

LSR Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 25, 2019

H. Smit, Ed.  
G. Van de Velde  
Nokia  
October 22, 2018

IS-IS Sparse Link-State Flooding  
draft-hsmit-lsr-isis-dnfm-00

Abstract

This document describes a technology extension to reduce link-state flooding in highly resilient dense networks. It does this by using simple and backwards-compatible extensions to reduce the number of adjacencies over which link-state flooding takes place.

"IS-IS Sparse Link-State Flooding" is an extension to the IS-IS routing protocol.

It is relatively easy to understand and implement. It is backwards compatible. It requires no per-node configuration. It uses a distributed algorithm, therefore no centralized computations are required. No complex computations are required on each node in the network. The algorithm has no requirements for the network topology. It can be deployed in a redundant way to improve robustness and convergence-times.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

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 April 25, 2019.

#### Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
2. High level overview of Sparse Link-State Flooding . . . . .	3
3. The Sparse Link-State Flooding algorithm in detail . . . . .	4
3.1. Role of the Anchor . . . . .	4
3.2. Bootstrapping the flooding . . . . .	4
3.3. Determining which adjacency a router wants to flood over . . . . .	5
3.4. Determining where flooding can be suppressed . . . . .	5
4. Using multiple concurrent flooding topologies . . . . .	7
5. Benefits of the Sparse Link-State Flooding algorithm . . . . .	7
6. Extensions to IS-IS PDUs . . . . .	8
6.1. Anchor TLV in LSPs . . . . .	8
6.2. Flooding-Suppression TLV in IIHs . . . . .	8
7. Operations of the new Sparse Link-State Flooding algorithm . . . . .	9
7.1. Flooding at the anchor itself . . . . .	9
7.2. New action after each SPF . . . . .	9
7.3. When sending a IIH . . . . .	10
7.4. When receiving a IIH . . . . .	10
7.5. When installing a new LSP in the LSDB . . . . .	10
7.6. Preventing loops in the flooding topology . . . . .	10
7.7. Fall-back to classic full flooding . . . . .	11
8. Security Considerations . . . . .	11
9. IANA Considerations . . . . .	11
10. References . . . . .	11
10.1. Normative References . . . . .	11
10.2. Informative References . . . . .	11
Authors' Addresses . . . . .	12

## 1. Introduction

In dense network topologies, using massive ECMP or massive numbers of resilient links, the flooding algorithm of link-state protocols is highly redundant. This results in unnecessary overhead, potentially overloading control planes, decreasing robustness and slowing down convergence. Because of this perceived inefficiency, some operators have resorted to using BGP as the IGP in their data center networks. Draft-li-dynamic-flooding [3] describes this in more detail. However it is very clear that using an Exterior Gateway Protocol as an IGP is sub-optimal, if only due to the configuration overhead.

This document proposes a technology extension to reduce the number of interfaces over which a link-state protocol floods its updates in highly resilient networks. The result is a sparse flooding topology over a dense physical network topology. We describe details how to implement this algorithm for the IS-IS protocol [2]. This algorithm can be extended to other link-state routing protocols, like OSPF. However, no details for protocols other IS-IS are included in this document.

This proposal uses simple and backwards-compatible extensions. It is easy to understand and relatively easy to implement for IS-IS coders. These proposed IS-IS extensions do not require additional configuration on every router. However, it might be beneficial for the operation of the algorithm to manually configure one or more routers as "anchors" in the network. The purpose of an "anchor" is explained in the next section of this document. This extension uses a distributed algorithm. No centralized calculations need to be performed. Each pair of routers decide for themselves where flooding can be suppressed. After every regular SPF computation a router can adjust the interfaces over which it does flooding. This decision requires no computational-complex calculations.

## 2. High level overview of Sparse Link-State Flooding

The goal of the new Sparse Link-State Flooding algorithm is to create a tree of nodes and links, over which updates will be flooded. This tree is called "the flooding topology". The flooding topology includes all the nodes in the network. But it includes only a (small) subset of all available links in the physical network.

The idea is that the flooding topology starts at a single router in the network. This single router is called "the anchor". Routers that are adjacent to the anchor will "attach" or "clamp" themselves to the flooding topology. Making the flooding topology bigger. Their neighbors will "attach" themselves as well, making the flooding topology spread out. In the end all routers will be part of the

flooding topology. The flooding topology resembles a tree, with the anchor as the root of the tree.

The decision to flood or not flood over an adjacency is a local matter. This makes the algorithm a distributed algorithm. The flooding topology itself is not flooded through the network. Only the location of the anchor(s) is announced in LSPs. An anchor announces itself by including this information in its LSP. Two adjacent routers determine whether they need to exchange LSPs or not via a mechanism using a new TLV in hello PDUs (IIHs).

