Optimized Unicast Frame Forwarding in Meshed Tree Protocol

Project Report Submitted By

Rahul Venugopala Pillai

Advised By

Dr. Leon Reznik
# Table of Contents

1. Abstract 4  
2. Introduction 5  
3. Background 6  
   3.1 Current Protocols 8  
      3.1.1 Spanning Tree Protocol 8  
      3.1.2 Rapid Spanning Tree Protocol 9  
   3.2 Spanning Tree Protocol Limitations 9  
4. Meshed Tree Protocol 10  
   4.1 Basic Operations 10  
5. Project Goal 13  
   5.1 Implementation of Unicast Frame Forwarding 13  
   5.2 Performance comparison with RSTP 14  
6. Assumptions 14  
7. System Specifications 15  
8. Implementation 15  
   8.1 Unicast Frames in MTP 15  
      8.1.1 Host Address Table (HAT) 17  
      8.1.2 Host Address Advertisement (HAA) Entry 18  
   8.2 Host Address Advertisement in MTP 19  
   8.3 Link Failure Detection in MTP 22  
   8.4 Link Failure Recovery in MTP 22  
   8.5 RSTP Configuration 24  
   8.6 Unicast Frame Generator for Testing 24  
9. Metrics Measured 25  
10. Test Topologies 26  
11. Results and Analysis 28  
12. Conclusion 31  
13. Future Work 32  
References 33
# Table of Figures

1. Unicast Frame Forwarding 7
2. Broadcast Frame Forwarding 7
3. Broadcast Storm 8
4. VID formation in MTP 12
5. Flowchart showing Handling of unicast frames 16
6. Flowchart of HAA handling 20
7. Link Failure Detection Flow Chart 22
8. Link Failure Recovery Flow Chart 23
9. Three node topology 26
10. Five node topology 27
11. Seven node topology 27
12. End-to-End Delay - Comparison 29
13. Packets Dropped during Link Failure – Comparison 30
1. Abstract

The communications in a network largely rely on the switching operations at layer 2 of the network protocol stack. Backbone networks, service provider, and data centers depend on these switching operations at layer 2. The network communication at this layer should be of high speed and reliable since it affects a large number of stakeholders. Delays and unstable links are something which cannot be avoided in any network. Operations in layer 2 should not get affected due to these issues and it should always ensure reliable communication. This can be achieved by adding redundancy to the system either by providing multiple paths to the end devices. Redundancy introduces loops in the network and during the event of broadcast, frames will be repeatedly transmitted by the switches resulting in a broadcast storm. Currently deployed protocols in layer 2 like Rapid Spanning Tree protocol handles this scenario by providing logical tree paths. However, this results in non-optimal paths for forwarding both broadcast and unicast frames. Additionally, during failures, the downtime for the current protocols to resolve the logical tree paths is also very high. The project aims at solving these issues by making use of Meshed Tree Protocol \cite{1}\cite{5}.
2. Introduction

Layer 2 switching operations influence the overall communication in a large network. Unstable links and events that result in outages are common in any network and cannot be avoided. However, the network protocols in layer 2 should be able to handle these issues and ensure stable and reliable high-speed communication. This is achieved by adding redundancy in the network. Redundancy can be added to a network either by adding duplicate links or by duplicate nodes and switches.

Adding redundancy to a network can result in network loops. Network loops cause issues while broadcasting messages. Messages keep on getting broadcasted in such loops and finally, the switches involved will crash. This ends up in interrupting any communications that happened through the same network. To avoid such issues, current protocols like Spanning Tree Protocol (STP) and Rapid Spanning Tree Protocol (RSTP) creates logical spanning trees from the existing network. This is achieved by blocking some of the ports and thereby removing loops from the network. As a result, broadcasting can be performed in the network without any issues. Additionally, unicast frame forwarding is also achieved using the same spanning tree path even if shorter paths are available. This is because of the reason that some of the ports are blocked for forming the spanning trees. Also, these protocols make use of a single spanning tree in the network, even if multiple spanning trees can be formed. Hence, the drawbacks of the current protocols running in layer 2 are high network delays and high convergence time during link failures.
Meshed Tree Protocol \cite{1} is an IEEE-recognized protocol and it focuses on solving current performance issues in layer 2 switch networks. This protocol creates multiple spanning trees in a network using the same root. As a result, during link failures, the network can easily switch from one tree to another and hence the downtime is considerably less compared to the existing protocols. One of the current problems with MTP is that there is no mechanism to have unicast frame forwarding. The project aims at solving this problem by making use of the fact that no ports are physically blocked in a switch when it runs MTP. So, the switch can make use of these ports to forward the unicast frames and we can select the best port from the available set of ports, based on least cost, to forward the unicast the frame. Also, it is possible to maintain a set of best ports to forward the unicast frame and in the case of any link failure, the alternate port can be used to forward the frames. This results in fewer network delays during link failures and high transfer rate since we are going for the optimal paths for frame forwarding.

