

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 6, 2018

H. Chen  
D. Cheng  
Huawei Technologies  
M. Toy  
Verizon  
Y. Yang  
IBM  
March 5, 2018

ISIS Flooding Reduction  
draft-cc-isis-flooding-reduction-00

## Abstract

This document proposes an approach to flood ISIS link state protocol data units on a topology that is a subgraph of the complete ISIS topology per underline physical network, so that the amount of flooding traffic in the network is greatly reduced, and it would reduce convergence time with a more stable and optimized routing environment. The approach can be applied to any network topology in a single ISIS area.

## 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 <http://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 September 6, 2018.

## Copyright Notice

Internet-Draft

ISIS Flooding Reduction

March 2018

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 (<http://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

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Problem Statement . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Flooding Topology . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Extensions to ISIS . . . . .	<a href="#">6</a>
<a href="#">5.</a>	Flooding Behavior . . . . .	<a href="#">7</a>
<a href="#">5.1.</a>	Nodes Support Flooding Reduction . . . . .	<a href="#">7</a>
<a href="#">5.1.1.</a>	Receiving an ISIS LSP . . . . .	<a href="#">7</a>
<a href="#">5.1.2.</a>	Originating an ISIS LSP . . . . .	<a href="#">8</a>
<a href="#">5.1.3.</a>	An Exception Case . . . . .	<a href="#">8</a>
<a href="#">5.1.4.</a>	One More Note . . . . .	<a href="#">9</a>
<a href="#">5.2.</a>	Nodes Not Support Flooding Reduction . . . . .	<a href="#">9</a>
<a href="#">6.</a>	Security Considerations . . . . .	<a href="#">9</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">9</a>
<a href="#">8.</a>	Acknowledgements . . . . .	<a href="#">9</a>
<a href="#">9.</a>	References . . . . .	<a href="#">9</a>
<a href="#">9.1.</a>	Normative References . . . . .	<a href="#">9</a>
<a href="#">9.2.</a>	Informative References . . . . .	<a href="#">10</a>
<a href="#">Appendix A.</a>	Algorithms to Build Flooding Topology . . . . .	<a href="#">10</a>
<a href="#">A.1.</a>	Algorithms to Build Tree without Considering Flag F . . . . .	<a href="#">10</a>
<a href="#">A.2.</a>	Algorithms to Build Tree Considering Flag F . . . . .	<a href="#">11</a>
<a href="#">A.3.</a>	Connecting Leaves . . . . .	<a href="#">14</a>
	Authors' Addresses . . . . .	<a href="#">14</a>

## [1.](#) Introduction

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

A flooding reduction method between spines and leaves is proposed in [\[I-D.shen-isis-spine-leaf-ext\]](#). The problem on flooding reduction and an architectural solution are discussed in [\[I-D.li-dynamic-flooding\]](#). This document proposes an approach to flood ISIS LSPs on a topology that is a subgraph of the entire ISIS topology per underline physical network, so that the amount of flooding traffic in the network is greatly reduced. The workload for processing the extra LSP flooding is decreased significantly. This would improve the scalability and speed up the network convergence, stable and optimize the routing environment.

The approach proposed is applicable to any network topology in a single ISIS area. The approach is backward compatible.