This algorithm can be run once, or multiple times in parallel. This creates one or more concurrent flooding topologies. This provides robustness and faster convergence to the flooding process. We envision that anchors are configured manually, like BGP's Route Reflectors. Or they can be elected automatically. For this the anchor-TLV contains a priority field, to allow operators to have influence on the location of the anchor(s).

### 3. The Sparse Link-State Flooding algorithm in detail

#### 3.1. Role of the Anchor

Each flooding topology needs a root of its tree. The router acting as root is called "the anchor" of a flooding topology. An anchor router includes information in its LSP to announce that it wants to function as an anchor. This information can be encoded as a new TLV, or as a new capability in the existing IS-IS capability TLV. This choice is open for discussion.

The content of this new TLV includes a priority. If multiple routers advertise their willingness to act as an anchor, the anchor with the highest priority is chosen as the anchor. If multiple potential anchors have the same priority, then the router with the highest system-id is chosen as the anchor.

Besides announcing itself as an anchor in its LSP, the role of the anchor-route is purely passive. No extra actions are required of the anchor.

#### 3.2. Bootstrapping the flooding

When a router boots, or when a new adjacency comes up, routers need to synchronize their LSDBs. The reason is that a network could have been partitioned in two separate parts. And flooding over the new adjacency might be the only way to make the two parts of the network aware of each other.

After the LSDBs are synchronized, and at least one SPF computation has been executed, the new algorithm can be used. An implementation could use a longer grace period to wait before using the new algorithm, to ensure all or most of the LSPs in a network have been received.

### 3.3. Determining which adjacency a router wants to flood over

The decision to do regular flooding, or suppress flooding, is done as follows. After each SPF computation, a router looks at the newly computed route towards the anchor. Each router wants to do flooding over the adjacency to a router that is closer to the anchor than it is itself. This guarantees that each router will do flooding with a router that is already part of the flooding topology.

If there are multiple (equal-cost) paths towards the anchor, one of the next-hop adjacencies of the route towards the anchor is chosen to flood over. It doesn't matter which adjacency that is, as long as the adjacent router is closer to the anchor.

When the flooding topology breaks, the two routers next to the point of breakage will notice. They will each generate a new LSP. And they will send out that new LSP over the old flooding topology. The LSP generated by the router that is still reachable through the old flooding topology will be received by all routers on their side of the breakage. This will trigger new SPF computations on all those routers. This SPF computation will compute a new path towards the anchor. The routers will now adjust their flooding topology according to the new path they have just computed. All routers in the network do this. New LSPs will be flooded over the new flooding topology. Which might trigger a follow-up SPF computation. Which might cause routers to adjust their flooding topology again. After a while all routers will have received all new LSPs. Which will guarantee that they will all compute a new correct flooding topology.

A requirement is that when routers start using an adjacency for their flooding topology, they need to synchronize LSDBs first. This is done by exchanging CSNPs. This can potentially be done more reliable and faster when doing IS-IS Flooding over TCP [4].

### 3.4. Determining where flooding can be suppressed

The decision whether to flood over an adjacency or not is a local matter. Only the two routers of the adjacency are involved in this decision. Both routers have a say in whether flooding will be suppressed or not.

This document defines a new TLV, called the Flooding-Suppression TLV, to be included in Hello PDUs (IIHs). This new TLV includes a field that indicates whether a router wants to do flooding over this interface, or wants to suppress flooding. The content of this TLV is set according to the decision made after each SPF, as explained in the previous section of this document.

As a result, a router keeps two new pieces of state for each adjacency.

- o Does the router itself want to flood over this adjacency ? We'll call this the adjacency's "suppression-local-request-state".
- o Does the neighbor want to flood over this adjacency ? We'll call this the adjacency's "suppression-neighbor-request-state".

The suppression-local-request-state is determined after each SPF computation.

The suppression-neighbor-request-state is learned from examining the Flooding-Suppression TLV in each received IIH. If a router did not include the new Flooding-Suppression TLV in its IIH, it is assumed that the neighbor does want to flood over this adjacency.

When both "suppression-local-request-state" and "suppression-neighbor-request-state" are true, then the overall "suppression-state" of the interface is set to true. In that case flooding over the interface is to be suppressed. In all 3 other cases, where at least one of the two routers does not want to suppress flooding, flooding is done in the normal way.

So flooding over an adjacency is only suppressed when both neighbors have indicated that they want to suppress flooding over the adjacency. This means that when one of the two routers does not support this new algorithm, and thus does not include the new TLV in its IIH, flooding is always done. This makes the algorithm backwards compatible with routers that do not support this new extension of the protocol.

