

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

H. Chen
D. Cheng
Huawei Technologies
M. Toy
Verizon
Y. Yang
IBM
July 2, 2018

OSPF Flooding Reduction
draft-cc-ospf-flooding-reduction-02

Abstract

This document proposes an approach to flood OSPF link state advertisements on a topology that is a subgraph of the complete OSPF 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 OSPF area, and can be used in both OSPFv2 ([[RFC2328](#)]) network and OSPFv3 ([[RFC5340](#)]) network.

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/>.

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 January 3, 2019.

Internet-Draft

OSPF Flooding Reduction

July 2018

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 | 2 |
| 2. | Problem Statement | 3 |
| 3. | Flooding Topology | 4 |
| 4. | Extensions to OSPF | 6 |
| 5. | Flooding Behavior | 8 |
| 5.1. | Nodes Perform Flooding Reduction | 8 |
| 5.1.1. | Receiving an OSPF LSA | 8 |
| 5.1.2. | Originating an OSPF LSA | 9 |
| 5.1.3. | An Exception Case | 9 |
| 5.1.4. | One More Note | 9 |
| 5.2. | Nodes Not Support Flooding Reduction | 10 |
| 6. | Security Considerations | 10 |
| 7. | IANA Considerations | 10 |
| 8. | Acknowledgements | 10 |
| 9. | References | 10 |
| 9.1. | Normative References | 10 |
| 9.2. | Informative References | 11 |
| Appendix A. | Algorithms to Build Flooding Topology | 11 |
| A.1. | Algorithms to Build Tree without Considering Others | 11 |
| A.2. | Algorithms to Build Tree Considering Others | 12 |
| A.3. | Connecting Leaves | 15 |
| | Authors' Addresses | 16 |

[1.](#) Introduction

For some networks such as dense Data Center (DC) networks, the

existing OSPF Link State Advertisement (LSA) flooding mechanism is not efficient and may have some issues. The extra LSA flooding consumes network bandwidth. Processing the extra LSA flooding, including receiving, buffering and decoding the extra LSAs, 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](#)]. A flooding reduction focusing on central computation of flooding topology is discussed in [[I-D.li-dynamic-flooding](#)]. This document proposes an approach to flood OSPF LSAs on a topology that is a subgraph of the entire OSPF topology per underline physical network, so that the amount of flooding traffic in the network is greatly reduced. The workload for processing the extra LSA flooding is decreased significantly. This would improve the scalability and speed up the network convergence, stable and optimize the routing environment.

This approach is flexible. It has multiple modes for computation of flooding topology. Users can select a mode they prefer, and smoothly switch from one current mode to another. The approach proposed is applicable to any network topology in a single OSPF area. It can be used in both a OSPFv2 network and a OSPFv3 network. The approach is backward compatible.

[2.](#) Problem Statement

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

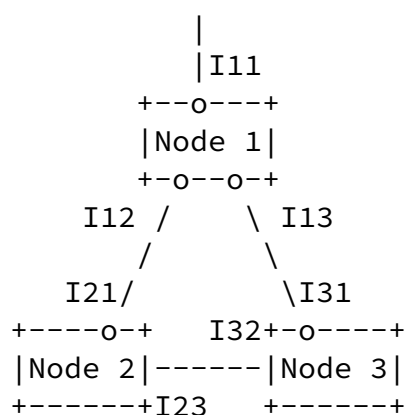
The problem explained above becomes more serious in OSPF networks

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.

