Meshed Tree Bridging – A Clean Slate Solution for Switched Networks

Nirmala Shenoy, PhD
nxsvks@rit.edu
Professor ISchool, Golisano College of Computing and Information Sciences
Director, Lab for Networking and Security
Rochester Institute of Technology
New York, USA

Overview

- Growing demand for Switched Networks
- Challenges faced -> the need for a new approach
- Meshed Tree Algorithms and Protocols – A Clean Slate Solution
- Meshed Tree Protocol – Prototype Evaluated
    - Fast convergence – with backup broadcast paths
    - Optimal root redundancy
    - Independent Unicast paths improves link utilization and unicast performance
    - Fast Failure detection and dissemination
    - Multi Rooted Meshed Trees for Data Center Networks
- Comparison with Rapid Spanning Tree Protocol

Switched Networks - Growth

- Switched Networks within organizations – customer networks
- Service Provider Networks (SPN) – to connect customer VLANs across locations
- Backbone Provider Networks (BPN)– connect SPNs over wider areas to extend customer VLANs.
- Data Center Networks – to connect multiple servers, to access data fast

- Essential - High Resiliency
- Preferred – Low Complexity, Reduced Resource Usage

Switched Network - Challenges

- Switched networks use meshed topologies to provide physical redundancy
- To carry broadcast, multicast and unicast traffic without looping switched networks use loop-avoidance protocols
- Loop-avoidance protocols construct a logical tree topology on the physical meshed topology
  - Tree algorithms such as spanning tree or Dijkstra trees are used for the purpose
  - Frames are constrained to travel along the logical tree paths
  - Links under-utilized
- On network component failures, the trees need to be re-constructed
  - Contributes to convergence latency – and impairs network performance
- Spanning tree based protocols – have high convergence latency on root switch failures
- Dijkstra tree based protocols – incur high computational overhead

A Clean Slate Solution

- A new tree algorithm that supports multiple pre-constructed non-looping paths
- A simplify protocol for tree construction and faster convergence
- Protocol decouple broadcast and unicast frame forwarding paths
  - improves link utilization
- Protocol provision faster failure detection and dissemination
- Isolates failure impacts

The Solution: A Meshed Tree Protocol Based on the Meshed Tree Algorithm

Meshed Tree Algorithm Makes a Difference

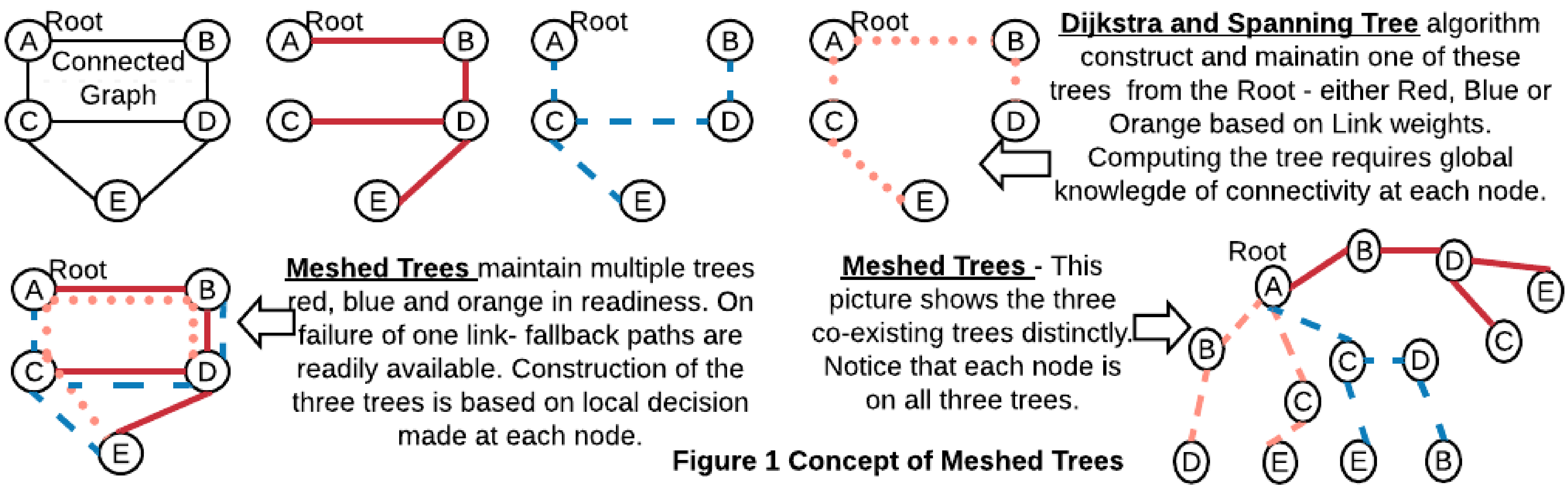
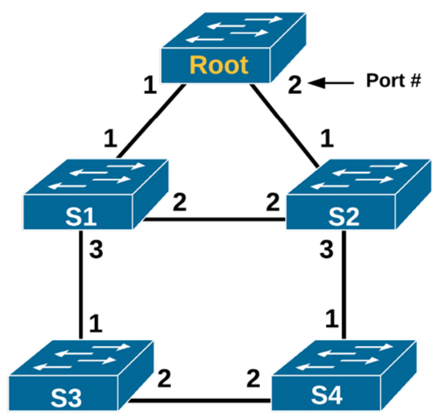
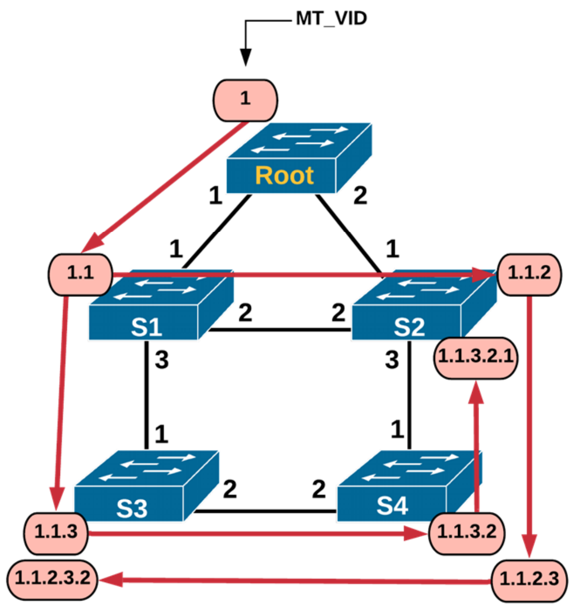


