Workgroup: Network Working Group
Internet-Draft:
draft-ietf-lsr-flooding-topo-min-degree-04
Published: 31 December 2021
Intended Status: Standards Track
Expires: 4 July 2022
Authors: H. Chen M. Toy Y. Yang A. Wang
Futurewei Verizon IBM China Telecom
X. Liu Y. Fan L. Liu
Volta Networks Casa Systems Fujitsu

Flooding Topology Minimum Degree Algorithm

Abstract

This document proposes an algorithm for a node to compute a flooding topology, which is a subgraph of the complete topology per underline physical network. When every node in an area automatically calculates a flooding topology by using a same algorithm and floods the link states using the flooding topology, the amount of flooding traffic in the network is greatly reduced. This would reduce convergence time with a more stable and optimized routing environment.

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 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

For some networks such as dense Data Center (DC) networks, the existing Link State (LS) flooding mechanism is not efficient and may have some issues. The extra LS flooding consumes network bandwidth. Processing the extra LS flooding, including receiving, buffering and decoding the extra LSs, wastes memory space and processor time. This may cause scalability issues and affect the network convergence negatively.

This document proposes an algorithm for a node to compute a flooding topology, which is a subgraph of the complete topology per underline physical network. The physical network can be any network, including clos leaf spine network. It can be used in the distributed mode of flooding topology computation for flooding reduction and the centralized mode, which are described in [I-D.ietf-lsr-dynamic-flooding]. When the distributed mode is selected, every node in an area automatically calculates a flooding topology by using a same algorithm and floods the link states using the flooding topology, the amount of flooding traffic in the network is greatly reduced. This would reduce convergence time with a more stable and optimized routing environment.

There may be multiple algorithms for computing a flooding topology. Users can select one they prefer, and smoothly switch from one to another.

2. Terminology

LSA: A Link State Advertisement in OSPF.

LSP: A Link State Protocol Data Unit (PDU) in IS-IS.

LS: A Link Sate, which is an LSA or LSP.

FT: Flooding Topology.

FTC: Flooding Topology Computation.

3. Flooding Topology

For a given network topology, a flooding topology is a sub-graph or sub-network of the given network topology that has the same reachability to every node as the given network topology. Thus all the nodes in the given network topology MUST be in the flooding topology. All the nodes MUST be inter-connected directly or indirectly. As a result, LS flooding will in most cases occur only on the flooding topology, that includes all nodes but a subset of links. Note even though the flooding topology is a sub-graph of the original topology, any single LS MUST still be disseminated in the entire network.

3.1. Flooding Topology Construction

Many different flooding topologies can be constructed for a given network topology. For example, a chain connecting all the nodes in the given network topology is a flooding topology. A circle connecting all the nodes is another flooding topology. A tree connecting all the nodes is a flooding topology. In addition, the tree plus the connections between some leaves of the tree and branch nodes of the tree is a flooding topology.

The following parameters need to be considered for constructing a flooding topology:

* Degree: The degree of the flooding topology is the maximum degree among the degrees of the nodes on the flooding topology. The degree of a node on the flooding topology is the number of connections on the flooding topology it has to other nodes.

* Number of links: The number of links on the flooding topology is a key factor for reducing the amount of LS flooding. In general, the smaller the number of links, the less the amount of LS flooding.

* Diameter: The diameter of the flooding topology is the shortest distance between the two most distant nodes on the flooding topology. It is a key factor for reducing the network convergence time. The smaller the diameter, the less the convergence time.

* Redundancy: The redundancy of the flooding topology means a tolerance to the failures of some links and nodes on the flooding topology. If the flooding topology is split by some failures, it is not tolerant to these failures. In general, the larger the number of links on the flooding topology is, the more tolerant the flooding topology to failures.

Note that the flooding topology constructed by a node is dynamic in nature, that means when the base topology (the entire topology graph) changes, the flooding topology (the sub-graph) MUST be re-computed/re-constructed to ensure that any node that is reachable on the base topology MUST also be reachable on the flooding topology.

4. Algorithms to Compute Flooding Topology

There are many algorithms to compute a flooding topology. A simple and efficient one is briefed, which comprises:

* Selecting a node R0 with the smallest node ID;

* Building a tree using R0 as root in breadth first; and then

* Connecting each node whose degree is one to another node to have a flooding topology.