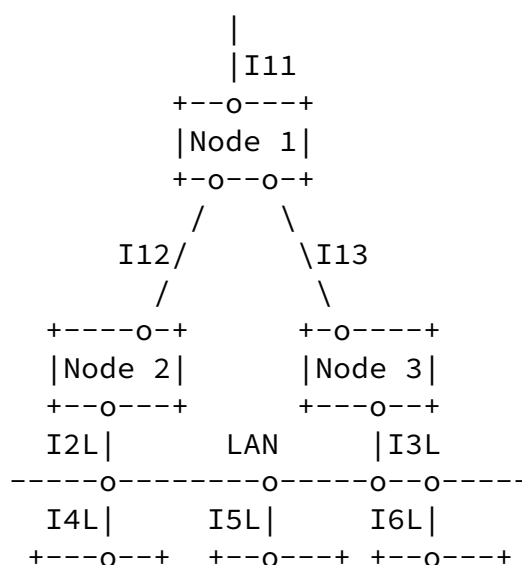
One example is as shown in Figure 1 (a), where Node 1, Node 2 and Node 3 are interconnected in a mesh. When Node 1 receives a new or updated OSPF LSA on its interface I11, it by default would forward to its interface I12 and I13 towards Node 2 and Node 3, respectively,

after processing. Node 2 and Node 3 upon reception of the LSA and after processing, would potentially flood the same LSA 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 LSA 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.



(a)



| | | |
|---------|---------|---------|
| Node 4 | Node 5 | Node 6 |
| +-----+ | +-----+ | +-----+ |

(b)

Figure 1

3. Flooding Topology

It is a norm that an OSPF node sending a received LSA and self-originated LSA to all its OSPF interfaces (except that where a LSA is received), as the reliable-flooding mechanism requires, i.e., any OSPF LSA would potentially traverses on each OSPF link in a given OSPF network topology, sometimes both directions. As demonstrated in [Section 2](#), dissemination over the entire OSPF network topology has drawbacks.

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

There are many different flooding topologies for a given OSPF 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 OSPF network topology. A few of them are listed below:

- o Central Mode: 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 Distributed Mode: Each node in the network automatically calculates a flooding topology by using the same algorithm (No flooding for flooding topology);
- o Static Mode: Links on the flooding topology are configured statically.

The minimum requirement for a flooding topology is all OSPF 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 OSPF 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 OSPF 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 OSPF node is dynamic in nature, that means when the OSPF'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.

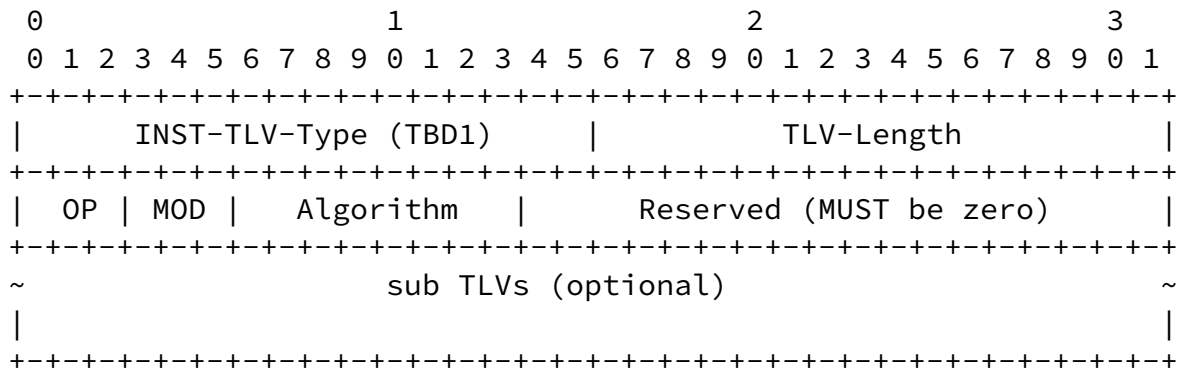
For reference purpose, some algorithms that allow OSPF nodes to automatically compute flooding topology are elaborated in [Appendix A](#). However, this document does not attempt to standardize how a flooding topology is established.

[4.](#) Extensions to OSPF

A couple of TLVs are defined in OSPF RI LSA [[RFC7770](#)]. One TLV contains instructions about flooding reduction, which is called Flooding Reduction Instruction TLV or Instruction TLV for short. This TLV is originated from only one node at any time. Another TLV

includes the information on flooding reduction of a node, which is called Flooding Reduction Information TLV or Information TLV for short. This TLV is generated by every node that supports flooding reduction in general.