## [2.](#) Problem Statement

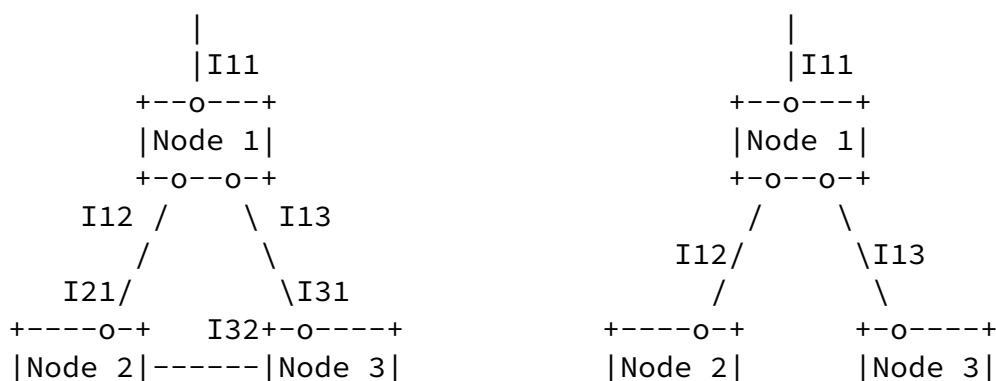
ISIS, like other link-state routing protocols, deploys a so-called reliable flooding mechanism, where a node must transmit a received or self-originated LSP to all its ISIS interfaces (except the interface where a LSP is received) in the defined context. While this mechanism assures each LSP being distributed to every ISIS node in the relevant routing area or domain, the side-effect is that the mechanism often causes redundant LSPs in individual network segments (e.g., on an ISIS point-to-point link or a broadcast subnet), which in turn forces ISIS nodes to process identical LSPs more than once.

This results waste of ISIS link bandwidth and ISIS nodes' computing resources, and the delay of ISIS topology convergence.

The problem explained above becomes more serious in ISIS networks with large number of nodes and links, and in particular, higher degree of interconnection (e.g., meshed topology, spine-leaf topology, etc.). In some environment such as in data centers, the drawback of the existing flooding mechanism has already caused operational problems, including repeated and waves of flooding storms, chock of computing resources, slow convergence, oscillating topology changes, instability of routing environment.

One example is as shown in Figure 1 (a), where Node 1, Node 2 and Node 3 are interconnected in a mesh. When Node 1 receives a new or updated ISIS LSP on its interface I11, it by default would forward to its interface I12 and I13 towards Node 2 and Node 3, respectively, after processing. Node 2 and Node 3 upon reception of the LSP and after processing, would potentially flood the same LSP over their respective interface I23 and I32 toward each other, which is obviously not necessary and at the cost of link bandwidth as well as both nodes' computing resource.

In example Figure 1 (b), Node 2 and Node 3 both connect to a LAN where Node 4, Node 5 and Node 6 also connect to. When Node 1 receives a LSP as in (a) and floods it to Node 2 and Node 3 respectively, the two nodes would in turn both (instead of one) flood to the LAN, which is unnecessary and at the cost of link bandwidth as well as computing resource of all nodes connected to the LAN.



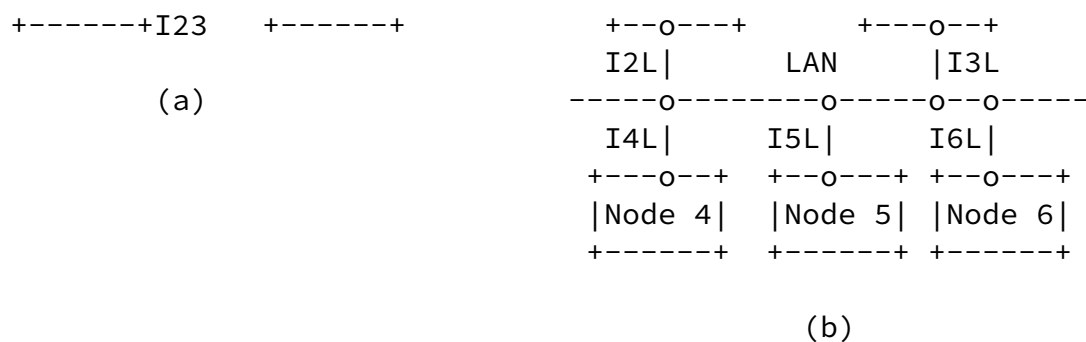


Figure 1

### 3. Flooding Topology

It is a norm that an ISIS node sending a received LSP and self-originated LSP to all its ISIS interfaces (except that where an LSP is received), as the reliable-flooding mechanism requires, i.e., any

ISIS LSP would potentially traverses on each ISIS link in a given ISIS network topology, sometimes both directions. As demonstrated in [Section 2](#), dissemination over the entire ISIS network topology has drawbacks.

