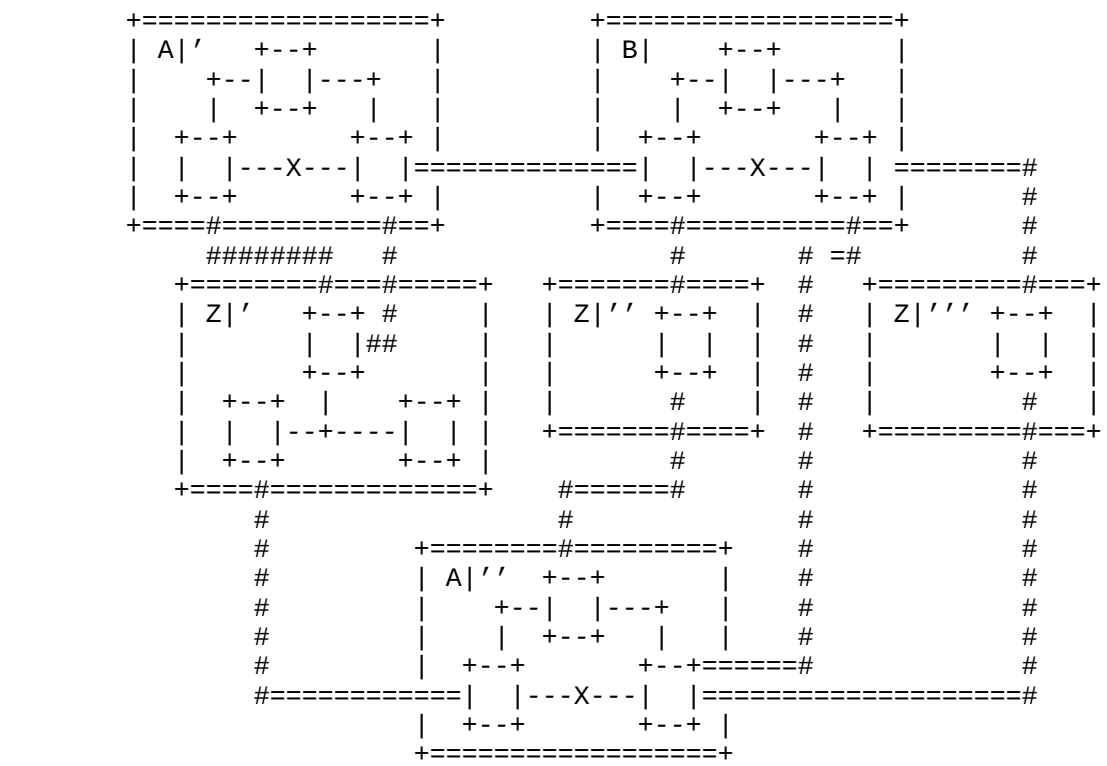


Figure 1: Network of Mixed Pruners

Figure 1 illustrates a network with three pruners running. Two components run pruner A and are denoted as A|' and A|'' and one component runs pruner B. Remaining three components run the _non-pruner_ algorithm (annotated as Z|', Z|'', and Z|'''). CDSes within components are shown by indicating the links that were pruned from flooding as crossed out. Additionally, the links that are included to connect the CDS of the component following the rules listed in Section 2.1.4 have been made thicker. Despite multiple algorithms and components being present in the network, the complete graph is obviously still covered by the involved CDSes.

Figure 1 also illustrates why the overall CDS can easily be more than just a spanning tree of the overall network. A node seeing its neighbor running another algorithm cannot always decide based on local knowledge whether the link should be included in flooding or not. Such a decision could be based on the overall view of the network using some global tie-breaking algorithm. However, due to the potential long flooding paths and one-link minimal cuts, such an algorithm is not considered here but could be proposed in the future.

## 2.4. Signaling

The only signaling required by this framework is a Sub-TLV of the IS-IS Router Capability TLV-242 that is defined in [RFC7981] with the following format. The Sub-TLV MUST be advertised by a node that is actively running any pruner except a non-pruner. The absence of this Sub-TLV signifies within this framework a node being a 'non-pruner' or an algorithm behaving within its component in an equivalent fashion while also guaranteeing flooding on links where it connects to other components.

The Sub-TLV MUST be flooded within a Router Capability TLV that is strictly area scoped and is never leaked between levels.

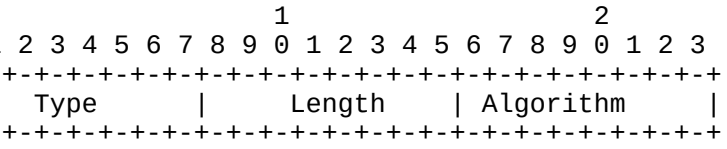


Figure 2

* Type: TBD1

* Length: 1

* Algorithm: 8 bits of a numeric identifier as allocated in the "IGP Algorithm Type For Computing Flooding Topology" registry.

## 3. Security Considerations

This document outlines a framework for extensions to an IGP protocol for operation on high-density network topologies. Implementations SHOULD implement cryptographic authentication compliant to e.g. [RFC5304], and should enable other security measures in accordance with the best common practices for the relevant IGP protocol.

## 4. Contributors

The following people have contributed to this draft and are mentioned without any particular order: Jordan Head, Acee Lindem, Raj Chetan, Les Ginsberg, Peter Psenak and Tony Li.

## 5. Normative References

[RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <https://www.rfc-editor.org/info/rfc7981>.

## 6. Informative References

[RFC5304] Li, T. and R. Atkinson, "IS-IS Cryptographic Authentication", RFC 5304, DOI 10.17487/RFC5304, October 2008, <https://www.rfc-editor.org/info/rfc5304>.

[RFC9667] Li, T., Ed., Psenak, P., Ed., Chen, H., Jalil, L., and S. Dontula, "Dynamic Flooding on Dense Graphs", RFC 9667, DOI 10.17487/RFC9667, October 2024, <https://www.rfc-editor.org/info/rfc9667>.

## Authors' Addresses

Tony Przygienda
Juniper Networks
Email: prz@juniper.net

Shraddha Hegde
Juniper Networks
Email: shraddha@juniper.net