Flooding Topology Computation Algorithm
draft-cc-lsr-flooding-reduction-04

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

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/.
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. 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.

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 State, 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 below.

- Select a node R0 according to a rule such as the node with the smallest node ID;

- Build a tree using R0 as root of the tree; and then

- Connect each node whose degree is one to the tree to have a flooding topology.
4.1. Algorithm with Considering Degree

The algorithm starts from node R0 as root with a given maximum degree MaxD, a candidate queue Cq = {(R0, D = 0, PHs = { })}, and an empty flooding topology FT = { }. Cq contains one element (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.

1. Finding and removing the first element with node A in Cq that is not on FT and one PH’s D in PHs < MaxD.

   If there is no element with a node in Cq whose PHs != { } and one PH in PHs whose D < MaxD,
   then MaxD++ and restarts algorithm from R0, MaxD, Cq = {(R0,D=0,PHs = { })}, FT = { };
   otherwise (i.e., A with one PH’s D in PHs < MaxD or PHs = { }),
   if PHs = { } (i.e., A is the root),
   then mark A on FT and add A with D=0 and PHs={ } into FT;
   otherwise (i.e., A is not the root. Assume that PH is the first one in PHs whose D < MaxD), PH’s D++, mark A on FT and add A with D=1 and PHs={PH} to FT.

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 Xi’s PHs and elements in PHs are in an increasing order by their degrees first and then IDs; Goto step 1.

4. For each node B in FT whose D is one, find a link L attached to B such that L’s remote node R whose D and ID are minimum; increase B’s D and R’s D by one. Return FT.

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 such as ConMaxD = 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 in Cq that is not on FT and one PH’s D in PHs < MaxD and PH’s D < PH’s ConMaxD.

   If there is no element with a node in Cq whose PHs != { } and one PH in PHs whose D < MaxD and PH’s D < PH’s ConMaxD,

   then MaxD++ and restarts algorithm from R0, MaxD, Cq = {(R0,D=0,PHs = { })}, FT = { };

   otherwise (i.e., node A with one PH’s D in PHs < MaxD and PH’s D < PH’s ConMaxD or PHs = { }),

   if PHs = { } (i.e., A is the root),

   then mark A on FT and add A with D=0 and PHs={ } into FT;

   otherwise (i.e., A is not the root. Assume that PH is the first one in PHs whose D < MaxD and PH’s D < PH’s ConMaxD),

   PH’s D++, mark A on FT and add A with D=1 and PHs={PH} to FT.

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:

<table>
<thead>
<tr>
<th>Type Value</th>
<th>Type Name</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Breadth First Minimum Degree Algorithm</td>
<td>This document</td>
</tr>
</tbody>
</table>
7. Acknowledgements

The authors would like to thank 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]


8.2. Informative References

[I-D.ietf-rtgwg-spf-uloop-pb-statement]


Authors’ Addresses
Huaimo Chen  
Futurewei  
Boston  
USA  

Email: huaimo.chen@futurewei.com

Dean Cheng  
Individual  
Santa Clara  
USA  

Email: deanccheng@gmail.com

Mehmet Toy  
Verizon  
USA  

Email: mehmet.toy@verizon.com

Yi Yang  
IBM  
Cary, NC  
United States of America  

Email: yyietf@gmail.com

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

Email: wangaj.bri@chinatelecom.cn

Xufeng Liu  
Volta Networks  
McLean, VA  
USA  

Email: xufeng.liu.ietf@gmail.com
Yanhe Fan
Casa Systems
USA

Email: yfan@casa-systems.com

Lei Liu
Fujitsu
USA

Email: liulei.kddi@gmail.com