Figure 1 Concept of Meshed Trees

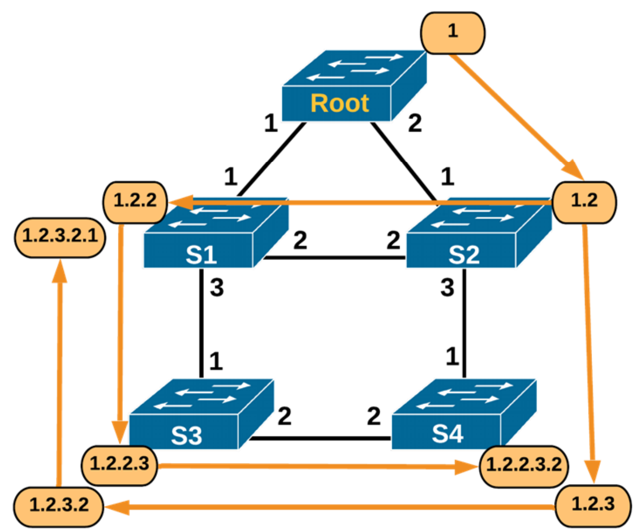
Constructing Meshed Trees in Bridged Networks
the Virtual Identifier (VID) approach



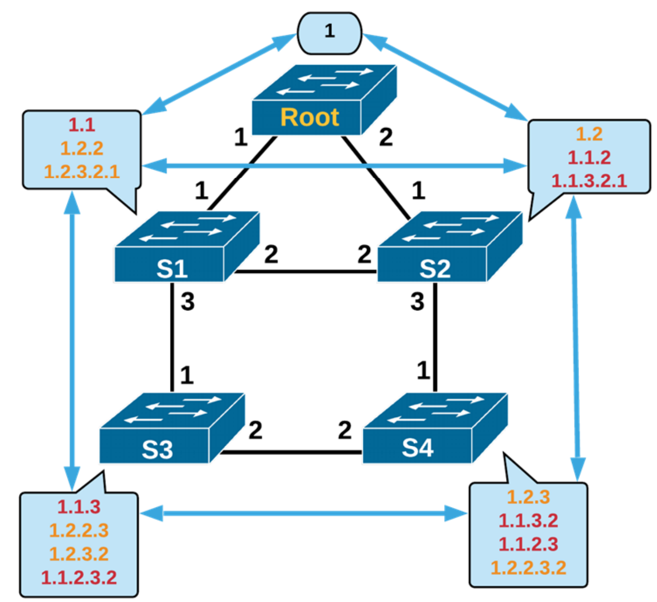
A. A 5 switch Topology. Root Switch is designated and assigned a VID = 1



