Meshed Trees for Resilient Switched Networks

Peter Willis[1], Nirmala Shenoy[2]
Dept. of Information Sciences and Technologies,
Golisano College of Computing and Information Sciences
Rochester Institute of Technology, Rochester, New York, USA
e-mail: [1]pjw7904@rit.edu, [2]nxsvks@rit.edu

***Abstract*— Layer 2 (L2) protocols are fundamental to all network communications. Loop-avoidance in L2 operations is essential for forwarding broadcast frames without them looping throughout network. Loop-avoidance protocols construct a logical tree on the meshed topology, normally used to provide path redundancy in switched networks. Repairing the tree on topology changes results in expensive network downtime and is major challenge faced in L2 networks. In this article, we present the Meshed Tree Protocol (MTP) based on a novel Meshed Tree Algorithm (MTA) as a clean-slate approach to loop avoidance in switched network. MTP leverages the connectivity in the meshed topology to pre-construct several trees from a root. Multiple backup paths are in readiness to takeover in the event of failure of the main path for fast convergence. We limit our work in this article to a comparison of a coded prototype implementation of MTP vs. the Rapid Spanning Tree Protocol (RSTP) in L2 customer networks. The evaluation was conducted on the GENI (Global Environment for Network Innovation) testbed.**

Keywords-Meshed Trees; Pre-constructed Paths; Path Vector VIDs; Hysteresis in Failure Detection.

I. INTRODUCTION

High-performance switched networks are in great demand with the growth in L2 Customer (C), Service Provider (SP) and Backbone Provider (BP) networks. Meshed topologies are adopted in switched networks to provide path redundancy. Consequently, handling of broadcast frames poses a challenge. When a switch receives a broadcast frame on a port it forwards it on all other ports (except the port it was received on). Because there are physical loops in the network due to the meshed topology, this can result in broadcast frames looping in the network infinitely and crashing the network. For this purpose, it is important to forward brought broadcast frames on paths that do not loop. The traditional approach is to construct a logical tree on the physical meshed topology and allow broadcast frames to be forwarded along the tree paths. To construct a logical tree on the switched network, loop avoidance protocols are used. Loop-avoidance protocols use tree algorithms such as Spanning tree and Dijkstra tree to construct the logical tree. Tree algorithms allow construction of a single tree from a root. Hence on a link or switch failure when a tree branch fails, protocols based of these algorithms must reconstruct/repair the tree. As a result, the convergence latency in the event of a network component failure can be high. This is a setback for applications running on switched networks that desire high availability.

Constructing a single logical tree on a meshed topology logically sacrifices the rich path redundancy in the meshed topology. We propose a novel meshed trees algorithm (MTA) that allows construction of multiple trees from a single root. The branches from the multiple trees mesh at the switches thus keeping the redundant paths in readiness and the failover in the event of a (currently used) path failure is immediate. We further propose a novel virtual identifier to pre-construct and maintain the multiple trees. This simplifies the meshed tree protocol implementation significantly, making the protocol lightweight and robust.

In this article, we limit our performance comparison of MTP vs RSTP to highlight the significant performance improvement (several magnitudes) in terms of convergence on link failures that can be achieved with a simple and robust protocol, especially in L2 customer networks. RSTP is a standard protocol and its code is available on the GENI testbed, hence it is readily available to compare the two working prototypes. RSTP also serves as a reference. We collect measurements of an implementation of MTP and RSTP on the GENI testbed [10], for multiple network topologies. A detailed analysis and study of MTP vs RSTP convergence process considering tree construction and recovery on failures, based on message exchanges and port role changes is provided. Several sets of test cases were evaluated. The tree construction process with MTP and RSTP are highlighted and contrasted to explain the difference in operational complexity between the two protocols and to justify the performance improvements with MTP. Future work will cover a study of MTP vs IS-IS based loop-avoidance protocols for L2 SP and BP networks.

The rest of the paper is structured as follows. In Section II, we discuss background work on loop-avoidance protocols, primarily focusing on standards, followed by an introduction to meshed trees. We discuss IS-IS based protocols, the latest loop avoidance protocol standards to address convergence delays experienced by spanning tree protocols, only to highlight their complexity and limitations. We also introduce and compare meshed trees with Dijkstra and spanning trees. Section III describes a meshed trees implementation in a switched network with an example highlighting several of its attributes. Section IV focusses on the performance metrics that will be studied and their significance. Section V provides details on protocol evaluations, and tools and techniques to assess performance. Subsections discuss in detail the performance in three different topologies for multiple test cases. The failures were limited to single link failures and the relative position in the tree which plays an important role in the

 ISBN: 978-1-61208-786-3

convergence latency experienced in switched networks. Section VI provides a summary of the results highlighting the significant difference protocol recovery latency and messages among the 2 protocols. Section VII provides a brief conclusion and future work.

II. BACKGROUND

In this section, we present the two major categories of loop avoidance protocols for use in switched networks. Though other alternatives have been proposed we focus on these two categories as they are widely used and serve as a standard for performance comparison. They are:

1. The Spanning Tree Protocol family
2. Dijkstra's Tree Algorithm based Protocols

A. The Spanning Tree Protocol (STP) Family

This includes STP and its faster version – Rapid STP (RSTP), for construction of a single tree in a local area network (LAN). (We limit the discussions in this article to a single LAN). STP has been an IEEE standard since 1998 [8]. In STP, switches exchange Bridge Protocol Data Units (BPDUs) to decide on a logical spanning tree. Roles are assigned to ports so they can allow or block frames. Bridge Medium Access Control (MAC) addresses and port numbers are used to break ties during spanning tree construction. STP recovery process on topology changes result in transients and high recovery delays, as STP uses many timers. RSTP IEEE 802.1w [2] avoids delays incurred due to extensive timer usage by STP and speeds up convergence through fast exchange of proposal and agreement message among switches. RSTP further reduced convergence times by holding a port in readiness if the best port to reach the root bridge (the root port) fails.