The format of a Flooding Reduction Instruction TLV is as follows.



A OP field of three bits is defined in the TLV. It may have a value of the followings.

- o 0x001 (R): Perform flooding Reduction, which instructs the nodes in a network to perform flooding reduction.
- o 0x010 (N): Roll back to Normal flooding, which instructs the nodes in a network to roll back to perform normal flooding.

When any of the other values is received, it is ignored.

A MOD field of three bits is defined in the TLV and may have a value of the followings.

- o 0x001 (C): Central Mode, which instructs 1) the nodes in a network to select a leader node and a backup leader node; 2) the leader node in a network to compute a flooding topology and flood the flooding topology to all the other nodes in the network; 3) every node in the network to receive and use the flooding topology originated by the leader node.
- o 0x010 (D): Distributed Mode, which instructs every node in a

network to compute and use its own flooding topology.

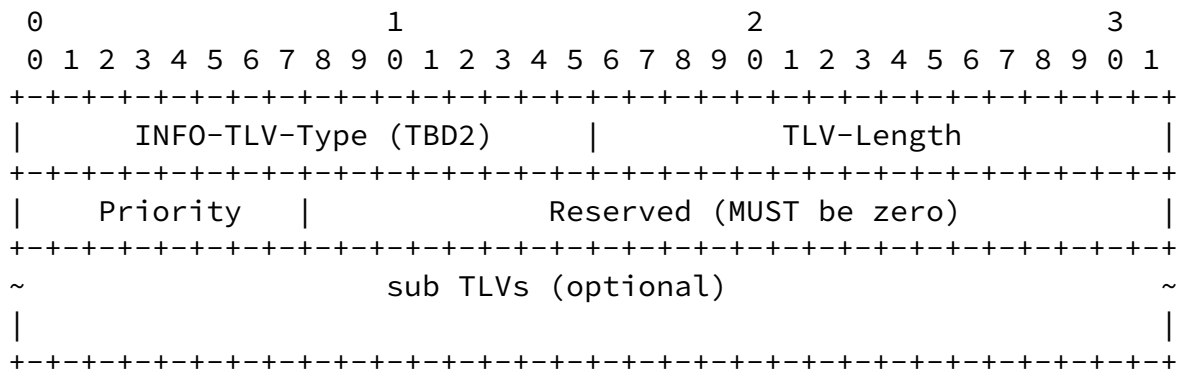
- o 0x010 (S): Static Mode, which instructs every node in a network to use the flooding topology statically configured on the node.

When any of the other values is received, it is ignored.

An Algorithm field of eight bits is defined in the TLV to instruct the leader node in central mode or every node in distributed mode to use the algorithm indicated in this field for computing a flooding topology.

Some optional sub TLVs may be defined in the future, but none is defined now.

The format of a Flooding Reduction Information TLV is as follows.



A Priority field of eight bits is defined in the TLV to indicate the priority of the node originating the TLV to become the leader node in central mode.

Some optional sub TLVs may be defined in the future, but none is defined now.

[5.1.](#) Nodes Perform Flooding Reduction

This section describes OSPF flooding behavior for OSPF nodes that perform flooding reduction described in this document. The flooding behavior for these nodes differs from that as specified in OSPFv2 ([RFC2328]) and OSPFv3 ([RFC5340]). [Section 5.1.1](#) describes the flooding behavior when an OSPF node receives an OSPF LSA from one of its interfaces, and [Section 5.1.2](#) describes the flooding behavior for LSA originated by itself.

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

It assumes that the OSPF node of which the flooding behavior is described below is on the flooding topology, i.e., the node and at least one of its OSPF 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 OSPF 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 OSPF LSA

The flooding behavior when an OSPF node receives a newer OSPF LSA that is not originated by itself from one of its OSPF interfaces is as follows:

1. The LSA is received on a link that is on the flooding topology. The LSA is flooded only to all the other interfaces that are on the flooding topology.
2. The LSA 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 LSA sending neighbor does not support the OSPF flooding reduction; it can also happen as the receiving link is a

broadcast-type interface. The LSA is flooded only to all other interfaces that are on the flooding topology.