B. The pink boxes show a set of VIDs, that provide two paths from Root to S2, S3 and S4

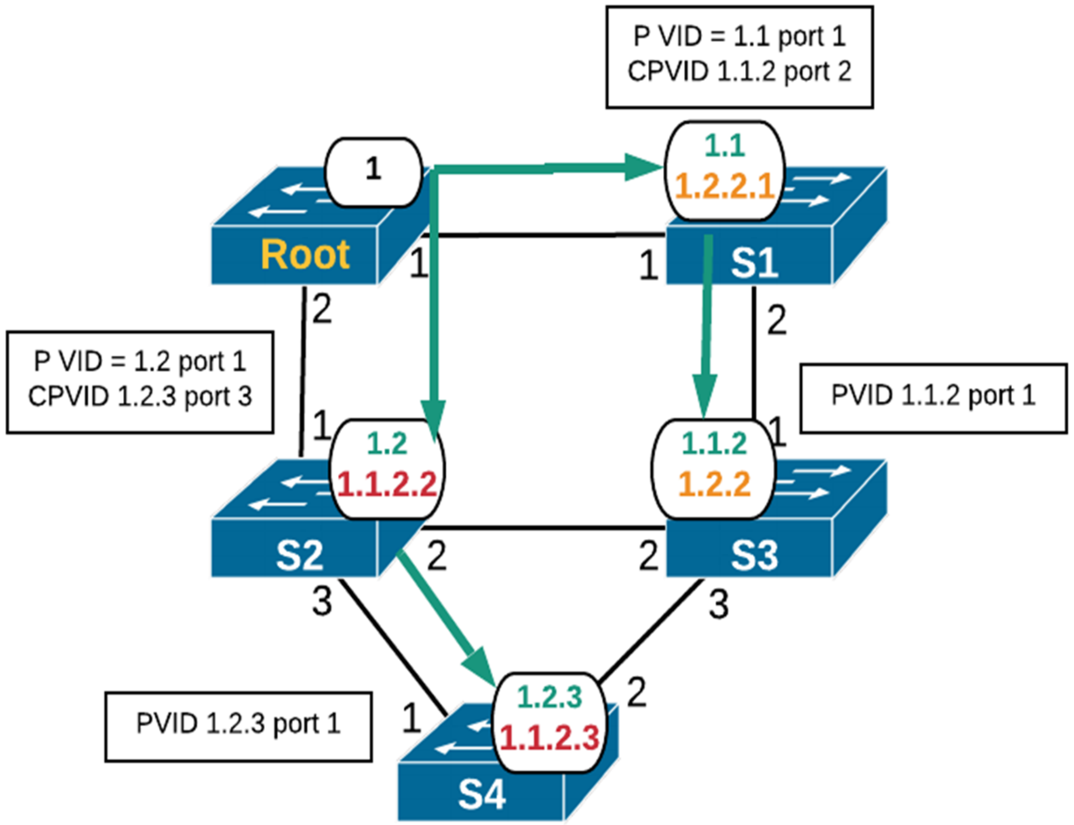


C. The yellow boxes show another set of VIDs, that provide two other paths from Root to S1, S3 and S4



D. The merger of the two sets of VIDs sorted and stored at the switches. The least hop VID is the preferred path. On its failure other paths are ready.

The Broadcast Tree



- Fig. D from previous slide is redrawn showing the Primary VID (PVID) and their ports of acquisition.
- A parent stores the PVID of a downstream switch that has its PVID from this switch as the Child PVID (CPVID).
- PVIDs and CPVIDs define the broadcast tree

Prototype Evaluation using the GENI testbed

- The Global Environment for Network Innovations (GENI) testbed[1] was used for testing the Meshed Tree Protocol (MTP) against Rapid Spanning Tree Protocol.
  - Custom C implementation of MTP
  - Open view Switch (OvS) has an implementation of RSTP