B. Dijkstra's Tree Algorithm based Protocols

For high performance networks, such as L2 SP and BP networks, the delays incurred with RSTP were unacceptable. As a consequence, Inter-System Inter-System (IS-IS) based Layer 3 routing solutions that use Dijkstra trees were introduced into L2 operations to improve path and root switch failure resiliency. The IS-IS based solutions construct shortest paths trees from every switch (as a root), to cut down root reelection time. Loop avoidance protocols that use Dijkstra's algorithm are TRILL (Transparent Interconnection of Lots of Links) on RBridges (Router Bridges) [4][5] and Shortest Path Bridging (SPB) [7]. Radia Perlman, the inventor of STP, introduced TRILL [5] as an Internet Engineering Task Force (IETF) effort, as it operates above L2. The TRILL protocol uses the IS-IS routing protocol to take advantage of the numerous trees constructed using Dijkstra's algorithm. Similar to TRILL, SPB, is an IEEE effort that introduced IS-IS link state routing into L2 [7]. Using IS-IS links state routing incurs the following overhead/limitations:

- IS-IS messages are encapsulated in L2 or TRILL messages adding to operational overhead and complexity.
- IS-IS cannot guarantee true loop-freedom; a hop count is included to track and discard looping frames.
- Dijkstra's tree construction is computation intensive [9] and not suited for fast convergence in dynamic networks. During the re-computation time on topology changes, frame delivery is not guaranteed.
- Any link state change must be propagated to all switches, which then wait for a settling time to compute new Dijkstra trees. During this period, frame forwarding is unreliable.
- Reverse congruency requires all switches to compute Dijkstra trees from all other switches, so they can use the same ports in source address tables to forward frames to end devices connected to the other switch. This multiplies the operation complexity and overhead by the number of switches and is a scalability issue.

C. The Need for a New Approach

Current loop-avoidance protocols construct a single logical tree to avoid looping of broadcast frames in meshed networks. This is true for protocols based on spanning trees or on Dijkstra trees. In the event of a single link failure both categories of protocols require dissemination of this information to all switches so they can reconstruct/repair the trees. Repairing trees takes time which is the reason for the convergence latency faced in current loop avoidance protocols. We propose a clean slate approach, which uses Meshed Tree Algorithm that enables construction of multiple trees from a single root.

D. The Meshed Tree Algorithm

Instead of a single spanning tree or shortest path tree as possible with current tree algorithms, MTP [1] adopts a new MTA to compute multiple trees from a root. The multiple paths constructed mesh at the switches and hence are called meshed trees. MTP leverages the connectivity in meshed networks to provision redundant paths from every switch to the root. The redundant paths are in readiness to takeover in the event of a link or main path failure. Maintaining multiple branches would seem to add to the operational complexity, but we use a novel Path Vector virtual ID technique to construct and maintain meshed trees resulting in an extremely light weight protocol, as discussed in Section III.

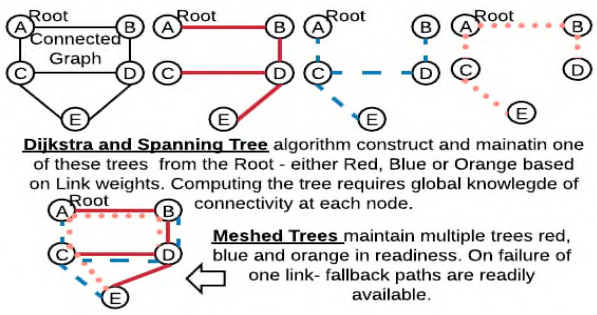

Figure 1. Meshed Trees vs Spanning/Dijkstra Tree

The Meshed Tree Algorithm – is the first of its kind that allows preconstruction of multiple tree branches from a single root. We describe the concept of Meshed trees using Figure 1. In Figure 1, we compare spanning /shortest path tree constructed using Dijkstra and Spanning Tree

 ISBN: 978-1-61208-786-3

algorithms to meshed trees. On the top left is a 5-node, 6-edge connected graph. Using either Dijkstra's or Spanning Tree algorithm, one of three trees, i.e. red, blue or orange tree, can be constructed – the outcome is always one tree, from the root node A. The bottom left shows a *meshed tree,* in which all three trees (red, blue and orange) co-exist allowing each node to have multiple paths to the root node A and to reach any other node. In the event a path fails, another path is readily available.

III. The Meshed Tree Protocol

The Meshed Tree Protocol builds fault tolerance in switched networks by computing multiple tree paths from each switch to the root. This ensures that there are alternative paths in readiness on switch or link failures. The number of alternative paths that a switch records can be set as a parameter. MTP computes and maintain these paths with very low overhead and operational complexity using virtual identifiers (VIDs). A VID at a switch identifies a path to the root. A switch acquires and stores multiple VIDs. Hop count is inherent in the VID. VIDs also aid in loop-detection, cutting down on processing time and operational overhead significantly. VIDs are discussed in the Sections below.

A. Meshed Tree Protocol Overview

In this section we explain the construction of meshed trees by MTP in a switched network. We start by designating a root switch it and assigning it a unique VID. Rationale for Root switch designation is discussed next.

1) Root Switch Designation

A switch is designated to be the root. We decided to designate a root to avoid root election delays. However, in the event of failure of this root, we need another root to take over. In spanning tree-based protocols, on the failure of the root, a root election is conducted to elect the next root, subsequent to which the spanning tree is constructed. The root election process incurs heavy delays. Spanning tree protocols, further bias the root election by setting the switch priority such that the switch with higher capacity gets elected as root. This is necessary as the traffic carried by the root switch is significantly higher than other switches. In the case of Dijkstra's algorithm based protocols, every switch is a root, which is an overkill as the number of trees constructed equals the number of switches.

With Mesh Tree Protocol, we designate an optimal number of switches to be roots – one is the primary root, the next is the secondary root and so on. The number of meshed trees constructed equals the number of roots designated. Depending on the network availability needs and based on the services they support, the number of roots can be optimized. The rational for using an optimal set of roots is to avoid root election delay if we used one root, as in spanning tree protocols, and at the same time avoid excessive computations required in computing trees from every switch as with Dijkstra algorithm based protocols. In this article, we restrict our presentation and analysis to a single rooted meshed tree. The goal is to demonstrate fast convergence and quick recovery of broadcast frame forwarding (due to the pre-constructed backup paths) in the event of link failures.

2) Meshed Tree Construction

Meshed trees are constructed and maintained at the switches where the tree information is stored as VIDs. As previously stated, a root is designated and is assigned a VID - say '1'. All other switches acquire VIDs as MTP is executed in the switches. A VID stored at a switch (not the root) is a concatenation of numbers where the first number is the VID of the root. The numbers following the root VID are the port numbers of switches that identify a path to the root. We use a dotted decimal notation for this purpose. For example, a switch – say S1 connected on port 2 of the root switch, will acquire a VID of 1.2. A switch S2 connected on port 3 of switch S1, will acquire a VID of 1.2.3 where the VID 1.2.3 defines the path from switch S2 to the root via switch S1. We next describe the process of using VIDs to define multiple tree paths from every switch to the root using an example 5-switch network.