To change ISIS's aggressive flooding behavior, a flooding topology is introduced. For a given ISIS 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, ISIS flooding will in most cases occur only on the flooding topology, that includes all ISIS nodes but a subset of ISIS links. Note even the flooding topology is a sub-graph of the original ISIS topology, any single LSP MUST still be disseminated in the entire ISIS network.

There are many different flooding topologies for a given ISIS network topology. 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.

There are many different ways to construct a flooding topology for a given ISIS network topology. A few of them are listed below:

- o One node in the network builds a flooding topology and floods the flooding topology to all the other nodes in the network (This seems not very good. Flooding the flooding topology may increase the flooding.);
- o Each node in the network automatically calculates a flooding topology by using the same algorithm (No flooding for flooding topology);
- o Links on the flooding topology are configured statically.

The minimum requirement for a flooding topology is all ISIS nodes are interconnected (directly or indirectly), but there is only one path from any node to any other node. While this lean-and-mean type of flooding topology degrades ISIS flooding traffic volume to the least, it may introduce some delay of topology convergence in the network with some network topologies. To compensate convergence efficiency, additional ISIS links may be added as part of the flooding topology. There is a trade-off between the density of the flooding topology and the convergence efficiency.

Note that the flooding topology constructed by an ISIS node is dynamic in nature, that means when the ISIS's 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.

For reference purpose, some algorithms that allow ISIS nodes to automatically compute flooding topology are elaborated in [Appendix A](#). However, this document does not attempt to standardize how a flooding topology is established.

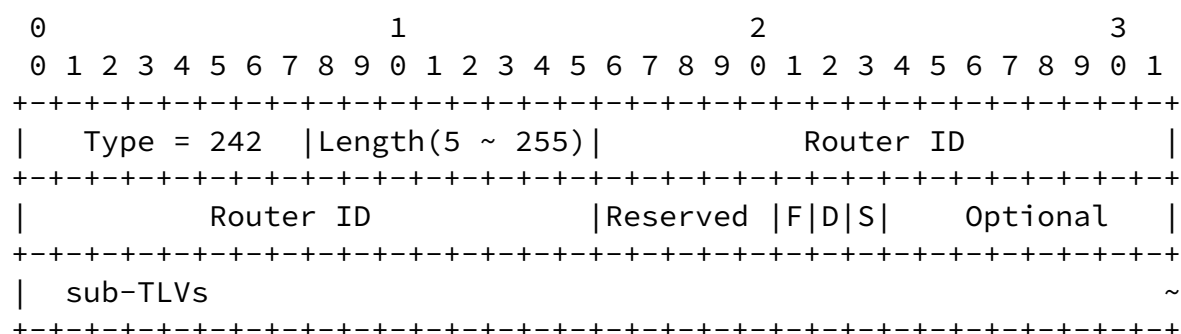
#### [4.](#) Extensions to ISIS

A 1-bit flag F is defined in an ISIS router capability TLV. Flag F set to 1 indicates that the router supports ISIS LSP flood reduction described in this document; and Flag F set to 0 indicates that the router does not do so.

This flag is used for an ISIS node during the process of computing a flooding topology. An ISIS node that advertises its LSP containing a capability TLV with "F" bit set to 1 MUST always be included in the flooding topology computed by other ISIS nodes; but in contrast, the node with "F" bit set to zero may or may not be included in the flooding topology by other nodes, depending on how other nodes construct their flooding topology.

This flag can also be used for an ISIS node to trigger a decision whether it wants to perform LSP flooding to its neighbor.

The format of an ISIS router capability TLV with flag F is shown below.