3. Background

In this section, we will be discussing about some of the basics of networking that will be useful for understanding the project. Unicast frame forwarding, broadcast frame forwarding and broadcast storm are explained below with the help of figures.

**Unicast Frame Forwarding**

In unicast forwarding, messages will be sent from one device to another device in the network. The messages are destined to reach exactly one destination device.
Broadcast Frame Forwarding

In broadcast forwarding, messages will be sent from one device to all other devices in the network. Every message is destined to reach every other device in the network.

Broadcast Storm

In a network, when redundancy is added, there are chances of forming loops within it. So, in such a network, if broadcast message arrives it keeps on getting transferred from one switch to all other switches in the network. Eventually, switches need to handle too many messages and it will finally crash. This scenario is called Broadcast storm and it needs to be avoided in any network to ensure stable communication.
3.1 Current Protocols

The protocols currently in layer 2 networks basically forms a spanning tree from the network from a single root. There are no loops in a spanning tree. Hence the broadcast messages are forwarded using the spanning tree and it reaches all the other devices in the network. The protocols which are deployed in layer 2 are discussed in detail below:

3.1.1 Spanning Tree Protocol

The spanning tree protocol [10] as discussed previously works based on the formation of a spanning tree using the devices in the network. A root election process happens initially and the root node will be identified. After this process, in all other devices, the ports are classified into root ports, designated ports and blocked ports. The ports which are no more used are identified as blocked ports whereas ports which are connected to the root switch is identified as the root port. Designated ports are those which connects other switches to it.
When a message needs to be sent to all the devices, it makes use of the spanning tree formed and the message gets broadcasted across all devices in the network. For a unicast message also, it makes use of the spanning tree to forward the frame. In this case of unicast message forwarding, even if direct links are available between two switches, it may follow a longer path to communicate each other. This is because some ports are already blocked to form the spanning tree.

### 3.1.2 Rapid Spanning Tree Protocol

Rapid spanning tree protocol \(^2\)\[^{10}\] is an advanced version of spanning tree protocol. As the name suggests, it is faster than the spanning tree protocol and it is in terms of convergence time during network initialization and link failure. The root election process and ports identification process in spanning tree protocol are further optimized in this protocol and hence the delay to setup the network is much lesser. Once the spanning tree is established all kind of messages are forwarded like STP.

### 3.2 Spanning Tree Protocol - Limitations

**High convergence time during link failures**

During link failures \(^2\)\[^{10}\], the spanning tree needs to get formed again and this is a time-consuming process. The network initialization is a time-consuming process and with MTP, this is avoided. Hence the downtime is more in the current protocols during failures compared to MTP.

**Non-optimal path for unicast frame forwarding**

Current protocols like STP \(^2\)\[^{10}\] and RSTP \(^2\)\[^{10}\] make use of spanning tree formed to send a unicast frame from one device to another device in the network.
Hence in most of the cases, the unicast frame makes use of a longer path than available shorter paths. This is because some of the ports are already blocked to form the spanning tree and they might be the part of the shortest path.

4.0 Meshed Tree Protocol

Meshed Tree Protocol \(^{[1][5]}\) is an IEEE-recognized protocol research project. MTP also works based on the concept of spanning trees. When compared to STP and RSTP, MTP relies on multiple spanning trees to forward messages. MTP initially identifies the different spanning trees that can be formed in a network. This is based on the name of each switch called VID \(^{[1][5]}\) (Virtual ID).