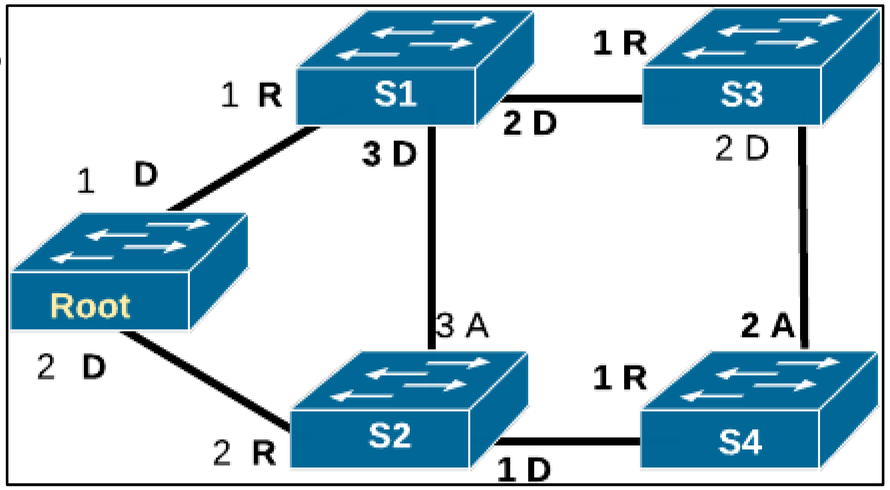
- 3 Topologies were created for testing
  - 5, 8, and 17 switches

[1] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri,R. Ricci, and I. Seskar, “Geni: A federated testbed for innovative network experiments,” Computer Networks, vol. 61, pp. 5 – 23, 2014. Special issue on Future Internet Testbeds – Part I.

5 Switch Topology – Several Failure Test Cases

Failure Case [device(port)]	Port	Failure Detection Latency	Protocol Recovery Latency	Convergence Latency	Port Role /State changes	Topology Control Notifications
Root(2)	D	5.115s by S2	3.519s	8.634s	9	24
S1(1)	R	S1 Initiates	3.523s	3.523s	13	20
S1(3)	D	4.030s by S2	2.999s	7.029s	3	16
S1(2)	D	4.810s by S3	03.018s	7.828s	8	25
S3(1)	R	S3 Initiates	18ms	18 ms	5	26
S3(2)	D	4.733s by S4	3.164s	7.897s	3	18
S4(1)	R	S4 Initiates	9ms	9 ms	5	none

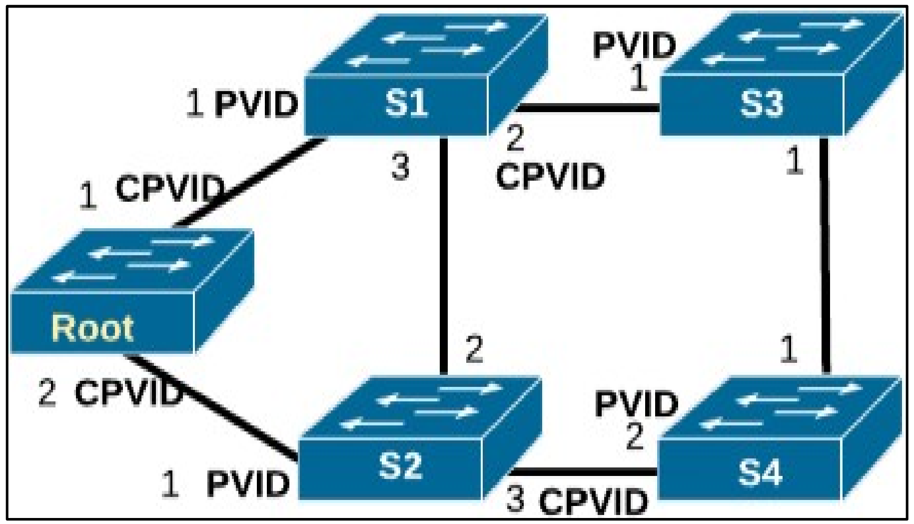
RSTP



D- Designated Port, R- Root Port, A- Alternate Port

Failure Case [device(port)]	Port	Failure Detection Latency	Protocol Recovery Latency	Convergence Latency	Number of messages
Root(1)	CPVID	1.15s	1.5 ms	1.15s	2
S1(1)	PVID	2.71s	0.14 ms	2.71s	3
S1(3)	No Impact				
S1(2)	CPVID		1.8 ms	1.8 ms	4
S3(1)	No Impact				
S3(2)	PVID		1.77 ms	1.77 ms	2
S4(1)	PVID		0.7 ms	0.7 ms	2