We use Figure 2 to explain meshed tree construction in a 5- switch network. Figure 2A shows the 5-switch topology. Switch port numbers are noted besides each switch. In Figure 2B, a pink logical tree starting at the root and reaching all switches is shown. We now walk through the process of how the pink logical tree was constructed. The tree construction begins from the root. The root offers (through an advertisement) VID 1.1 on port 1 (appending the outgoing port number 1 to its VID 1). Switch S1 that receives this offer accepts and joins the pink tree by storing VID 1.1. Next, S1 offers VID 1.1.2 to S3 on its port 2. S3 offers VID 1.1.2.2 to S2, and 1.1.2.3 to S4 (as S2 is connected at port 2, and S4 at port 3, of S3). VIDs 1, 1.1, 1.1.2, 1.1.2.2, 1.1.2.3 define the pink logical tree.

In Figure 2C, a similar procedure is used to build the orange tree, from the Root via switch S2 and this tree is defined by VIDs 1, 1.2, 1.2.2, 1.2.3, 1.2.2.1. Figure 2D shows how both trees, are maintained at the switches by *simply storing both sets of VIDs*.

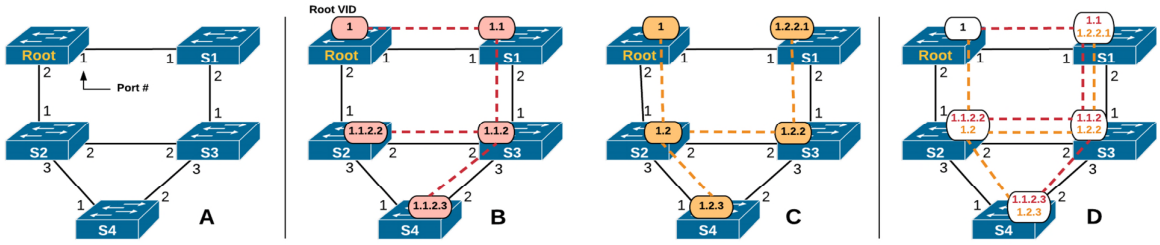

Figure 2. Meshed Trees in Switched Networks

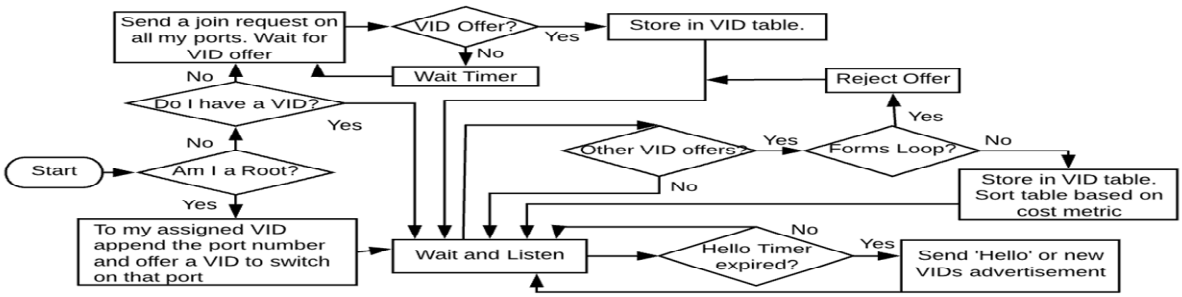

Figure 3. Flow Chart for Simplified Meshed Tree Construction in Switched Network

MTP uses messages that advertise VIDs, and a switch that accepts a VID informs the upstream switch of the VIDs it accepts. We do not include the protocol implementation details in this article, but restrict our presentation to its operations. On startup as only the root switch has a VID, switches that do not have a VID will issue a join-request message on their ports. Figure 3 is a flow chart used at the switches to construct meshed trees.

When MTP is started in a switch, the switch first checks if it is designated root. If this is true (it will have an assigned VID), the root will send a VID advertisement on all its ports. The VID advertisement on each port will be different because when advertising its VID it will append the port number on which the VID advertisement will be sent. Switches that are not designated roots will send a join-request to receive VID advertisements from its neighbors. Every switch that acquires a VID will advertise on all other ports except the port on which the VID was acquired. Thus, switches receive multiple VID advertisements and are able to select VIDs based on the path metric and store them in a VID table, in order of preference. Before accepting a VID, switches do a loop check which is explained below. MTP uses the hello timer to keep its neighbors informed about its current active status. Any changes to the VID tables are advertised.

3) Loop Detection and Preemption

We explain loop detection and pre-emption using an example from Figure 2D. Let S3 offer VID 1.1.2.1 to switch S1. S1 compares the offered VID with its current VIDs, 1.1 and 1.2.2.1. Its current VID 1.1 is a *proper prefix* of the offered VID 1.1.2.1, so S1 knows it is its own ancestor in the offered path to the root. S1 thus detects a loop and will not accept this VID. (Note - the number of digits in the VID is direct measure of hop count).

4) The Distributed Approach to Tree Construction

The process of selecting the best set of VIDs is decided by each switch independently based on the advertisements it hears from its neighbors. On receiving multiple VIDs, a switch stores them in order of preference based on hop count, path cost, or any other metric. In this article, hop count is used. To limit the tree meshing and conserve on memory usage the number of VIDs stored by a switch is limited. *The limited number of VID's in this implementation is 3*. The fact that each switch independently decides on the VIDs it stores based on a preference criterion makes the protocol robust.

5) The Broadcast Tree

Of the multiple tree paths from each switch to the root, at any time only one path will be used to avoid looping of broadcast frames. For this purpose, a switch declares one of its VIDs as Primary VID (PVID). The PVID is decided based hop count in this study. Thus, in this study, switches use the shortest VID from the set of VID's that they have stored, as the PVID. A PVID and the PVID port provide the lowest cost path from a switch to the root.

A switch also records its neighbors who have chosen it as the "PVID parent", noting their PVID as a child-PVID (CPVID) and their port of connection. While a PVID and its port of acquisition connect a switch to the Root, a CPVID port and its port of connection provides the link between a downstream switch and an upstream switch. The PVID, CPVID ports map out the broadcast tree to reach every switch – the broadcast tree is the tree that spans all switches and is used to forward the broadcast frame. We use Figure 2D to explain the broadcast tree defined by MTP. In Figure 4 we show Figure 2D only with the PVID selected by each switch, its port of acquisition and the CPVID port recorded by a switch and the port on which the CPVID was issued. In Figure 4, the connected CPVID and PVID ports using green arrows represent the broadcast tree. Note that the broadcast tree reach is limited to the switches and not extended to the links between the switches. This is in line with the later loop avoidance protocols that do not support hubs or shared media: example IS-IS based protocols. This further simplifies the tree construction as compared to spanning tree protocols.