A router will always have one or more flooding adjacencies. One adjacency that the router itself needs, to "clamp" on to the part of the flooding topology that is closer to the anchor than it is itself. This adjacency points towards the anchor. And zero or more adjacencies that its neighbors, downstream of the anchor, use to clamp themselves onto the flooding topology. These adjacencies point away from the anchor.

#### 4. Using multiple concurrent flooding topologies

It is possible to use more than one flooding topology in parallel. This requires more than one anchor. For each anchor a new flooding topology is built. These flooding topologies can co-exist without problems.

All that is required is that after each SPF computation, the router examines the shortest path to each anchor. And sets the local state of each adjacency according to this. This guarantees that the router will "clamp onto" each flooding topology.

To ensure an optimal use of parallel flooding topologies, all routers in an IS-IS flooding domain (area or level-2 backbone) should use the same number of parallel flooding topologies. This can be done through configuration. Or an easier way would be to include the number of parallel flooding topologies to use, inside the new Anchor TLV. When looking for Anchors, a router must first find all LSPs with the new Anchor TLV. It then selects the router with the highest Anchor-priority as the main anchor. If multiple router use the same priority, the router with the highest system-id is selected as the anchor. Once the main anchor has been determined, a router looks inside the new anchor-TLV to determine how many parallel flooding topologies it should use. It then selects that amount of anchors with the highest priorities, to set the flooding-state of adjacencies pointing towards those anchors.

Flooding suppression is a local matter. Therefore an implementation can decide to flood over more adjacencies than the minimum to build the minimal flooding topology. It can signal this through the Flooding-Suppression TLV in its IIHs. This can improve robustness and convergence times, at the cost of some extra flooding overhead.

#### 5. Benefits of the Sparse Link-State Flooding algorithm

The algorithm described in this document has a number of advantages.

- o The algorithm is a distributed algorithm. Distributed algorithms are usually more robust than centralized algorithms. The flooding topology itself does not need to be flooded, which makes the algorithm easier when the flooding topology breaks.
- o The algorithm is backwards compatible. No flag-day is required to introduce this new sparse-flooding extension. Older routers that do not support the new extension will obviously not include the flooding-state TLV in their IIHs. The result of this is that regular flooding is done over all adjacencies of those older

routers. This guarantees that older routers will never break the flooding topology.

- o No extra computations have to be done to compute the flooding topology. Using the result of the regular SPF computation suffices to determine over which adjacencies a router wants to flood.
- o The proposed algorithm is robust and guarantees that a flooding topology eventually heals so that all routers are included in the flooding again.
- o Several instances of the algorithm can be run in parallel. This results in multiple parallel flooding topologies. Although parallel flooding topologies are not required for correct operation of the algorithm, it will help in speeding up the healing of the flooding topology. And thus convergence times in general.

## 6. Extensions to IS-IS PDUs

To implement this algorithm, we need two extensions of IS-IS PDUs.

### 6.1. Anchor TLV in LSPs

A new Anchor TLV in the LinkState PDUs. This TLV indicates that a router can be used as an anchor. This new TLV must include a priority field. And it should include a field that suggests how many parallel flooding topologies all routers should use.

### 6.2. Flooding-Suppression TLV in IIHs

A new Flooding-Suppression TLV in the IIH PDUs. This TLV is used to indicate to the neighbor if a router wants to suppress flooding over the adjacency. This new TLV holds three fields:

- o Flooding suppression suggestion field: this field indicates whether the sending router would like to suppress flooding over this interface or not. The value of this field is set to the current "suppression-local-request-state". Note, only when two routers both indicate they want to suppress flooding, then flooding will indeed be suppressed.
- o Resulting actual suppression field: this field indicates whether the sending router will or will not do flooding. The value of this field is set to the current "suppression-state" of the interface. This field is included only for debugging purposes. The first field (the received suppression-local-request-state



field) is used to make the flooding decision. The result of that decision is announced in the second field.

- o The number of currently active flooding adjacencies. This field can be used by the receiving router to pick a flooding adjacency when there are multiple ECMP paths towards the anchor. A router can pick the upstream router with the least amount of flooding adjacencies. In dense networks with many parallel paths, this can help spreading out the load of flooding equally over multiple routers.

Backward compatibility: when a router does not include the Flooding-State TLV in the IIHs it sends out, it can be treated as if that router included the Flooding-State TLV while setting the first field to: "I do not want to suppress flooding".

## 7. Operations of the new Sparse Link-State Flooding algorithm

### 7.1. Flooding at the anchor itself

When a router is acting as the anchor, it floods over all its interfaces. It does include the Flooding-Suppression TLV in its IIHs, but it always sets the value inside the new TLV to "I do not want to suppress flooding".