MTP

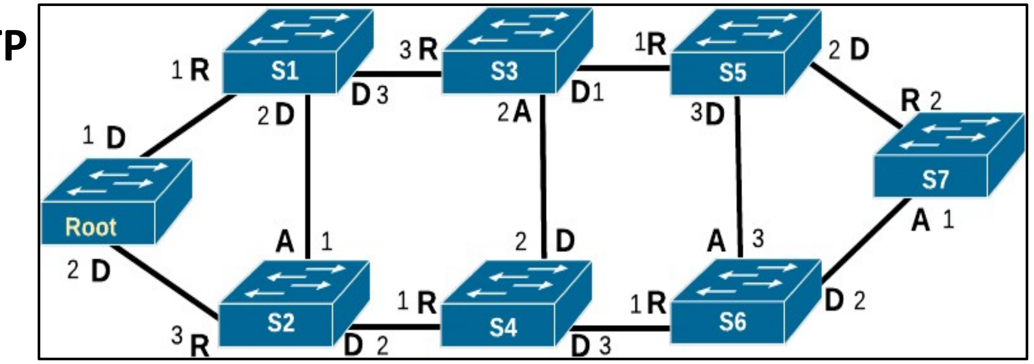


Note vast difference in MTP vs RSTP protocol recovery latencies ms to seconds.

8 Switch Topology Results

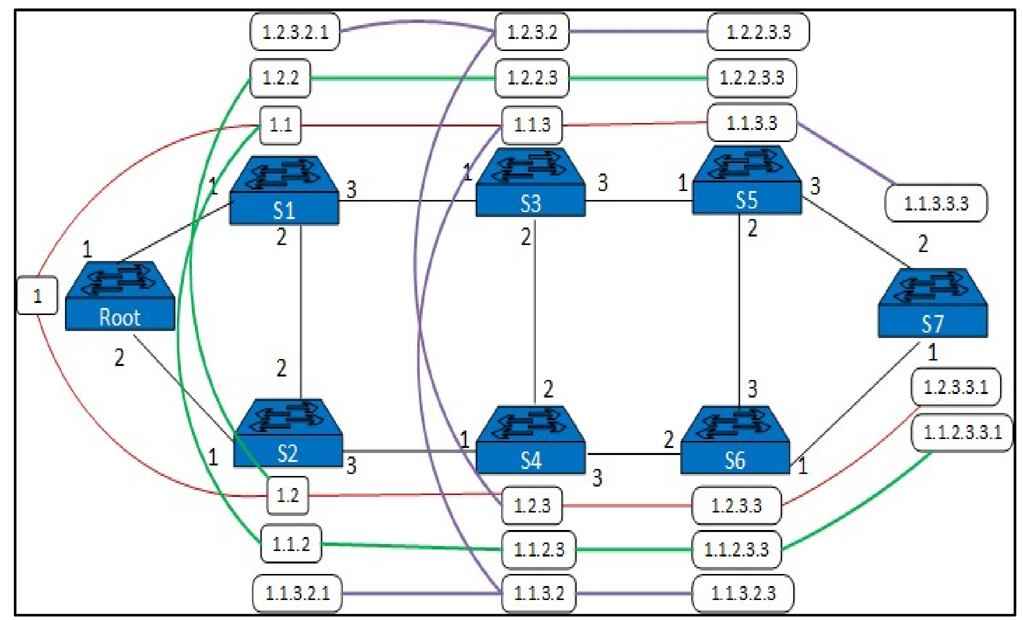
Failure Case [device(port)]	Failed Port role	Failure Detection Latency	Protocol Recovery Latency	Convergence Latency	Port Role /State changes	Topology Control Notifications
Root(1)	D	5.037s by S1	3.021s	8.058s	22	75
S1(1)	R	S1 initiates	3.032s	3.032s	19	68
S1(2)	D	4.023s by S2	3.005s	7.028s	3	37
S1(3)	D	5.206s by S3	3.027s	8.233s	13	59
S4(1)	R	S4 initiates	2.528s	2.528s	15	72
S4(2)	D	5.017s by S4	3.004s	8.021s	3	37
S4(3)	D	5.526s by S6	3.014s	8.540s	6	30
S5(1)	R	S5 initiates	3.525s	3.525s	15	40
S5(2)	D	4.199s by S7	3.012s	7.211s	6	35
S5(3)	D	5.018s by S6	3.005s	8.023s	3	39
S7(2)	R	S7 initiates	12 ms	12 ms	3	36