6) Reducing Failure Detection Time

Failure detection time is one of the major contributors to convergence latency in most routing and loop-avoidance protocols. To reduce failure detection, Bidirectional Failure Detection (BFD) protocols was introduced [13]. BFD can be invoked by any protocol or application that

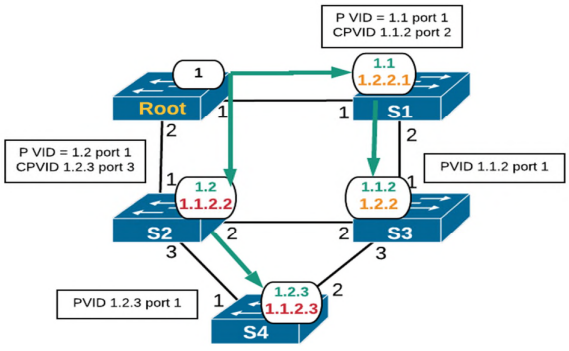

Figure 4. Broadcast Tree for Meshed tree in Figure 2D

requires faster notification on connection failures. It is implemented in the forwarding engine of the system. The idea behind BFD is similar to a 'hello protocol' and it sends hello messages at a higher frequency. However, a flapping interface can generate excessive change notifications in the system resulting in an unstable system. BFD introduces dampening [14] for this purpose, where the client application is not informed if the interface is flapping. *However, during the time that an interface is flapping, frames being forwarded or received via that interface may not be forwarded correctly.*

RSTP did not adopt BFD and instead uses a hello timer of 2 seconds and a dead time interval which is 3 times the hello timer – these are the default settings. This results in a failure detection time of anywhere between 5 to 7 seconds. In the set of studies presented in this article, we used these default settings as they serve as a reference. However, we separately recorded the failure detection time and the protocol recovery time to highlight the significant role played by protocols in the recovery time experienced in networks. It is the protocol recovery time and the messages exchanged during convergence that we compare primarily between RSTP and MTP.

7) Failure Detection with Hysteresis in MTP

The VID based approach in MTP and the storage of multiple VIDs provides a mechanism for us to speed up failure detection with MTP without the need for BFD. This approach also avoids the flapping interface problem. Because there are multiple pre-constructed paths which are identified by the VID's, in the event of a single missing hello message from a neighbor switch, a switch that has a VID derived on the port (of failure) removes this VID immediately into a quarantine table. The switch falls back to the next VID in its VID table. The VID changes (deletes) are advertised in the switched network and other switches if required update their VID table. However, if the port comes up active, this switch will wait for 'n' consecutive hello messages before re-instating the deleted VID from the quarantine table. Maintaining the deleted VID in the quarantine table helps to identify a recovering VID and avoid impacts of interface flapping – we call this the hysteresis approach. The value of 'n' was set to 3 in the studies. This avoids the flapping interface problem and hello messages between switches could be exchanged at a higher frequency. Thus MTP provides an efficient solution to the flapping interface problem.

In this experimental study, we set the hello interval to 1 second and the dead time interval to 2 times the hello interval. The hello interval and dead time interval can be further adjusted to speed up failure detection. The hello messages used in MTP are single byte messages and incur very little overhead- considerably less than BFD messages. Tuning the hello interval and a dead time interval of MTP would be a more desirable approach as there is no need to set up communication between MTP and BFD and also establishing sessions between every pair of switches – which is typically the approach used when BFD is used for failure detection.

With MTP, we are able quarantine the deleted VID and fallback on the next VID as all these VID's are precomputed and stored in the VID table. No re-computation is required on a topology change and MTP bypasses this delay completely. This is the first protocol that can support fast failure detection using a hysteresis approach.

IV. PERFORMANCE METRICS

The performance metrics analyzed are 1) protocol recovery delay, 2) failure detection delays 3) number of messages exchanged during convergence and the 4) number of port role /states changed (only for RSTP). Metrics 3 and 4 provide a means to assess a protocol's processing needs on topology changes. Typically:

Convergence Delay = Failure Detection Delay + Change Dissemination Delay + Recovery Delay

Dissemination Delay + Recovery Delay = Protocol Recovery Delay.

Protocol Recovery Delay depends on a protocol's recovery process subsequent to failure detection. With RSTP, the protocol recovery latency depends on network size, connectivity, the point of failure on the logical tree i.e. the relative position of the failure point with respect to the root and also on the port role of the failed port. RSTP recovery delay includes dissemination and tree recovery. MTP dissemination and recovery of tree proceed simultaneously. MTP recovery latency has low dependency on the network size/connectivity, the tree pruning is done with minimal number of messages and is significantly faster than RSTP.

A. Protocol Recovery Delay Contributors

In RSTP, in the case of a designated port failure, the switch on the other end of the link connected to the designated port must wait for failure detection before it can send topology change notifications. As per standards specifications, when a designated port fails, the port roles and states stabilize after a handshake between the node whose designated port failed, and any downstream switches connected on other designated ports. Proposal agreements speed up RSTP convergence – but these handshake messages must go down the branch(es) and switches are in 'discarding' state until a concurrence of port role changes arrives from downstream switches. When a root port fails, that switch immediately falls back on the alternate port, if any, else it assumes it is the root switch and messages with its neighbors for a quicker resolution of the tree. Besides, when a root port fails and if there is an alternate port, this port becomes the root port and initiates a TCN. Else it is initiated by a node that receives the changed BPDU from the node with the failed root port. In larger diameter topology this takes time. These are discussed under the protocol evaluation Section.

With MTP, on the deletion of a lost VID, which is immediate on failure detection, a delete message is sent out on all control ports of this switch (i.e. ports connecting to other switches running MTP). The switches that receive the delete message, remove any VIDs derived from the deleted VID and further propagate the message. Frame forwarding is impacted, only if the PVID in a switch is deleted. This can be noted in the results that we recorded during the prototype evaluations.

 ISBN: 978-1-61208-786-3

Using Figure 5, we explain the tree pruning process with MTP, on topology changes. In Figure 5, the link between the root switch and switch S1 is failed at port 1 of switch S1. On the failure of interface (port) 1, switch S1 deletes VID 1.1 acquired on port 1. It then sends out a delete message on its port 2 (containing the deleted VID 1.1). Switch S3 deletes its primary VID 1.1.2, moves its VID 1.2.1.2 as its PVID and sends out a delete message on its ports 2 and 3. Switch S2 deletes VID 1.1.2.2 that is derived from 1.1.2, but this has no change on the PVID. Similarly switch S4 deletes VID 1.12.3. Note the change in the broadcast tree (indicated with green arrows in Figure 5) –was achieved with three messages, as seen in Figure 5.