4.1. Algorithm with Considering Degree

The algorithm is described below, where a variable MaxD with an initial value 3, data structures candidate queue Cq and flooding topology FT are used. Cq and FT comprise elements of form (N, D, PHs), where N represents a Node, D is the Degree of node N, and PHs contains the Previous Hops of node N. The detailed FT computation by the algorithm is illustrated in Appendix A through an example.

The algorithm starts from node R0 as root with

*a maximum degree MaxD of value 3;

*an initial flooding topology FT = {(R0, D=0, PHs={ })}, where node R0 is the root, D = 0 indicates that the Degree (D for short) of R0 is 0 (i.e., the number of links on the flooding topology connected to R0 is 0), PHs = { } indicates that the Previous Hops (PHs for short) of R0 is empty;

*an initial candidate queue Cq = {(R1,D=0, PHs={R0}), (R2,D=0, PHs={R0}), ..., (Rm,D=0, PHs={R0})}, where each of nodes R1 to Rm is connected to R0, its Degree D = 0 and Previous Hops PHs ={R0}, R1 to Rm are in increasing order by their IDs.

1. Finding and removing the first element with node A from Cq that is not on FT and one PH's D in PHs < MaxD, and add the element with A into FT; Set A's D to one, increase A's PH's D by one. If no element in Cq satisfies the conditions, algorithm is restarted with ++MaxD, the initial FT and Cq.

2. If all the nodes are on the FT, then goto step 4;

3. Suppose that node Xi (i = 1, 2,..., n) is connected to node A and not on FT, and X1, X2,..., Xn are in an increasing order by their IDs (i.e., X1's ID < X2's ID < ... < Xn's ID). If Xi is not in Cq, then add it into the end of Cq with D = 0 and PHs = {A}; otherwise (i.e., Xi is in Cq), add A into the end of Xi's PHs; Goto step 1.

4. For each node B on FT whose D is one (from minimum to maximum node ID), find a link L attached to B such that L's remote node R has minimum D and ID and R'D < MaxD, add link L between B and R into FT and increase B's D and R's D by one. Return FT if every node on FT has its D > 1; otherwise, algorithm is restarted with ++MaxD, the initial FT and Cq.

4.2. Algorithm with Considering Others

There may be some constraints on some nodes in a network. For example, in a spine-and-leaf network, there may be a constraint on the degree of every leaf node on the flooding topology, which is that the degree of every leaf node is not greater than a given number ConMaxD of value 2. For each of the other nodes such as the spine nodes, there is no such constraint, that is that ConMaxD is a huge number for each of these nodes.

Step 1 of the algorithm described above is updated below to consider this constraint. In addition to checking constraint PH's D < MaxD, step 1 checks another constraint PH's D < PH's ConMaxD.

1. Finding and removing the first element with node A from Cq that is not on FT and one PH's D in PHs < MaxD and PH's D < PH's ConMaxD, and add the element with A into FT; Set A's D to one, increase A's PH's D by one. If no element in Cq satisfies the conditions, algorithm is restarted with ++MaxD, the initial FT and Cq.

Similarly, step 4 of the algorithm described above is updated to consider this constraint. In addition to checking constraint R's D < MaxD, step 4 checks another constraint R's D < R's ConMaxD.

5. Security Considerations

This document does not introduce any new security issue.

6. IANA Considerations

Under Registry Name: "IGP Algorithm Type For Computing Flooding Topology" under an existing "Interior Gateway Protocol (IGP) Parameters" IANA registries (refer to Section 7.3. IGP [I-D.ietf-lsr-dynamic-flooding]), IANA is requested to assign one value of IGP Algorithm Type For Computing Flooding Topology as follows:

Type Value	Type Name	reference
1	Breadth First Minimum Degree Algorithm	This document
2	Breadth First Leaf Constraint Algorithm	This document

7. Acknowledgements

The authors would like to thank Dean Cheng, Acee Lindem, Zhibo Hu, Robin Li, Stephane Litkowski and Alvaro Retana for their valuable suggestions and comments on this draft.