## 5. Flooding Behavior

### 5.1. Nodes Support Flooding Reduction

This section describes ISIS flooding behavior for ISIS nodes that support flooding reduction described in this document. For these nodes, they MUST set "F" bit to 1 in their LSPs (see [Section 4](#)). The flooding behavior for these nodes differs from that as specified in

ISIS ([RFC1195]). [Section 5.1.1](#) describes the flooding behavior when an ISIS node receives an ISIS LSP from one of its interface, and [Section 5.1.2](#) describes the flooding behavior for LSP originated by itself.

The revised flooding procedure MUST flood LSPs to every node in the network in any case, as the standard ISIS flooding procedure does.

It assumes that the ISIS node of which the flooding behavior is described below is on the flooding topology, i.e., the node and at least one of its ISIS interface are on the flooding topology, where:

1. When the node has only one interface on the flooding topology, the node is a leaf on the topology.
2. When the node has two interfaces on the flooding topology, the node is a transit node on the topology.
3. A flooding topology with nodes having one or two interfaces on the topology is a lean graph, i.e., there is only one path from any node to any other node on the graph. For flooding efficiency, there could be extra ISIS interfaces that are on the flooding topology, i.e., a node may have more than two interfaces that belong to the flooding topology.

#### [5.1.1](#). Receiving an ISIS LSP

The flooding behavior when an ISIS node receives a newer ISIS LSP that is not originated by itself from one of its ISIS interface is as follows:

1. The LSP is received on a link that is on the flooding topology. The LSP is flooded only to all the other interfaces that are on the flooding topology.
2. The LSP is received on a link that is not on the flooding topology. This situation can happen when a neighboring node on a point-to-point link newly forms adjacency with the receiving node, or is not currently on the flooding topology; it can happen when the LSP sending neighbor does not support the ISIS flooding

reduction (i.e., with "F" bit set to zero); it can also happen as



the receiving link is a broadcast-type interface. The LSP is flooded only to all other interfaces that are on the flooding topology.

3. In both cases above, if there is any neighboring node that is advertising its Router LSP with "F" bit set to zero (see [Section 4](#)) but it is not on the flooding topology, the received LSP MUST also be sent to this neighboring node.

In any case, the LSP must not be transmitted back to the receiving interface.

Note before forwarding a received LSP, the ISIS node would do the normal processing as usual.

#### [5.1.2](#). Originating an ISIS LSP

The flooding behavior when an ISIS node originates an ISIS LSP is as follows:

1. If it is a refresh LSP, i.e., there is no significant change contained in the LSP comparing to the previous LSP, the LSP is transmitted over links on the flooding topology. In addition, if there is any neighboring node that is advertising its Router LSP with "F" bit set to zero (see [Section 4](#)) but it is not on the flooding topology, the LSP MUST also be sent to this neighboring node.
2. Otherwise, the LSP is transmitted to all ISIS interfaces. Choosing this action instead of limiting to links on flooding topology would speed up the synchronization around the advertising node's neighbors, which could then disseminate the new LSP quickly.

#### [5.1.3](#). An Exception Case

In [Section 5.1.1](#) and [Section 5.1.2](#), there are times when an ISIS node sending out a LSP to an interface on the flooding topology detects an interface or node failure. Note the flooding topology was pre-computed/pre-constructed; but if at the time the interface or the neighboring node goes down before a re-newed flooding topology can be computed/constructed, the node MUST send out the LSP to all interfaces (except where it is received from) as a traditional ISIS node would do. This handling is also taking place if there are more than one egress interfaces on the existing flooding topology, i.e., if at least one egress interface or neighboring node fails, the ISIS node does traditional flooding before the flooding topology is re-

built.

#### [5.1.4.](#) One More Note

The destination address that is used when an ISIS node sends out a LSP on an interface on its flooding topology follows the specification in ISIS ([RFC1195]). This means on a local LAN, all other ISIS nodes will receive the LSP.

#### [5.2.](#) Nodes Not Support Flooding Reduction

For ISIS nodes that do not support flooding reduction as described in this document, they MUST set "F" bit to 0 in their Router LSP (see [Section 4](#)); note this is also a default setting. These nodes may or may not be on the flooding topology constructed by other nodes that support flooding reduction in the same ISIS area, however that is not a business these nodes need to concern.