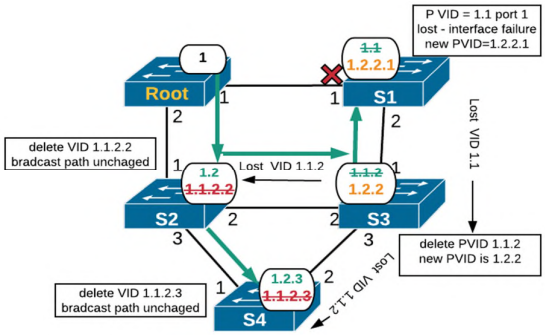

Figure 5. Meshed Tree Pruning on Link Root-B1 Failure

There is only one PVID change – thus, disruption to the broadcast tree is minimal and tree convergence is very fast.

V. Prototype Evaluations

The hardware for the implementation of MTP was acquired using GENI testbeds [10]. GENI is an open infrastructure for at scale networking and distributed systems research and education that spans the United States. It provides the infrastructure needed to carry out networking research. A set of compute resources like switches and network links to connect them can be acquired from GENI to set up the desired topology. RSTP is available on Open View switches at certain GENI sites and has been used in this performance study. MTP code was written in the C language and tested and deployed on GENI switches. The computer resources used from GENI ran Linux distributions. We used the Linux kernel networking stack to create raw Ethernet frames to carry MTP messages. The type field in the Ethernet frame was set to an unused number and messages sent by MTP would be picked up at Layer 2 and delivered to MTP processes at a receiving switch.

Identical topologies were used to evaluate the two protocols. To emulate identical operational conditions, we biased a switch's priority in networks running RSTP so it would get elected the root. An identical positioned switch in networks running MTP, was designated as the root switch. We used three different sized topologies: a simple 5-node 2-loop topology, a moderate 8-node 4-loop topology, and a more connected 17-node topology, with 7-hop diameter – the max specified in IEEE 802.1 standards [2]. This allowed us to assess and understand the protocol operation in a simple topology and study how the protocol scaled with increased network size and in more connected topologies.

A. Methodology

Tshark [11] was installed at all active GENI node interfaces to capture RSTP Bridge Protocol Data Units (BPDUs). RSTP operational states were logged at every node. The logs and captures were scanned to collect BPDUs generated and port roles/state changes as RSTP converged on topology changes. Chrony [12] was installed in all nodes to ensure the clocks on the nodes were consistent and time drifts minimal. GENI allows recording timestamps to an accuracy of 1 millisecond (ms). MTP code recorded events using the system's timer, with a timing accuracy of microseconds. Automation scripts were written in Python, to

- upload and execute the code in the GENI nodes,
- continuously collect the results into log files as the protocol is running at the GENI nodes,
- transfer the log files into our local system, and scan the log files to collect relevant data,

The automation scripts were written to repeat the experiment in each topology 5 times. The averages of these 5 runs were then recorded in the tables provided below.

B. Small Topology (5 switches)

In tests running RSTP, the *Convergence Time* (CT) is the time when a port is brought down to the time when ports that are changing roles and states in all switches settle down to their final roles and states. The *Failure Detection Time* (FDT is the time a switch across from the failed port recognized the failure and initiates action. The *Protocol Recovery Time* (PRT) is the time taken by the protocol to recover from the failure and converge after failure detection. Thus, CT is FDT plus PRT. In certain cases, we noticed that Topology Change Notifications (TCN) continue even after the port roles and states (PRS) stabilized. This is because, switches are required to send TCNs for a duration of tcWhile [8]. This was discounted in the calculations.

The RSTP 5-node topology with port roles (R for Root port, D for Designated port and A for Alternate port) are shown in Figure 6. The performance recorded in Table 1 reflect the dependencies on the port roles of ports that were failed noted in column 2 from Table 1. For the 5-node topology the test cases are noted in column 1 from Table 1. Single link failures were introduced by disabling a port in a switch. S1(1) indicates failed port 1 in switch S1. The role of the disabled port is noted in column 2 - D for Designated port and R for Root. PRS changes and TCN message exchanges that happen during convergence are noted. Under the FDT column we also note the switch that initiated change notifications.

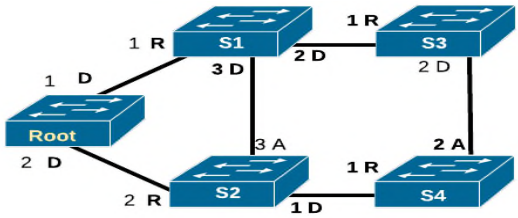

Figure 6. RSTP 5 Node Topology with Port Roles

1) Convergence Process in 5-node RSTP Topology

Root Port Failures: This happens in cases 2, 5, 7. Note that there is no FDT time, as the switch with the failed port initiates TCNs immediately. However, in case 2 the PRT was recorded as 3.523 seconds with 13 port role and state changes, (which indicates message exchanges such as proposal-agreements among switches) and 20 TCNs. In case 5, the PRT was recorded as 18 ms, with 5 port role/state changes and 26 TCNs. Under case 7, the PRT is very low, as this required S4 to fallback on its alternate port. The failure was at the edge of the tree – and this did not require TCN dissemination.

Designated Port Failures: Cases 1, 3, 4, and 6 relate to designated port failures. In all these cases the PRT was recorded to be around 3 seconds and TCNs varied from 20 to 30 messages. In certain cases the PRS were as low as 3 while we recorded 9 PRS under case 1.

2) Convergence in 5-switch MTP Topology

The broadcast tree established by MTP is given by the PVID and CPVID ports as discussed using Figure 4. In Figure 7, the PVID and CPVID ports are shown for the 5-node topology. Ports identical to the ones failed in the RSTP topology were failed in the MTP network. With MTP, in certain cases (e.g. 3 and 5 in Table 2), failing a port has no impact on the broadcast tree. *This is because the MTP broadcast tree stops at the switches and is not extended to links.*

TABLE I. RSTP CONVERGENCE IN A 5-NODE TOPOLOGY

Case	Failed Port - Role	FDT-initiated	PRT	PRS	TCNs
1	Root(2) ---D	5.115s by S2	**3.519s**	9	24
2	S1(1) -----R	0s by S1	**3.523s**	13	20
3	S1(3) -----D	4.030s by S2	**2.999s**	3	16
4	S1(2)------D	4.810s by S3	**3.018s**	8	25
5	S3(1) ----R	0s by S3	**18ms**	5	26
6	S3(2)------D	4.733s by S4	**3.164s**	3	18
7	S4(1) -----R	0s by S4	**9ms**	5	0

R- Root port, D- Designated port, FDT – Failure Detection time, PRT – Protocol Recovery Time, CT- Convergence time, PRS – Port/Role State Changes, TCN – Topology Change Notifications