8. References

8.1. Normative References

[I-D.ietf-lsr-dynamic-flooding]
Li, T., Przygienda, T., Psenak, P., Ginsberg, L., Chen, H., Cooper, D., Jalil, L., Dontula, S., and G. S. Mishra, "Dynamic Flooding on Dense Graphs", Work in Progress, Internet-Draft, draft-ietf-lsr-dynamic-flooding-10, 7

December 2021, <https://www.ietf.org/archive/id/draft-ietf-lsr-dynamic-flooding-10.txt>.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

8.2. Informative References

[I-D.ietf-rtgwg-spf-uloop-pb-statement] Litkowski, S., Decraene, B., and M. Horneffer, "Impact of Shortest Path First (SPF) Trigger and Delay Strategies on IGP Micro-loops", Work in Progress, Internet-Draft, draft-ietf-rtgwg-spf-uloop-pb-statement-10, 16 January 2019, <https://www.ietf.org/archive/id/draft-ietf-rtgwg-spf-uloop-pb-statement-10.txt>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Appendix A. FT Computation Details through Example

This section presents the details on FT computation by the algorithm through an example. The detailed procedure of computing a FT for a network of five nodes with full mess connections is illustrated. Suppose that the network has five nodes R0, R1, R2, R3 and R4; R0's ID < R1's ID < R2's ID < R3's ID < R4's ID. The algorithm starts with MaxD = 3, FT = {(R0, D=0, PH={})}, Cq = { (R1,0,{R0}), (R2,0,{R0}), (R3,0,{R0}), (R4,0,{R0}) }.

```
1. //remove first element (R1,D=0,PHs={R0}) from Cq, R0's D=0 < MaxD
   Cq = { (R2,0,{R0}), (R3,0,{R0}), (R4,0,{R0}) };
   // add (R1,1,{R0}) into FT, increase PH R0's D by one
   FT = { (R0,1, { }), (R1,1, {R0}) }; // Link R1--R0 on FT
             ^^^       ^^^^^^^^^^^^
   // for Ri connected to R1 (in Cq) not on FT, append R1 to Ri's PHs
   Cq = { (R2,0, {R0,R1}), (R3,0, {R0,R1}), (R4,0,{R0,R1}) }.
                     ^^               ^^               ^^
                                R0            ==== Link on FT
                         __//== O ---\__
                      __//     / \      \__   link R1--R0 added to FT
                   __//       /   \        \__
                __//         /     \          \__
               //           /       \            \
           R1 O--\_--------/---------\--------_/--O R4
               \   \____  /           \  ____/   /
                \       \/             \/       /
                 \      /\_____    ____/\      /
                  \    /    ___\__/___   \    /
                   \  /    /          \   \  /
                    \/____/            \___\/
                 R2 O --------------------- O R3

2. // remove the first element (R2,0, {R0,R1}) from Cq, R0's D=1 < MaxD
   Cq = { (R3,0, {R0,R1}), (R4,0,{R0,R1}) }
   // add (R2,1,{R0}) into FT, increase R0's D by one
   FT = { (R0,2,{ }), (R1,1,{R0}), (R2,1,{R0}) } //Link R2--R0 on FT
             ^^^                     ^^^^^^^^^^^
   // for Ri connected to R2 (in Cq) not on FT, append R2 to Ri's PHs
   Cq = { (R3,0, {R0,R1,R2}), (R4,0,{R0,R1,R2}) }
                        ^^                  ^^
                                R0           ==== Link on FT
                         __//== O ---\__
                      __//    // \      \__  link R2--R0 added to FT
                   __//      //   \        \__
                __//        //     \          \__
               //          //       \            \
           R1 O--\_-------//---------\--------_/--O R4
               \   \____ //           \  ____/   /
                \       \/             \/       /
                 \     //\_____    ____/\      /
                  \   //    ___\__/___   \    /
                   \ //    /          \   \  /
                    \/____/            \___\/
                 R2 O --------------------- O R3
```

```
3. //remove the 1st element (R3,0,{R0,R1,R2}) from Cq, R0's D=2 < MaxD
   Cq = { (R4,0,{R0,R1,R2}) }
   // add (R3,1,{R0}) into FT, increase R0's D by one
   FT = { (R0,3,{}), (R1,1,{R0}), (R2,1,{R0}), (R3,1,{R0}) }
             ^^^                               ^^^^^^^^^^^
   // for Ri connected to R3 (in Cq) not on FT, append R3 to Ri's PHs
   Cq = { (R4,0,{R0,R1,R2,R3}) }.
                          ^^
                                R0           ==== Link on FT
                         __//== O ---\__
                      __//    // \\     \__  link R3--R0 added to FT
                   __//      //   \\       \__
                __//        //     \\         \__
               //          //       \\           \
           R1 O--\_-------//---------\\-------_/--O R4
               \   \____ //           \\ ____/   /
                \       \/             \/       /
                 \     //\______   _____/\\    /
                  \   //    ___\__/___   \\   /
                   \ //    /          \   \\ /
                    \/____/            \___\/
                 R2 O --------------------- O R3

4. //remove the 1st element (R4,0,{R0,R1,R2,R3}) from Cq,R1's D=1 < MaxD
   Cq = { }
   // add (R4,1,{R1}) into FT, increase R1's D by one
   FT = {(R0,3,{}), (R1,2,{R0}), (R2,1,{R0}), (R3,1,{R0}), (R4,1,{R1})}
                       ^^^                                 ^^^^^^^^^^^
                                R0           ==== Link on FT
                         __//== O ---\__
                      __//    // \\     \__  link R4--R1 added to FT
                   __//      //   \\       \__
                __//        //     \\         \__
               //          //       \\           \
           R1 O==\_=======//=========\\=======_/==O R4
               \   \____ //           \\ ____/   /
                \       \/             \/       /
                 \     //\______   _____/\\    /
                  \   //    ___\__/___   \\   /
                   \ //    /          \   \\ /
                    \/____/            \___\/
                 R2 O --------------------- O R3
```