The LSP flooding behavior of ISIS nodes that do not support reduction as described in this document MUST follow that as specified in ISIS ([RFC1195]).

### [6.](#) Security Considerations

This document does not introduce any security issue.

### [7.](#) IANA Considerations

This document has no request to IANA.

### [8.](#) Acknowledgements

TBD.

### [9.](#) References

#### [9.1.](#) Normative References

- [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](#), March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.

## [9.2.](#) Informative References

[I-D.li-dynamic-flooding]

Li, T., "An Architecture for Dynamic Flooding on Dense Graphs", [draft-li-dynamic-flooding-02](#) (work in progress), March 2018.

[I-D.shen-isis-spine-leaf-ext]

Shen, N., Ginsberg, L., and S. Thyamagundalu, "IS-IS Routing for Spine-Leaf Topology", [draft-shen-isis-spine-leaf-ext-05](#) (work in progress), January 2018.

## [Appendix A.](#) Algorithms to Build Flooding Topology

There are many algorithms to build a flooding topology. A simple and efficient one is briefed below.

- o Select a node R according to a rule such as the node with the biggest/smallest node ID;
- o Build a tree using R as root of the tree (details below); and then
- o Connect  $k$  ( $k \geq 0$ ) leaves to the tree to have a flooding topology (details follow).

### [A.1.](#) Algorithms to Build Tree without Considering Flag F

An algorithm for building a tree from node R as root starts with a candidate queue Cq containing R and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft
2. If Cq is empty, then return with Ft

3. Suppose that node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node A and not in Ft and  $X_1, X_2, \dots, X_n$  are in a special order. For example,  $X_1, X_2, \dots, X_n$  are ordered by the cost of the link between A and  $X_i$ . The cost of the link between A and  $X_i$  is less than the cost of the link between A and  $X_j$  ( $j = i + 1$ ). If two costs are the same,  $X_i$ 's ID is less than  $X_j$ 's ID. In another example,  $X_1, X_2, \dots, X_n$  are ordered by their IDs. If they are not ordered, then make them in the order.

4. Add  $X_i$  ( $i = 1, 2, \dots, n$ ) into the end of Cq, goto step 1.

Another algorithm for building a tree from node R as root starts with a candidate queue Cq containing R and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft
2. If Cq is empty, then return with Ft
3. Suppose that node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node A and not in Ft and  $X_1, X_2, \dots, X_n$  are in a special order. For example,  $X_1, X_2, \dots, X_n$  are ordered by the cost of the link between A and  $X_i$ . The cost of the link between A and  $X_i$  is less than the cost of the link between A and  $X_j$  ( $j = i + 1$ ). If two costs are the same,  $X_i$ 's ID is less than  $X_j$ 's ID. In another example,  $X_1, X_2, \dots, X_n$  are ordered by their IDs. If they are not ordered, then make them in the order.
4. Add  $X_i$  ( $i = 1, 2, \dots, n$ ) into the front of Cq and goto step 1.

A third algorithm for building a tree from node R as root starts with a candidate list Cq containing R associated with cost 0 and an empty flooding topology Ft:

1. Remove the first node A from Cq and add A into Ft
2. If all the nodes are on Ft, then return with Ft
3. Suppose that node A is associated with a cost  $C_a$  which is the cost from root R to node A, node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node A and not in Ft and the cost of the link between A and  $X_i$  is  $LC_i$  ( $i=1, 2, \dots, n$ ). Compute  $C_i = C_a + LC_i$ ,

check if  $X_i$  is in  $C_q$  and if  $C_{xi}$  (cost from  $R$  to  $X_i$ )  $< C_i$ . If  $X_i$  is not in  $C_q$ , then add  $X_i$  with cost  $C_i$  into  $C_q$ ; If  $X_i$  is in  $C_q$ , then If  $C_{xi} > C_i$  then replace  $X_i$  with cost  $C_{xi}$  by  $X_i$  with  $C_i$  in  $C_q$ ; If  $C_{xi} == C_i$  then add  $X_i$  with cost  $C_i$  into  $C_q$ .