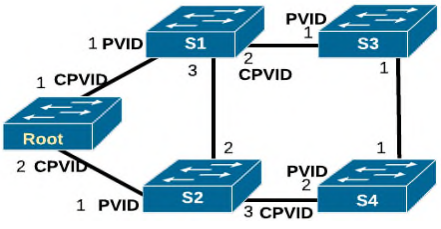

Figure 7. MTP 5 Node Topology with PVID, CPVID

TABLE II. MTP CONVERGENCE IN A 5-NODE TOPOLOGY

Case	Failed Port – (C)PVID	FDT	PRT	Messages
1	Root(1)-----CPVID	1.15s	**1.5 ms**	2
2	S1(1)-----PVID	2.71s	**0.14 ms**	3
3	S1(3)-----X		**No Impact**	
4	S1(2)-----CPVID		**1.8 ms**	4
5	S3(1)----- X		**No Impact**	
6	S3(2)-----PVID		**1.77 ms**	2
7	S4(1)----- PVID		**0.7 ms**	2

X – Port not part of tree

PVID Port Failures: Case 2, 6 and 7 involve a PVID port failure. In each of the cases the number of messages required to prune the tree is less than 3 – which should be compared to the number of TCN messages exchanged with RSTP. Following the messages exchanged, in case 2, switch S1 informs switches S2 and S3 about the failed VID, Switch S2 updates its PVID and informs its new PVID parent. The PRT is 1ms or less in these cases which should be compared to approximately 3 sec PRT recorded with RSTP.

CPVID Port Failures: In cases 1 and 4, the CPVID port is failed. The PRT with MTP is around 1.5 ms several magnitudes less than a similar port failure with RSTP. This value is less than the lowest recorded with RSTP (under case 7, when a port at the tree edge was failed). In most cases with MTP the tree resolves with less than 5 MTP messages.

3) Summary 5-switch Protocol recovery

The profound difference in PRT noticed between RSTP and MTP is because of the VIDs used by MTP to simplify both tree construction and pruning operations. With MTP, the VIDs maintain the tree information, whereas with RSTP, each switch has to maintain its ports in Root, Designated or Alternate role to define the spanning tree. Thus, in the event of a topology changes, all switches exchange BPDUs, to resolve the port roles and repair the tree.

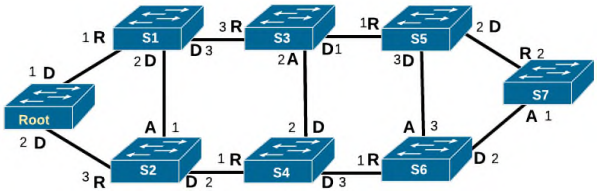

Figure 8. RSTP 8 Node Topology with Port Roles

C. Medium Topology (8 switches)

The 8-switch topology running RSTP, along with the port roles is given in Figure 8. As before we failed single ports as indicated in column 2 of Table 3. We also record the port roles.

1) Convergence in 8-node RSTP Topology

Root Port Failures: Cases 2, 5, 8, and 11 record data on Root port failures. Notice that the FDT is 0 seconds as the switch with the failed Root port initiates TCN immediately on the port failure detection. However, the PRT is approximately 3 seconds except in the case 11, where the failed Root port was at the tree edge, and the switch had an alternate port, which took over immediately on the root port failure. Compared to the PRS and TCNs recorded for the 5-switch topology, in this case there is an increase in the PRS and TCNs. This is because these messages are generated by more switches, and more switches change their port roles/states. This could contribute to high PRT. This also indicates instability in the network while the network is stabilizing after the port failure.

In case 2, on *S1(1)* (root port) failure, S1 immediately initiates recovery. However, as S1 does not have an alternate port, S1 assumes it is the root and negotiates with its neighbors. Hence the PRT was 3.032s. Failure of S5(1) (root port) in the middle of the network resulted in a PRT of 3.525s, even though S5 immediately takes action on its port failure. Failure of S7(2), (root port) edge of the tree, resulted in its alternate port taking over within 12ms.

Designated Port Failures: Cases 1, 3, 4, 6, 7, 9, and 10 record data collected on a Designated port failure. The PRT is again approximately 3 seconds. In case 3, even though there were only 3 PRS, a total of 37 TCNs were recorded and RSTP still took approximately 3 seconds of PRT.

The PRT with RSTP is very much dependent on the point of failure with respect to the root and also if the failed port was designated or root.

2) Convergence in 8-Node MTP Topology

The 8-switch MTP topology is given Figure 9. Instead of providing the broadcast tree, we decided to show the three VIDs stored at each switch. The PVIDs at each switch connected by red lines and show the broadcast tree. Following the VID dotted decimal format the multiple paths from each switch to the root switch will be clear. The green lines and purple lines show how the other 2 VIDs

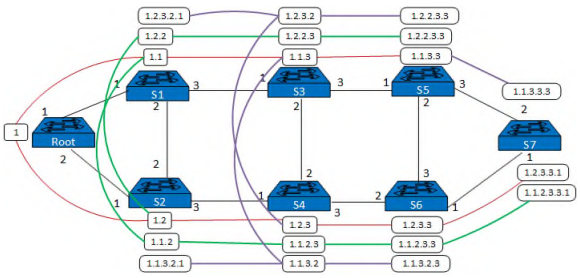

Figure 9. MTP 8 -Node Topology with Meshed

were derived at each switch. They also show how the VIDs provide the path from a non-root switch to the root switch. In Table 4, we record only 9 cases as we did not record port failures that had no impact on the broadcast tree. PRT is still very low – tens of ms. In certain cases we recorded microseconds and these are noted as <1ms in the table.

TABLE III. RSTP CONVERGENCE IN A 8-NODE TOPOLOGY

Case	Failed Port --Role	FDT-initiated	PRT	PRS	TCN
1	Root(1) -----D	5.037s by S1	**3.021s**	22	75
2	S1(1)-----R	0s S1	**3.032s**	19	68
3	S1(2) -----D	4.023s by S2	**3.005s**	3	37
4	S1(3)-----D	5.206s by S3	**3.027s**	13	59
5	S4(1)-----R	0s S4	**2.528s**	15	72
6	S4(2)-----D	5.017s by S4	**3.004s**	3	37
7	S4(3)-----D	5.526s by S6	**3.014s**	6	30
8	S5(1)----R	0s S5	**3.525s**	15	40
9	S5(2)-----D	4.199s by S7	**3.012s**	6	35
10	S5(3)-----D	5.018s by S6	**3.005s**	3	39
11	S7(2)-----R	0s S7	**12 ms**	3	36

PVID Port Failures: Cases 2, 5, 8 and 11 are PVID port failures – i.e. the main path between that switch and root switch failed. This is however repaired by MTP in around17 ms, when the PVID at a switch closer to the root failed (worst case). As the PVID port is further away from the switch the recovery time with MTP drops down – and is less than 1 ms in cases 11 and 8. The fast recovery is attributed to the fact that the only action in these cases is reinstating the next VID in the VID table as the PVID. Note the very low number of message exchanges.