A switch can have multiple VIDs and one of them is identified as the primary VID. All other VIDs will be considered as the backup VIDs. Message forwarding is performed based on the primary VID and using the corresponding spanning tree. In the case of link failure, immediately one of the VIDs in the backup table will be considered as the primary one and message forwarding is performed. Hence the downtime is very less here during link failures.

4.1 Basic Operations – Meshed Tree Protocol

This section describes about the basic operations in MTP. The basic operations include the root election process and virtual ID (VID) generation. In the root election process, one of the switch is identified as the root and others as non-root switches. Root switch will be the root of the logical spanning tree generated. The processes are explained in detail below:

**Root Election Process**

As discussed previously, root election process \(^{[1][5]}\) is the first process in
MTP also. It is done like STP and RSTP. The switch with lowest bridge ID which is the combination of MAC ID and highest priority will be considered as the root. All switches exchange their IDs each other and will identify themselves as root or not.

**Virtual IDs (VID) and assignment**

VIDs are generated for each switch in the network in MTP\(^1\)\(^5\). VIDs gives the information about the path from a switch to root. From a single root, multiple spanning trees can be formed and hence a switch in the network may have multiple VIDs. One of the VIDs will be identified as the primary VID and rest all will be considered as the secondary VIDs. The information regarding the VIDs will be stored in a table and it is called VID table.

Switches in MTP acquire VIDs as a part of the Meshed Tree formation process. The VIDs are acquired by each switch using the below three types of control messages:

1) **Periodic Hello Messages**: MTP switches which has received VID, keeps sending periodic ‘HELLO’ messages to its neighbors\(^1\)\(^5\). Whenever a switch receives a ‘HELLO’ message, it compares the VID present in the message with the entries present in its table. If the entry is not present, a ‘JOIN’ message is sent from the switch. Otherwise it updates the entry with the new timestamp, to keep the track of active links. At any point of time, if a switch misses three continuous ‘HELLO’ messages from its neighbor, the link is assumed to be failed and subsequently the associated VID entries are removed from its table.

2) **Join Messages**: When a neighbor receives a JOIN message\(^1\)\(^5\), an advertisement (ADVT) message is given back, which contains details about all
the VIDs present in the Root VID table. In this way, the new device in the network will assign its VID based on the port from which it received the ‘Join’ message and VID received in the message. Each switch will send its table, which contains details about all the switches connected to it, to all its neighbors.

3) **Advertisement Message (ADVT):**
   The advertisement message contains details about all the VIDs associated with the switch \(^{[1]} \text{[5]}\). This message is used by the neighbor switch to assign VIDs to itself and send it across other switches based on Join requests.

![Diagram of VID formation in MTP](image)

**Figure 4: VID formation in MTP**

**Broadcast Frame forwarding**

In MTP, switches can have multiple VIDs. One of the VID is identified as the primary VID and it is called PVID. In case of broadcast frame forwarding, MTP switch makes use of the PVID of the switch and the message is send to the neighbors based on this VID. The message will be broadcasted across the network based on the PVID of each switch and no ports are disabled in this process.
5. Project Goal

This section gives us the idea about the project goal. The goal of the project is subdivided into two sections based on its individual significance. The first section deals with the implementation of unicast frame forwarding in MTP and the last section is all about the performance comparison of MTP with RSTP. They are discussed in detail below:

5.1 Implementation of Unicast Frame Forwarding

The main goal of the project is to design and implement unicast frame forwarding in MTP. Initially, switches do not have any information about the hosts locally connected to the switch. Once a host sends a message across the network, the switches will learn about the host and it will store the information locally. This information is also advertised across the network and the information is stored in a table called Host Address Table (HAT).

Initially, when a unicast frame is sent across the network the switches may not be having the information about the destination host. In this case, the message will be broadcasted across the network. Once the switches learn about the hosts, it makes use of the shortest path available in the network to forward the unicast frame. In this way, the most optimal path can be selected for unicast frame forwarding.

Host address advertisements (HAA) is performed in MTP so that all switches can learn about all others hosts connected to the network. Multiple port entry information about each host is stored in the host address table. This is to switch to next shortest backup path in case of any issues in the shortest path available.
5.2 Performance comparison with RSTP

The final goal of the project is all about performance comparison. The performance of MTP needs to be compared with current protocol like RSTP. RSTP protocol needs to be configured and run in the switches. Once RSTP is configured, the performance must be compared with MTP. The time taken to send unicast frame, to converge during link failures and similar parameters can be used to compare the performance of both MTP and RSTP.