4. Make sure  $C_q$  is in a special order. Suppose that  $A_i$  ( $i=1, 2, \dots, m$ ) are the nodes in  $C_q$ ,  $C_{ai}$  is the cost associated with  $A_i$ , and  $ID_i$  is the ID of  $A_i$ . One order is that for any  $k = 1, 2, \dots, m-1$ ,  $C_{ak} < C_{aj}$  ( $j = k+1$ ) or  $C_{ak} = C_{aj}$  and  $ID_k < ID_j$ . Goto step 1.

## [A.2.](#) Algorithms to Build Tree Considering Flag $F$

An algorithm for building a tree from node  $R$  as root with consideration of flag  $F$  starts with a candidate queue  $C_q$  containing  $R$  associated with previous hop  $PH=0$  and an empty flooding topology  $F_t$ :

1. Remove the first node  $A$  with its flag  $F$  set to one from the candidate queue  $C_q$  if there is such a node  $A$ ; otherwise (i.e., if there is not such node  $A$  in  $C_q$ ), then remove the first node  $A$  from  $C_q$ . Add  $A$  into the flooding topology  $F_t$ .
2. If  $C_q$  is empty or all nodes are on  $F_t$ , then return with  $F_t$
3. Suppose that node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node  $A$  and not in the flooding topology  $F_t$  and  $X_1, X_2, \dots, X_n$  are in a special order considering whether some of them with flag  $F = 1$ . For example,  $X_1, X_2, \dots, X_n$  are ordered by the cost of the link between  $A$  and  $X_i$ . The cost of the link between  $A$  and  $X_i$  is less than that of the link between  $A$  and  $X_j$  ( $j = i + 1$ ). If two costs are the same,  $X_i$ 's ID is less than  $X_j$ 's ID. The cost of a link is redefined such that 1) the cost of a link between  $A$  and  $X_i$  both with  $F = 1$  is much less than the cost of any link between  $A$  and  $X_k$  where  $X_k$  with  $F=0$ ; 2) the real metric of a link between  $A$  and  $X_i$  and the real metric of a link between  $A$  and  $X_k$  are used as their costs for determining the order of  $X_i$  and  $X_k$  if they all (i.e.,  $A, X_i$  and  $X_k$ ) with  $F = 1$  or none of  $X_i$  and  $X_k$  with  $F = 1$ .
4. Add  $X_i$  ( $i = 1, 2, \dots, n$ ) associated with previous hop  $PH=A$  into the end of the candidate queue  $C_q$ , and goto step 1.

Another algorithm for building a tree from node  $R$  as root with

consideration of flag  $F$  starts with a candidate queue  $C_q$  containing  $R$  associated with previous hop  $PH=0$  and an empty flooding topology  $F_t$ :

1. Remove the first node  $A$  with its flag  $F$  set to one from the candidate queue  $C_q$  if there is such a node  $A$ ; otherwise (i.e., if there is not such node  $A$  in  $C_q$ ), then remove the first node  $A$  from  $C_q$ . Add  $A$  into the flooding topology  $F_t$ .
2. If  $C_q$  is empty or all nodes are on  $F_t$ , then return with  $F_t$ .
3. Suppose that node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node  $A$  and not in the flooding topology  $F_t$  and  $X_1, X_2, \dots, X_n$  are in a special order considering whether some of them with  $F = 1$ . For example,  $X_1, X_2, \dots, X_n$  are ordered by the cost of the link between  $A$  and  $X_i$ . The cost of the link between  $A$  and  $X_i$  is less than the cost of the link between  $A$  and  $X_j$  ( $j = i + 1$ ). If two costs are the same,  $X_i$ 's ID is less than  $X_j$ 's ID. The cost of a link is redefined such that 1) the cost of a link between  $A$  and  $X_i$  both with  $F = 1$  is much less than the cost of any link between  $A$  and  $X_k$  where  $X_k$  with  $F = 0$ ; 2) the real metric of a link between  $A$  and  $X_i$  and the real metric of a link between  $A$  and  $X_k$  are used as their costs for determining the order of  $X_i$  and  $X_k$  if they all (i.e.,  $A, X_i$  and  $X_k$ ) have  $F = 1$  or none of  $X_i$  and  $X_k$  has

$F = 1$ .