TABLE IV. MTP CONVERGENCE IN A 8-NODE TOPOLOGY

Case	Failed Port– (C)PVID	FDT	PRT	Messages
1	Root(1)-----CPVID	1.740s	**14.6ms**	6
2	S1(1)-----PVID		**17ms**	4
3	S1(2)	**No Impact**		
4	S1(3)-----CPVID	1.024s	**15ms**	5
5	S4(1)-----PVID		**6ms**	4
6	S4(2)	**No Impact**		
7	S4(3)-----CPVID	2.407s	**7ms**	4
8	S5(1)-----PVID	2.290s	**< 1ms**	4
9	S5(2)	**No Impact**		
10	S5(3)-----CPVID	1.046s	**5ms**	3
11	S7(2)-----PVID	2.756s	**< 1ms**	0

CPVID Port Failure: In cases 1, 4, 7 and 10 a CPVID port was failed. The PRT in these cases also reduced as the failure port was further away from the Root. Closer to the root the PRT was around 15ms, further from the root the PRT was around 5 to 6 ms. The low number of message exchanges are primarily the delete messages sent to prune the VIDs from the deleted VID – as described in Section IV.A, Figure 5.

D. Large Topology (17 switches)

Figure 10 shows a bigger, and more connected 17-node topology, with a diameter of 7 hops. A higher diameter topology should result in higher convergence times, but the higher connectivity counteracts this. With higher connectivity, there are several alternate ports at switches S4, S6, S7, S9, S10, S11, S14 and S15. In Figure 10, we also show the spanning tree constructed by RSTP. We will refer to this tree when comparing results from the MTP 17-node topology. The single port failures are noted in column 1 Table 5.

 ISBN: 978-1-61208-786-3

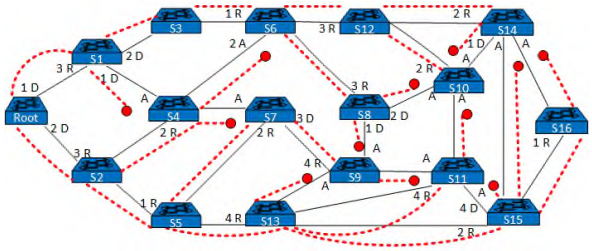

Figure 10. RSTP 17- Node Topology with Port Roles and Broadcast Tree

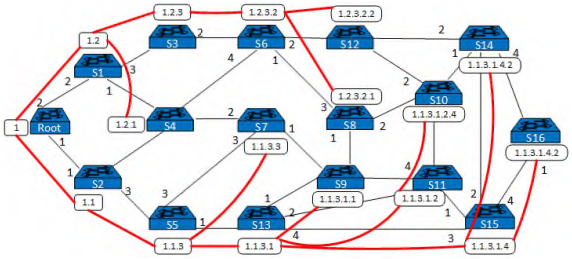

Figure 11. MTP 17- Node Topology with Broadcast Tree

1) Convergence in 17-Node RSTP Topology

Root Port Failures: Cases 4, 5, 9, 11, 12 and 14 introduce Root port failures. Except for case 12 in all other cases the PRT is low and varies from 15 ms to 40 ms. In case 12, the PRT is around 3 seconds –the reason is that at switch S15, there are no alternat ports to take over immediately. Though this is at the farther end from the Root, the switch declares itself as root switch which then takes time to resolve. In all other cases, the switch with a failed root port had an alternate port. The number of TCN messages exchanged are very high.

Designated Port Failures: All other failure cases were designated port failures and the PRT averaged around 3 seconds.

2) Convergence in 17-node MTP Topology

Figure 11 shows the 17-node MTP topology, with Broadcast Tree supported by PVIDs and CPVIDs.

PVID Port Failures: They are cases 4, 7, 9, 10, 12 and 14. In all these cases the PRT varies from 12 ms to 6 ms. The convergence is achieved with very few message exchanges much lower than RSTP.

CPVID Port Failures: These are cases 1, 3, and 13. The maximum PRT recorded was for case 1, where the failure of a port at the root switch results in the tree originating from this VID which had to be pruned through 9 messages. For cases 3 and 13 it was as low as 5 and 3 ms.

Compared to RSTP – where every change resulted in transients, this indicates a major reduction in processing overhead. PRT is in tens of ms compared to RSTP.

TABLE V. RSTP CONVERGENCE IN A 17-NODE TOPOLOGY

Case	Failed Port - Role	FDT	PRT	PRS	TCN
1	Root (1)-----D	4.501s by S3	3.462s	26	100
2	S1 (1)-----D	5.024s by S4	3.010s	3	80
3	S1 (2)-----D	4.086s by S3	3.028s	10	80
4	S1 (3)-----R	0s by S1	***40ms***	20	110
5	S7 (2)-----R	0s by S7	***24ms***	3	90
6	S7 (3)-----D	4.680s by S9	3.019s	6	85
7	S8 (1)-----D	3.231s by S8	3.000s	3	84
8	S8 (2)-----D	3.998s by S10	3.001s	3	84
9	S8 (3)-----R	0s by S8	***32ms***	10	106
10	S14 (1)-----D	4.466s by S10	3.007s	3	93
11	S14 (2)-----R	0s by S14	***25 ms***	3	100
12	S15 (2)-----R	0s by S15	3.054s	30	153
13	S15 (4)----D	5.475s by S11	3.011s	3	80
14	S16 (1)-----R	0s by S16	***15ms***	5	88

VI. SUMMARY

MTP and RSTP were evaluated for several failure test cases where we tracked the number of message exchanges and port role/state changes (for RSTP). The high number of message exchanges with RSTP indicate the complexity of the convergence process which in turn is reflected in the convergence latency. To capture the significant difference in PRT and messages exchanged we plotted MTP vs RSTP data on a log scale. They are provided in Figures 12-15. Figures 12 and 13 capture the PRT experienced for single port failures in the 5 and 8 node topologies. We used a log scale for the latency (seconds) to highlight the difference in PRT with MTP and RSTP. The green line in the figures show the cases where the port failure had no impact on broadcast tree. In certain root port failures RSTP recovered fast if there existed an alternate port. MTP consistently had a PRT lower than these cases of RSTP. The no impact points further prove the improved network stability with MTP for single port failures.