### 7.2. New action after each SPF

At the end of each SPF computation, a router looks at the best-path to reach the anchor-router. The router sets the "suppression-local-request-state" for that adjacency to false. The router sets the "suppression-local-request-state" for all other adjacencies to true.

If the best-path to the anchor-router's is load-balanced over multiple adjacencies, the router picks one of those adjacencies as its own "upstream flooding adjacencies".

A router must take effort to ensure it changes its "upstream flooding adjacency" as little as possible. Switching its upstream flooding adjacency is not without cost. Every time an adjacency changes from suppressed flooding to normal flooding, the LSDBs of the two routers must be synchronized.

If the "suppression-local-request-state" changed for one or more adjacencies, compared to the state after the previous SPF computation, the router will re-compute the "suppression-state". If the "suppression-state" of an adjacency changes, the router will start or stop flooding over that adjacency.

### 7.3. When sending a IIH

When a router sends an IIH, it includes the new Flooding-Suppression TLV.

For adjacencies that were selected as "upstream flooding adjacency", the value of the Flooding-Suppression TLV must be set to: "I do not want to suppress flooding". For all other adjacencies the value must be set to: "I do want to suppress flooding".

### 7.4. When receiving a IIH

When a router receives an IIH, it checks for the existence of the new Flooding-Suppression TLV.

If it there is none, the state of the neighbor is assumed to be: "I do not want to suppress flooding".

If the "suppression-remote-request-state" changed for this adjacency, compared to the state after receiving the previous IIH, the router will re-compute the "suppression-state". If the "suppression-state" of an adjacency changes, the router will start or stop flooding over that adjacency.

### 7.5. When installing a new LSP in the LSDB

When a router receives a new LSP, it installs it in the LSDB. It will normally then set the IS-IS SRM (Send Routing Message) bits for all adjacencies (in UP state). Now, with the new algorithm, it will set SRM-bits for only the adjacencies that are part of the reduced flooding topology.

### 7.6. Preventing loops in the flooding topology

When the flooding topology changes, during a short period of time different routers can have a different view of the flooding topology. This can make the actual flooding topology in use be a random cyclic graph, instead of a non-cyclic tree. This is not a problem. The flooding algorithm in link-state protocols deals with this by default. An LSP is only (scheduled to be) flooded the first time it is received and installed in the LSDB.

The Sparse Link-State Flooding algorithm has some resemblance to the Spanning Tree Protocol used by transparent bridges. Transient forwarding loops can be a huge problem in the operation of a SPT network. However, while the flooding topology can be looping for short periods of time, this is not a problem at all. Because as described in the previous paragraph, link-state flooding will take

care of this by default. This works because routers keep copies of the LSPs they forward in their LSDB. This allows them to determine if they have received an LSP before or not. In STP routers have no recollection of data-frames that they have forwarded in the past. So in STP looping frames can not be recognized as looping.

To improve convergence times during changes of the flooding topology it is recommended that when a router changes the state of an adjacency from flooding to non-flooding, both routers keep flooding over this adjacency for a short period of time. A suggested value for this is 30 or 60 seconds. By doing this, during changes of the flooding topology, both the old and the new topology will be in use. This guarantees that LSPs are flooded as quickly as possible. This will also help in repairing the flooding topology itself.

#### 7.7. Fall-back to classic full flooding

When a router thinks it might have got behind on flooding, it can always fall back to normal flooding behaviour. It omits including the Flooding-Suppression TLV from its IIHs. Consequently, classic flooding will allow guaranteed synchronization of its IS-IS LSDB with all neighbors. This can be done on all adjacencies at once, or on subset.

#### 8. Security Considerations

This draft introduces no new security considerations.

#### 9. IANA Considerations

This document requests a new TLV and sub-TLV for IS-IS.

#### 10. References

##### 10.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <<http://xml.resource.org/public/rfc/html/rfc2119.html>>.

##### 10.2. Informative References

- [2] International Standard 10589, "Intermediate System to Intermediate System intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473), Second Edition.", 2002.

- [3] Li, T. and P. Psenak, "Dynamic Flooding on Dense Graphs", June 2018.
- [4] Smit, H. and G. Van De Velde, "IS-IS Flooding over TCP", October 2018.

Authors' Addresses

Henk Smit (editor)  
NL

Email: [hhw.smit@xs4all.nl](mailto:hhw.smit@xs4all.nl)

Gunter Van de Velde  
Nokia  
Copernicuslaan 50  
Antwerp  
BE

Email: [gunter.van\\_de\\_velde@nokia.com](mailto:gunter.van_de_velde@nokia.com)