4. Add  $X_i$  ( $i = 1, 2, \dots, n$ ) associated with previous hop  $PH=A$  into the front of the candidate queue  $C_q$ , and goto step 1.

A third algorithm for building a tree from node  $R$  as root with consideration of flag  $F$  starts with a candidate list  $C_q$  containing  $R$  associated with low order cost  $L_c=0$ , high order cost  $H_c=0$  and previous hop ID  $PH=0$ , and an empty flooding topology  $F_t$ :

1. Remove the first node  $A$  from  $C_q$  and add  $A$  into  $F_t$ .
2. If all the nodes are on  $F_t$ , then return with  $F_t$ .
3. Suppose that node  $A$  is associated with a cost  $C_a$  which is the cost from root  $R$  to node  $A$ , node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node  $A$  and not in  $F_t$  and the cost of the link between  $A$  and  $X_i$  is  $LC_i$  ( $i=1, 2, \dots, n$ ). Compute  $C_i = C_a + LC_i$ ,

- check if  $X_i$  is in  $C_q$  and if  $C_{xi}$  (cost from  $R$  to  $X_i$ )  $< C_i$ . If  $X_i$  is not in  $C_q$ , then add  $X_i$  with cost  $C_i$  into  $C_q$ ; If  $X_i$  is in  $C_q$ , then If  $C_{xi} > C_i$  then replace  $X_i$  with cost  $C_{xi}$  by  $X_i$  with  $C_i$  in  $C_q$ ; If  $C_{xi} == C_i$  then add  $X_i$  with cost  $C_i$  into  $C_q$ .
4. Suppose that node  $A$  is associated with a low order cost  $LC_a$  which is the low order cost from root  $R$  to node  $A$  and a high order cost  $HC_a$  which is the high order cost from  $R$  to  $A$ , node  $X_i$  ( $i = 1, 2, \dots, n$ ) is connected to node  $A$  and not in the flooding topology  $F_t$  and the real cost of the link between  $A$  and  $X_i$  is  $C_i$  ( $i=1, 2, \dots, n$ ). Compute  $LC_{xi}$  and  $HC_{xi}$ :  $LC_{xi} = LC_a + C_i$  if both  $A$  and  $X_i$  have flag  $F$  set to one, otherwise  $LC_{xi} = LC_a$   $HC_{xi} = HC_a + C_i$  if  $A$  or  $X_i$  does not have flag  $F$  set to one, otherwise  $HC_{xi} = HC_a$  If  $X_i$  is not in  $C_q$ , then add  $X_i$  associated with  $LC_{xi}$ ,  $HC_{xi}$  and  $PH = A$  into  $C_q$ ; If  $X_i$  associated with  $LC_{xi}'$  and  $HC_{xi}'$  and  $PH_{xi}'$  is in  $C_q$ , then If  $HC_{xi}' > HC_{xi}$  then replace  $X_i$  with  $HC_{xi}'$ ,  $LC_{xi}'$  and  $PH_{xi}'$  by  $X_i$  with  $HC_{xi}$ ,  $LC_{xi}$  and  $PH=A$  in  $C_q$ ; otherwise (i.e.,  $HC_{xi}' == HC_{xi}$ ) if  $LC_{xi}' > LC_{xi}$ , then replace  $X_i$  with  $HC_{xi}'$ ,  $LC_{xi}'$  and  $PH_{xi}'$  by  $X_i$  with  $HC_{xi}$ ,  $LC_{xi}$  and  $PH=A$  in  $C_q$ ; otherwise (i.e.,  $HC_{xi}' == HC_{xi}$  and  $LC_{xi}' == LC_{xi}$ ) if  $PH_{xi}' > PH$ , then replace  $X_i$  with  $HC_{xi}'$ ,  $LC_{xi}'$  and  $PH_{xi}'$  by  $X_i$  with  $HC_{xi}$ ,  $LC_{xi}$  and  $PH=A$  in  $C_q$ .
  5. Make sure  $C_q$  is in a special order. Suppose that  $A_i$  ( $i=1, 2, \dots, m$ ) are the nodes in  $C_q$ ,  $HC_{ai}$  and  $LC_{ai}$  are low order cost and high order cost associated with  $A_i$ , and  $ID_i$  is the ID of  $A_i$ . One order is that for any  $k = 1, 2, \dots, m-1$ ,  $HC_{ak} < HC_{aj}$  ( $j = k+1$ ) or  $HC_{ak} = HC_{aj}$  and  $LC_{ak} < LC_{aj}$  or  $HC_{ak} = HC_{aj}$  and  $LC_{ak} = LC_{aj}$  and  $ID_k < ID_j$ . Goto step 1.