In Figures 14 and 15, we plotted the number of messages exchanged on a topology change for MTP vs RSTP. While RSTP exchanged 10 to 100 messages, MTP exchanged less than 10 messages. This shows how lightweight MTP is as compared to RSTP. Even when the PRT was low, RSTP still exchanged several messages, the overhead and operational complexity with RSTP is very much higher than MTP.

TABLE VI. MTP CONVERGENCE IN A 17-NODE TOPOLOGY

Case	Failed Port – (C)PVID	FDT	**PRT**	Messages
1	Root (1)-----CPVID	2.763s by S2	**36ms**	9
2	S 1 (1)	**No Impact**		
3	S 1 (3)-----CPVID	2.726s by S3	**5ms**	2
4	S1 (2)-----PVID	-	**11ms**	4
5	S7 (1)	**No Impact**		
6	S7 (2)	**No Impact**		
7	S7 (3)-----PVID		**5.4ms**	2
8	S8 (2)	**No Impact**		
9	S8 (3)-----PVID	2.450 by S6	**7ms**	3
10	S14 (2)-----PVID	1.911 by S12	**6ms**	2
11	S14 (4)	**No Impact**		
12	S15 (3)-----PVID		**12ms**	5
13	S15 (4)-----CPVID	2.453s by S16	**3ms**	2
14	S16 (1)-----PVID	2.946s by S15	**6ms**	1

 ISBN: 978-1-61208-786-3

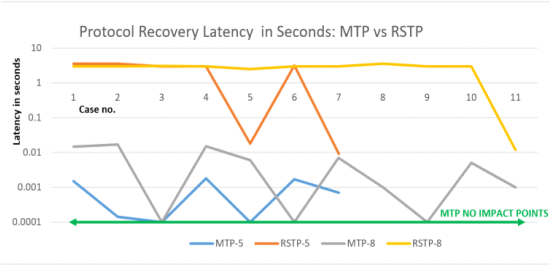

Figure 12. PRT - MTP vs RSTP (5 and 8 switches)

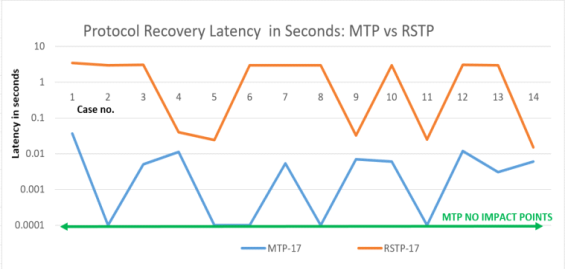

Figure 13. Message Exchanges MTP vs RSTP (5 and 8 switches)

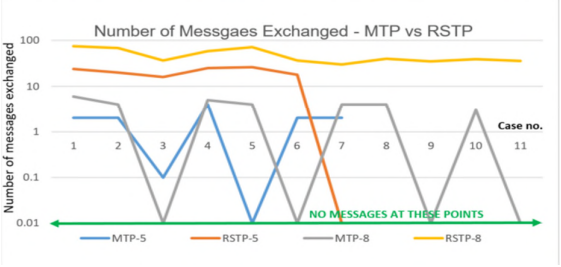

Figure 14. PRT - MTP vs RSTP (5 and 8 switch topology)

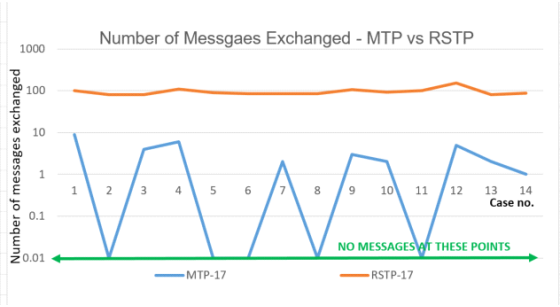

Figure 15. Message Exchange MTP vs RSTP (17 switch topology)

VII. Conclusion

MTP uses a novel meshed tree algorithm to construct and stores multiple tree information constructed from a single root, which is carried in simple structured VIDs. The VID information is very powerful both for tree construction, maintenance and pruning on failures. The presented data and tree construction and maintenance information indicates the robustness of the protocol and high resiliency to failures offered by MTP. Using MTP in customer, SP and BP networks will outperform RSTP and IS-IS based solutions in all aspects – performance, resource usage and reduced operational complexity. In this article, we limit the comparison to RSTP. Future work will extend the comparison studies to IS-IS based solutions.

References

[1] P. Willis and N. Shenoy, “A Meshed Tree Protocol for Loop Avoidance in Switched Networks”, Workshop, IEEE International Conference on Computing, Networking and Communications, 18-21 Feb. 2019, Honoulu, Hawai, USA, ICNC 2019

[2] IEEE 802.1w - Rapid Reconfiguration of Spanning Tree, supplement to ISO/IEC 15802-3:1998 (IEEE Std 802.1D-1998)

[3] R. Perlman, “Rbridges: Transparent Routing”, IEEE Proceedings of Infocomm 2004.

[4] R. Perlman, D. Eastlake, G. D. Dutt and A. G. Gai, “Rbridges: Base Protocol Specification”, RFC 6325, July 2011.

[5] J. Touch, R. Perlman, “Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement”, RFC 5556.

[6] P. Ashwood-Smith (24 February 2011). "Shortest Path Bridging IEEE 802.1aq Overview". Huawei. Retrieved 11 May 2012.

[7] "IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging", http://tools.ietf.org/html/rfc6329, IETF. April 2012. Retrieved 2 April 2012. Retrieved 4th Nov 2014.

[8] IEEE LAN/MAN Standards Committee of the IEEE Computer Society, ed. (1998). ANSI/IEEE Std 802.1D, 1998 Edition, Part 3: Media Access Control (MAC) Bridges

[9] L. Goodman, A. Lauschke, and E. W. Weisstein, “Dijkstra's Algorithm,” MathWorld—AWolfram Web Resource. [Online]. Available: https://mathworld.wolfram.com/DijkstrasAlgorithm.html. [Accessed: 10-Jun-2020].

[10] www.geni.net [Accessed: 10-Jun-2020].

[11] https://www.wireshark.org/docs/man-pages/tshark.html, Retrieved 31 Oct. 2018

[12] https://chrony.tuxfamily.org/ [Accessed:31-July-2020]

[13] Bidirectional Forwarding Detection https://tools.ietf.org/html/rfc5880#section-3.1 [Accessed:1-May-2020]

[14] BFD Dampening https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bfd/configuration/xe-3s/irb-xe-3s-book/irb-bfd-damp.html [Accessed:31-July-2020]

 ISBN: 978-1-61208-786-3