6. Assumptions

There are some assumptions that have been considered to implement the project. They are discussed in detail below:

1. The root election processes in MTP is like RSTP. Hence, this is not the part of the implementation. Currently, one of the switches is manually set as the root node.

2. It is also assumed that the broadcasting functionality of MTP is working fine and there are no issues such as storm-related to broadcast.

3. The test bed used for implementing and testing the project is GENI (Global Environment for Network Innovation) which is provided by National Science Foundation. It is assumed that the testbed provides a simulated environment which is like actual live environment where the project will be deployed.

4. The application created will be running on user space and not on kernel space. Hence, some of the latency factors are not considered during testing and implementation.
7. System Specifications

Using the standard network libraries, MTP is implemented using C language and it is deployed and tested on Linux machines provided by GENI [8]. The technical details of the project are mentioned below:

- Language used: C, Python
- Operating System: Ubuntu 14.06
- Source Code Management: GitHub
- Scripts: Python, Shell Scripts
- Test Environment: GENI

8. MTP Implementation

This section discusses the implementation of the project. The project implementation is divided into different sections. The first section discusses the handling of unicast frames when it reaches a switch, the second section is about the handling of Host Address Advertisement and the next section discusses link failures and its handling in MTP. One of the section also discusses configuring RSTP in GENI [8] nodes. The final section gives the details about the unicast frame generator that has been created to test the implementation. Each section contains a detailed flowchart showing the flow of execution and in-depth details of the design.

8.1 Unicast Frames in MTP

This section discusses the way unicast frames are handled in MTP [1][5]. This section also discusses the creation of Host Address Advertisement.
Whenever a host wants to send a message to another host, unicast messages are created and it is sent to the immediate switch connected to the sending host. When the switch receives the unicast message, it checks whether it knows about the route to reach the destination device using the destination mac address present in the unicast message. The switch checks its Host Address Table (HAT) to match an entry with the destination device. Every switch contains its own HAT, which has the details about all the host devices present in the network and the port through which messages can be sent to the destination host with the least cost. Once the switch identifies that the destination host is present in its table, it will forward the unicast message to the corresponding port.

![Flowchart showing Handling of unicast frames](image)

*Figure 5: Flowchart showing Handling of unicast frames*

If the switch is not able to find a match for the destination host, the message will be broadcasted using the VID\(^{[1]}\) entry present in each switch. In this way, the destination host receives the message finally.

The next step while receiving the unicast messages by a switch from a host is to identify whether there is an entry for the source host in the HAT. If there is no entry for the source host, a new entry will be created and records all the details
about the local host. The details such as the port through which the host is connected and the cost of reaching the host are recorded in the HAT. Once a new entry is created, a Host Address Advertisement is created and send across all other switches in the network. In this way, all the switches learn about the sending host machine details.

8.1.1 Host Address Table (HAT)

This section discusses more details about Host Address Table and the significance of each field in the table. The structure of HAT is as shown below:

<table>
<thead>
<tr>
<th>Switch Id</th>
<th>Local</th>
<th>Destination MAC</th>
<th>Sequence number</th>
<th>Cost</th>
<th>Port name</th>
</tr>
</thead>
</table>

The details regarding each field in the HAT is mentioned in detail below:

**Switch ID**: Each switch has its own physical address. This field denotes the address of the switch which is the lowest MAC address of all its available interfaces. The switch ID will be unique for each switch and it is a 6-byte address.

**Local**: This field indicated whether the host device is local to the switch or it is connected to some other switch. If this field is set to 0, it indicates that the host machine is locally connected otherwise it is set to 1.

**Destination MAC**: Like switches, each host device has its own physical address. This field is used to identify the physical address of the host device.

**Sequence Number**: The sequence number is used in the case of link failure. Whenever a new advertisement reaches a switch, it checks whether the sequence
number is a larger one compared to the existing one on the table. If yes, it accepts, the message otherwise it discards it.

**Cost:** Like other protocols, MTP also identifies the cost to send the message to each host device and this field is used to store the same. The cost is identified based on the interface bandwidth.