### [A.3.](#) Connecting Leaves

Suppose that we have a flooding topology  $F_t$  built by one of the algorithms described above.  $F_t$  is like a tree. We may connect  $k$  ( $k \geq 0$ ) leaves to the tree to have a enhanced flooding topology with more connectivity.

Suppose that there are  $m$  ( $0 < m$ ) leaves directly connected to a node  $X$  on the flooding topology  $F_t$ . Select  $k$  ( $k \leq m$ ) leaves through using a deterministic algorithm or rule. One algorithm or rule is to select  $k$  leaves that have smaller or larger IDs (i.e., the IDs of

these  $k$  leaves are smaller/bigger than the IDs of the other leaves directly connected to node  $X$ ). Since every node has a unique ID, selecting  $k$  leaves with smaller or larger IDs is deterministic.

If  $k = 1$ , the leaf selected has the smallest/largest node ID among the IDs of all the leaves directly connected to node  $X$ .

For a selected leaf  $L$  directly connected to a node  $N$  in the flooding topology  $F_t$ , select a connection/adjacency to another node from node  $L$  in  $F_t$  through using a deterministic algorithm or rule.

Suppose that leaf node  $L$  is directly connected to nodes  $N_i$  ( $i = 1, 2, \dots, s$ ) in the flooding topology  $F_t$  via adjacencies and node  $N_i$  is not node  $N$ ,  $ID_i$  is the ID of node  $N_i$ , and  $H_i$  ( $i = 1, 2, \dots, s$ ) is the number of hops from node  $L$  to node  $N_i$  in the flooding topology  $F_t$ .

One Algorithm or rule is to select the connection to node  $N_j$  ( $1 \leq j \leq s$ ) such that  $H_j$  is the largest among  $H_1, H_2, \dots, H_s$ . If there is another node  $N_a$  ( $1 \leq a \leq s$ ) and  $H_j = H_a$ , then select the one with smaller (or larger) node ID. That is that if  $H_j = H_a$  and  $ID_j < ID_a$  then select the connection to  $N_j$  for selecting the one with smaller node ID (or if  $H_j = H_a$  and  $ID_j < ID_a$  then select the connection to  $N_a$  for selecting the one with larger node ID).

Suppose that the number of connections in total between leaves selected and the nodes in the flooding topology  $F_t$  to be added is  $N_{Lc}$ . We may have a limit to  $N_{Lc}$ .

#### Authors' Addresses

Huaimo Chen  
Huawei Technologies

Email: [huaimo.chen@huawei.com](mailto:huaimo.chen@huawei.com)

Dean Cheng  
Huawei Technologies



Email: dean.cheng@huawei.com

Mehmet Toy  
Verizon  
USA

Email: mehmet.toy@verizon.com

Yi Yang  
IBM  
Cary, NC  
United States of America

Email: yyietf@gmail.com