All nodes are on FT now. In the following, for each node on FT whose D = 1 (from minimum to maximum ID), link L attached to it and not on FT is found such that L's remote node has minimum D and ID. L is added into FT.

```
5. // On FT, get node R2 with smallest ID whose D=1
   FT = {(R0,3,{}),(R1,2,{R0}),(R2,1,{R0}),(R3,1,{R0}), (R4,1,{R1})}
   // Add link R2--R3 to FT,   ^^^^^^^^^^^
   // where R2--R3 is not on FT, R3's D=1 is minimum first and then
   // R3's ID is minimum (R3 and R4 tie for D), R2's D++ and R3's D++
   FT = {(R0,3,{}),(R1,2,{R0}),(R2,2,{R0,R3}),(R3,2,{R0}),(R4,1,{R1})}
                                  ^^^    ^^      ^^^
                                R0           ==== Link on FT
                         __//== O ---\__
                      __//    // \\     \__  link R2--R3 added to FT
                   __//      //   \\       \__
                __//        //     \\         \__
               //           //       \\           \
           R1 O==\_=======//=========\\=======_/==O R4
               \   \____ //           \\ ____/   /
                \       \/             \/       /
                 \     //\_____    _____/\\    /
                  \   //    ___\__/___   \\   /
                   \ //    /          \   \\ /
                    \/____/            \___\/
                 R2 O ===================== O R3

6. // On FT, get node R4 with smallest ID whose D=1
   FT = {(R0,3,{}),(R1,2,{R0}),(R2,2,{R0,R3}),(R3,2,{R0}),(R4,1,{R1})}
   // Add link R4--R2 to FT, where                        ^^^^^^^^^^^
   // R4--R2 is not on FT, R2's D=2 is minimum first and then R2's ID is
   // minimum (R2 and R3 tie for D), increase R2's D and R4's D by one
   FT = {(R0,3,{}),(R1,2,{R0}),(R2,3,{R0,R3}),(R3,2,{R0}),(R4,2,{R1,R2})
                                  ^^^                        ^^^    ^^
                                R0           ==== Link on FT
                         __//== O ---\__
                      __//    // \\     \__  link R4--R2 added to FT
                   __//      //   \\       \__
                __//        //     \\         \__
               //           //       \\           \
           R1 O==\_=======//=========\\=======//==O R4
               \   \____ //           \\ ____//  /
                \       \/             \//      /
                 \     //\_____    ___//\\     /
                  \   //    ___\__//__   \\   /
                   \ //    //         \   \\ /
                    \/   _//           \___\/
                 R2 O ==//================= O R3
```

FT is computed, which has Degree of 3 and Diameter of 2.

Authors' Addresses

Huaimo Chen

Futurewei
Boston,
United States of America

Email: huaimo.chen@futurewei.com

Mehmet Toy
Verizon
United States of America

Email: mehmet.toy@verizon.com

Yi Yang
IBM
Cary, NC
United States of America

Email: yyietf@gmail.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing
102209
China

Email: wangaj3@chinatelecom.cn

Xufeng Liu
Volta Networks
McLean, VA
United States of America

Email: xufeng.liu.ietf@gmail.com

Yanhe Fan
Casa Systems
United States of America

Email: yfan@casa-systems.com

Lei Liu
Fujitsu
United States of America

Email: liulei.kddi@gmail.com