**Port:** This field is used to identify the port of the switch through which the host device is connected.

In each switch, there exists a primary HAT and a backup HAT. The primary HAT records the information about any host that will be used directly by the switch in case of transmitting unicast messages. The entries in the backup HAT table will be used in case of link failure to switch back to the backup path. Using a backup HAT table significantly reduces the convergence time during link failure.

### 8.1.2 Host Address Advertisement (HAA) Entry

In MTP [1][5], each switch advertises about any host connected to it using the Host Advertisement Policy. A Host Address Advertisement entry (HAA) is created and send across all other switches as a message. Other switches, checks locally if it contains the details about the host mentioned in the HAA and adds or discards the HAA entry based on some steps which are discussed later. The structure of an HAA message is as shown below:

<table>
<thead>
<tr>
<th>Switch Id</th>
<th>Host MAC</th>
<th>Sequence Number</th>
<th>Cost</th>
<th>Control Frame Type</th>
<th>Ethernet Frame Type</th>
<th>Destination MAC</th>
<th>Source MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 bits</td>
<td>48 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>16 bits</td>
<td>48 bits</td>
<td>48 bits</td>
</tr>
</tbody>
</table>

The details regarding each field in the HAA is mentioned below:
Switch ID: Each switch has its own physical address. This field denotes the physical address of the switch to which the host device is connected.

Host MAC: This field is used to identify the physical address of the host device mentioned in the HAA.

Sequence Number: This field is useful during a link failure to identify the entry as an expired one or the latest one. A higher number for this field indicates that the entry is a valid one otherwise not.

Cost: This field is used to identify the cost to send messages to the destination host mentioned in the HAA using the port from which a switch received the HAA.

Ethernet Frame Type: This field is used to identify the type of the Ethernet frame whether it is a control message or a data message. The Ethernet message type for HAA is of a fixed value and this is same for all the HAA messages.

Control Frame Type: This field is used to identify the type of the control message. The control message type for HAA is of a fixed value and this is same for all the HAA messages.

Destination MAC: This field represents the physical address of the switch to which the HAA is sent to.

Source MAC: This field represents the physical address of the switch from which the HAA is generated.

8.2 Host Address Advertisement in MTP

In this section, we will discuss the handling of Host Address Advertisement in MTP\(^1\)[5]. Whenever a switch receives an HAA message, it checks for various
parameters and identifies if the host entry present in the HAA message is useful for the switch to transmit messages to it. If yes, it updates its own HAT and sends the updated information as an HAA to other switches. To identify whether the entry is a valid one for the switch as discussed early certain parameters are checked and this is discussed in detail here.

Figure 6: Flowchart of HAA handling
When a switch receives an HAA message, it compares the entry present in the HAA with the entries present in its HAT. If the entry contains a totally new host device details, it is added to HAT and then sends as HAA to other switches. If the entry for the host device already exists, then the switch compares the parameters like cost, switch id, arrival port and sequence number in the HAA entry with the HAT table entry. If the HAA entry is of lower cost, then the HAT table is updated with this entry otherwise it is compared with the entry in the backup HAT. If the HAA entry is better than backup HAT entry, the backup HAT is updated with this information and otherwise, the HAA is discarded. A new HAA is created for the received HAA entry only if it updated the main HAT.

As mentioned in the flowchart, initially the HOST MAC entry in the HAA is compared with all the entries in the HAT. If the HOST MAC is a new one, this information is updated in the HAT, otherwise, the entry is compared with other parameters. If there exists an entry with the same host mac address, then the switch to which the host is connected is checked. If it is not same as before, the HAT is updated with this information and an HAA is send to all other switches. If it is same, then the port from which the HAA is received is checked. If it is same, the path cost is compared and identified whether the entry needs to be stored in HAT or the backup HAT.

If the port of arrival is also same, it checks for the sequence number and checks whether the entry is having a lesser sequence number and if yes, it discards the message. The next comparison is done in order to check whether a link failure happened and if the HAA is having a greater sequence number which indicates the HAA is having a path that got updated and it is updated in the HAT and the entries are reordered accordingly based on the cost field. If the cost field is set to a value 99 or greater, it indicates that the path does not exist and the corresponding entry is removed from the HAT.
8.3 Link Failure Detection in MTP

In MTP, link failure is detected using the Hello messages. Switches exchange Hello messages at regular interval of 2 seconds. Every two seconds, switches send hello messages to all its neighbors. Whenever a link failure happens, switches will not be receiving Hello messages in its interfaces. The switches keep a counter and checks if it is not receiving Hello messages from a neighbor switch for a continuous time frame of 6 seconds (Time for 3 Hello Messages). If it is not receiving Hello messages in this time frame, it considers that the link is down and the switch updates its HAT and send HAA messages across the network.