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

Note before forwarding a received LSA, the OSPF node would do the normal processing as usual.

[5.1.2.](#) Originating an OSPF LSA

The flooding behavior when an OSPF node originates an OSPF LSA is as follows:

1. If it is a refresh LSA, i.e., there is no significant change contained in the LSA comparing to the previous LSA, the LSA is transmitted over links on the flooding topology.
2. Otherwise, the LSA is transmitted to all OSPF 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 LSA quickly.

[5.1.3.](#) An Exception Case

In [Section 5.1.1](#) and [Section 5.1.2](#), there are times when an OSPF node sending out a LSA to an interface on the flooding topology detects a critical interface or node failure. A critical interface is an interface on the flooding topology and is the only connection among some nodes on the flooding topology. When this interface goes down, the flooding topology will be split. Note the flooding topology was pre-computed/pre-constructed; but if at the time a critical interface or a node goes down before a re-newed flooding topology can be computed/constructed, the OSPF node MUST send out the LSA to all interfaces (except where it is received from) as a traditional OSPF node would do. This handling is also taking place if there are more than one interfaces or nodes on the existing flooding topology fail, i.e., if more than one interfaces or nodes on the flooding topology fail, the OSPF 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 OSPF node sends out a LSA on an interface on its flooding topology follows the

specification in OSPFv2 ([RFC2328]) and OSPFv3 ([RFC5340]). This means on a local LAN, all other OSPF nodes will receive the LSA.

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

The LSA flooding behavior of OSPF nodes that do not support flooding reduction as described in this document MUST follow that as specified in OSPFv2 ([RFC2328]) and OSPFv3 ([RFC5340]).

[6.](#) Security Considerations

This document does not introduce any security issue.

[7.](#) IANA Considerations

Under Registry Name: OSPF Router Information (RI) TLVs [RFC7770], IANA is requested to assign two new TLV values for OSPF flooding reduction as follows:

| TLV Value | TLV Name | reference |
|-----------|-----------------|---------------|
| TBD1 | Instruction TLV | This document |
| TBD2 | Information TLV | This document |

[8.](#) 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.

[9.](#) References

[9.1.](#) Normative References

- [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](https://www.rfc-editor.org/info/rfc2328), DOI 10.17487/RFC2328, April 1998, <<https://www.rfc-editor.org/info/rfc2328>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", [RFC 5340](https://www.rfc-editor.org/info/rfc5340), DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.

Chen, et al. Expires January 3, 2019 [Page 10]

Internet-Draft OSPF Flooding Reduction July 2018

- [RFC7770] Lindem, A., Ed., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", [RFC 7770](https://www.rfc-editor.org/info/rfc7770), DOI 10.17487/RFC7770, February 2016, <<https://www.rfc-editor.org/info/rfc7770>>.

[9.2.](#) Informative References

- [I-D.li-dynamic-flooding]
Li, T. and P. Psenak, "Dynamic Flooding on Dense Graphs", [draft-li-dynamic-flooding-05](#) (work in progress), June 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-06](#) (work in progress), June 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 Others

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 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. 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 Others

An algorithm for building a tree from node R as root with consideration of others's support for flooding reduction 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 that supports flooding reduction 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 that support flooding reduction (. 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 support flooding reduction 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) support flooding reduction or none of X_i and X_k support flooding reduction.

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 others' support for flooding reduction 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 that supports flooding reduction 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 support flooding reduction. 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 support flooding reduction is much less than the cost of any link between A and X_k where X_k does not support flooding reduction; 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) support flooding reduction or none of X_i and X_k supports flooding reduction.

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 others' support for flooding reduction (using flag $F = 1$ for support, and $F = 0$ for not support in the following) 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 ($i=1, 2,$

..., 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

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