RSTP



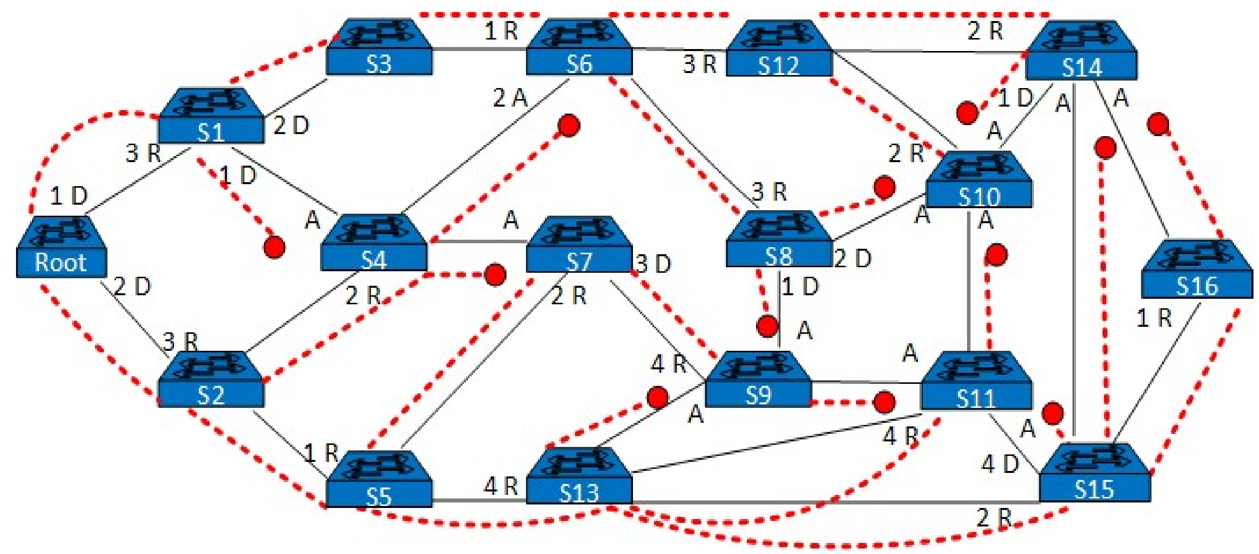
Failure Case [device(port)]	Port	Failure Detection Latency	Protocol Recovery Latency	Convergence Latency	Number of messages
Root(1)	CPVID	1.740s	14.6ms	1.74	6
S1(1)	PVID		17ms	17ms	4
S1(3)	CPVID	1.024s	15ms	1.024	5
S4(1)	PVID		6ms	0.006	4
S4(3)	CPVID	2.407s	7ms	2.407	4
S5(1)	PVID	2.290s	< 1ms	2.29	4
S5(3)	CPVID	1.046s	5ms	1.046	3
S7(2)	PVID	2.756s	< 1ms	2.756	0

MTP -
3 Paths Stored at every switch in order of preference



17 Switch Topology Results - RSTP

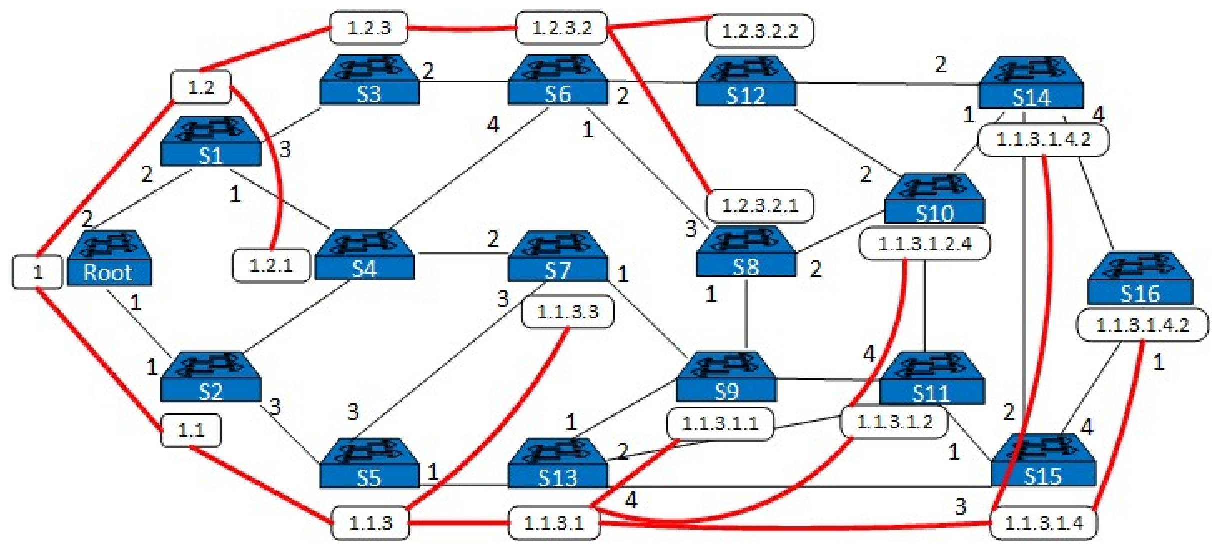
Failure Case [device(port)]	Failed Port role	Failure Detection Latency	Protocol Recovery Latency	Converge nce Latency	Port Role /State changes	Topology Control Notifications
Root (1)	D	4.501s by S3	3.462s	7.963s	26	100
S1 (1)	D	5.024s by S4	3.010s	8.034s	3	80
S1 (2)	D	4.086s by S3	3.028s	7.112s	10	80
S1 (3)	R	S1 Initiates	40ms	40ms	20	110
S7 (2)	R	S7 Initiates	24ms	24ms	3	90
S7 (3)	D	4.680s by S9	3.019s	7.699s	6	85
S8 (1)	D	3.231s by S8	3.000s	6.231s	3	84
S8 (2)	D	3.998s by S10	3.001s	6.999s	3	84
S8 (3)	R	S8 Initiates	32ms	32ms	10	106
S14 (1)	D	4.466s by S10	3.007s	7.473s	3	93
S14 (2)	R	S14 Initiates	0.025	25ms	3	100
S15 (2)	R	S15 Initiates	3.054s	3.054s	30	153
S15 (4)	D	5.475s by S11	3.011s	8.486s	3	80
S16 (1)	R	S16 Initiates	15ms	15ms	5	88