8.4 Link Failure Recovery in MTP

Whenever a link failure happens, there is a high chance that there are some entries in the HAT using the same link to send messages to different hosts. This needs to get updated immediately whenever a link failure happens and this process is called as Link Failure Recovery. The below flowchart indicates the sequence of steps that is taken care during a link failure to ensure recovery.

In MTP, the link failure is detected using the Hello Messages as discussed previously. Each switch keeps a timer regarding each entry and it checks if any switch is not sending Hello messages for a maximum time of 6 seconds. In that case, each entry in the HAT table is checked whether any of the entry is based on
the failed link. If yes, it creates an HAA with the same entry but with a path cost of 99 and sent it to other switches.

Figure 8: Link Failure Recovery - Flow Chart
Setting path cost to a default value of 99 (which can be set to the desired value later) helps other switches to identify that the entry is no more there and delete the entry from their HAT. In this way, the entire network will get notified about the link failure. Each switch, when it comes to knowing that a link is down, the backup HAT is immediately checked and the main HAT is updated if an entry is present in the backup HAT by promoting it. In this way, recovery also happens, which makes the network more stable in a lesser time.

8.5 RSTP configuration.

One of the major challenges faced in the project was to configure RSTP in GENI nodes. RSTP was successfully configured with the use of Open vSwitch [9][10][11] implementation. Open vSwitch is an open source project which contains the implementation for configuring virtual switches in a distributed network. Open vSwitch provides us with a set of commands which can be easily used to configure RSTP in any network on top of Linux boxes.

8.6 Unicast Frame Generator for Testing

Additionally, a unicast frame generator [3][4][6][7] was created during the project to test MTP implementation as well as to compare MTP and RSTP. The unicast frame generator developed can transmit any desired number of packets to any host in the network. The payload in the message can be also changed easily and it is very useful to obtain different performance metrics. Both MTP and RSTP was compared using the same unicast frame generator and it can be used with any other protocol with minimal changes.
9. Metrics Measured

The performance of MTP and RSTP in a network was evaluated based on many metrics. The metrics that were considered as a part of this project were:

End-to-End Delay

The end-to-end delay was used to evaluate the performance initially. The average time for a packet to reach from one host to another host was measured for this. The higher end-to-end delay indicates that for every packet it requires more time to transfer from a host to another host and lesser end-to-end delay indicates that it requires less time for the protocol to transfer messages from one host to another.

Number of packets lost during link failure

This metric was used to evaluate the performance of the protocol when a link fails. This metric gave us the count of the packets which were dropped whenever a link failed. A lesser number of packets dropped during any link failure indicated that the protocol is more resilient to failures and is a more stable one.

Convergence time during link failure

The time taken for the protocol to recover from a link failure was calculated for this metric. The larger the convergence time indicates that the protocol needs more time to take recovery mechanism and it is not good for a layer 2 network.

No of hops between hosts

This metric gives us the count of hops between hosts when a host sends a message to another host in the network. For a shorter path, there will be less
number of hops between the hosts and for a longer path, there will be more.

10. Test Topologies

The implementation of MTP was tested using GENI [8] testbeds. GENI provides the options to create network topologies with any number of nodes in it. We have tested the implementation using 3, 5 and 7 node topologies in GENI. The topologies are discussed in detail below:

Three Node Topologies

Three node topology was one of the simplest topologies for testing the unicast implementation. The three-node topology was configured for MTP in such a way that it contained one root node and two non-root nodes which acted like switches. The topology also contained three nodes connected to each of these three switches. The topology is as shown below:

![Three node topology diagram]

Figure 9: Three node topology

Five Node Topologies

Five-node topology was a more complex topology than a three-node topology. The five-node topology was configured for MTP in such a way that it
contained one root node and four non-root nodes which acted like switches. The
topology also contained three nodes connected to three switches. The topology is
as shown below:

**Figure 10: Five node topology**

**Seven Node Topologies**

Seven node topology was the most complex topology used for testing the
RSTP implementation. The seven-node topology was configured for MTP in such
a way that it contained one root node and six non-root nodes which acted like
switches. The topology also contained three nodes connected to three switches.
The topology is as shown below:

**Figure 11: Seven node topology**
11. Results and Analysis

Unicast frame forwarding is implemented in MTP. The implementation has been tested using various topologies in GENI. RSTP is also configured in GENI. In order to compare the performance of RSTP and MTP, various metrics were considered as discussed previously. A unicast frame generator was created and the same was used for obtaining different performance metrics. The results are as follows:

**End-to-End delay:**

In order to calculate the end-to-end delay, a stream of packets sent from host device to another host device in the network. This was performed in three-node, five-node, and seven-node topologies. One of the topologies was using MTP and the other one was using RSTP. The end to end delay was calculated for the same hosts in both the topologies

**END-TO-END DELAY (micro seconds, µs)**

<table>
<thead>
<tr>
<th>PROTOCOL USED</th>
<th>3 - NODE</th>
<th>5 - NODE</th>
<th>7 - NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTP</td>
<td>82.29 µs</td>
<td>104.93 µs</td>
<td>106 µs</td>
</tr>
<tr>
<td>RSTP</td>
<td>103.159 µs</td>
<td>134.85 µs</td>
<td>145.98 µs</td>
</tr>
</tbody>
</table>

It has been observed that for all the three topologies tested, MTP was having less end-to-end delay compared to RSTP. The average time taken for MTP to transfer the packets between the hosts was much lesser compared to RSTP.
Number of packets lost during link failure:

In order to calculate the number of packets lost and the convergence time, one of the links was failed during the transmission of a very large number of packets. Then the number of packets reached the destination was counted. This was performed in three-node, five-node, and seven-node topologies. One of the topologies was using MTP and the other one was using RSTP. The count was taken for both the topologies and the results are impressive.
The convergence time during link failure for MTP was much lesser compared to RSTP. The recovery mechanism in MTP was much faster and after a link failure, it recovered very fast. Hence only a few packets were dropped. But in the case of RSTP, the convergence time was much higher and a high number of packets were lost which is not desirable for a resilient network.

![Figure 13](image)

**Figure 13**: Packets Dropped during Link Failure - Comparison of RSTP and MTP
Convergence time during link failure:

The convergence time during link failure was also compared for both the protocols. The time for which the packets are not transferred across the network and the recovery happens is termed as convergence time. It was clearly observed that the convergence time for MTP was much lower than RSTP. Due to this reason, less number of packets were dropped in MTP during link failure compared to RSTP. One of the main reason for this is that MTP records multiple backup paths and in the case of any link failure, the primary path will be switched to one of the backup paths and this does not require much time. But in the case of RSTP, a new logical spanning tree should get formed and this takes more time and adds to the convergence time.

12. Conclusion

Unicast forwarding has been implemented in MTP. Unicast messages are transferred from one host to another host using the available shortest paths. Backup paths are also stored in the switches so that in the case of link failure, the path can be switched to the backup path. The performance of unicast frame forwarding was compared for MTP and RSTP. RSTP was configured in the GENI nodes using Open vSwitch\textsuperscript{[10][11]} implementation. The performance was compared based on different parameters such as end-to-end delay, the number of packets lost during link failure and convergence time during the same scenario. The performance of MTP was much better than RSTP with respect to all these parameters. MTP could transfer packets with less end-to-end delay and there was much less number of packets dropped during link failure compared to RSTP. A unicast frame generator was created for testing the performance for both MTP and RSTP.
13. Future Work

With respect to unicast frame forwarding, MTP outperformed RSTP in all the test cases during performance evaluation. The performance was compared on a three-node, five-node and seven-node topologies. More complex topologies can be tested in future to evaluate the performance of a much more complex network. Also, additional performance metrics can be collected to compare the performance of MTP and RSTP in a much more deeper level. Additional performance metrics include the number of hops between the hosts, the number of control messages transferred through the network and the convergence time during link failure for both MTP and RSTP.

References


[8] GENI (https://www.geni.net/about-geni/what-is-geni/)


[10] RSTP Wikipedia—