Note spanning tree paths

17 Switch Topology Results - MTP

Failure Case [device(port)]	Port	Failure Detection Latency	Protocol Recovery Latency	Convergence Latency	Number of messages
Root (1)	CPVID	2.763s –S2	36ms	2.763	9
S1 (1)	NO IMPACT				
S1 (3)	CPVID	2.726s -S3	5ms	2.726	2
S1 (2)	PVID	-	11ms	11ms	4
S7 (1)	NO IMPACT				
S7 (2)	NO IMPACT				
S7 (3)	PVID		5.4ms	5.4ms	2
S8 (2)	NO IMPACT				
S8 (3)	PVID	2.450-S6	7ms	2.45	3
S14 (2)	PVID	1.911 -S12	6ms	1.911	2
S14 (4)	NO IMPACT				
S15 (3)	PVID		12ms	12ms	5
S15 (4)	CPVID	2.453s -S16	3ms	2.453	2
S16 (1)	PVID	2.946s- S15	6ms	2.946	1



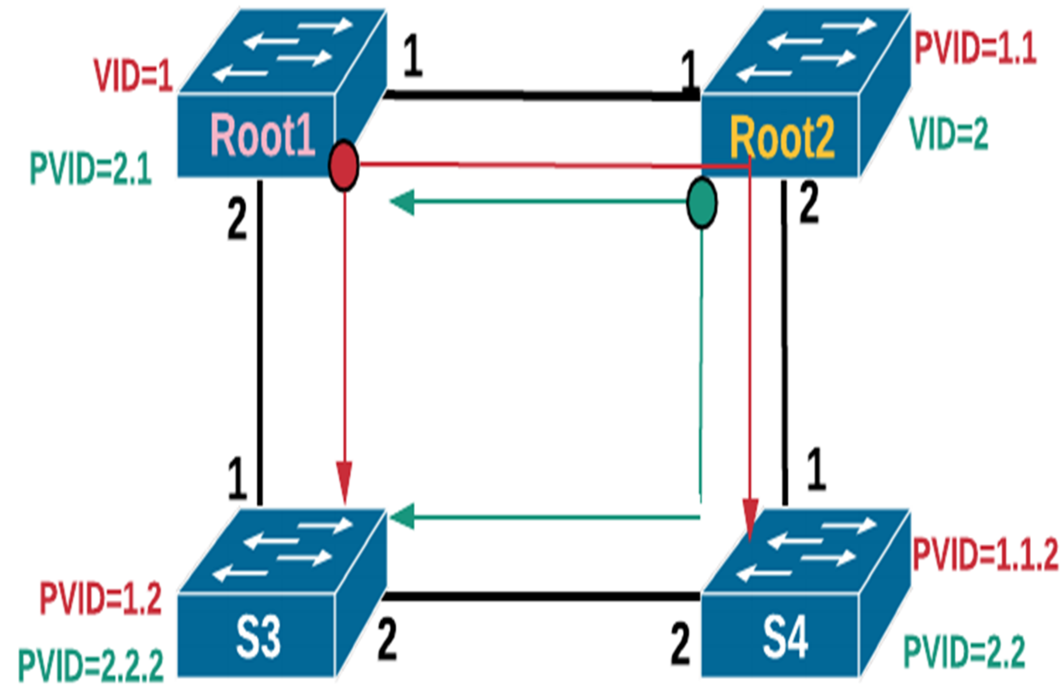
Only Broadcast Tree Shown

MTP broadcast tree covers all switches

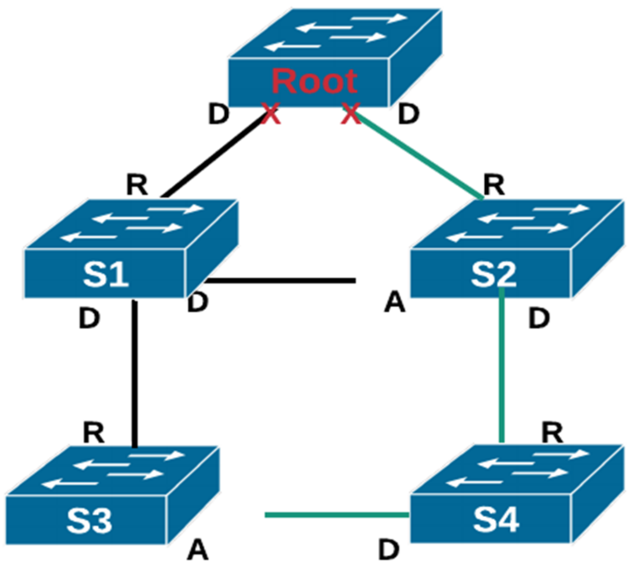
Optimal Root Redundancy with Meshed Tree Protocol

Two Root Implementation with MTP

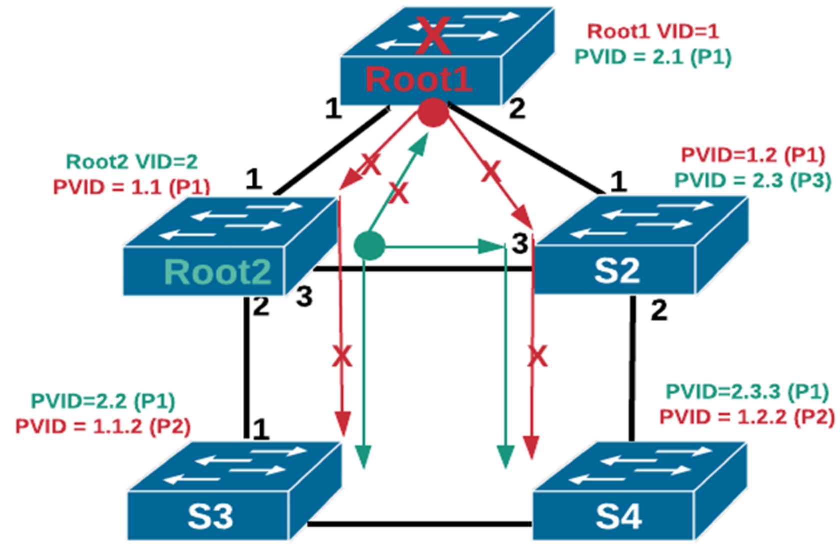
- Root1 predesignated and assigned VID =1
- Root2 predesignated and assigned VID =2
- Two trees – red and green. (Meshing within each tree not shown)
- Each tree constructed in a manner similar to the explanation in slide 7
- Default root is Root1
- Root1 fails – Root2 and green tree takes over



Root Failure Scenario (RSTP vs MTP)



RSTP: On root switch failure, tree is split into two
Each segment declares a root independently for that segment.
Race conditions to resolve a unique root can delay recovery significantly



MTP: On Root1 switch failure. Root2 and S2 detect failure first. Remove VIDs from Root1.
Root2 and S2 inform their neighbors.
Red tree is pruned and all switches loose VIDs from Root1.
All switches fallback to VIDs from Root2.
Failure recovery is very fast.

Root Failure Performance

- RSTP Failure Detection Time
  - Based on missing 3 hellos by neighbors
  - Hello interval controlled by network diameter
- RSTP Root Election Time
  - All switches collaboratively elect a new root switch
- RSTP New Tree Construction Time
  - All switches then construct a new tree from the new root

Test Topologies:

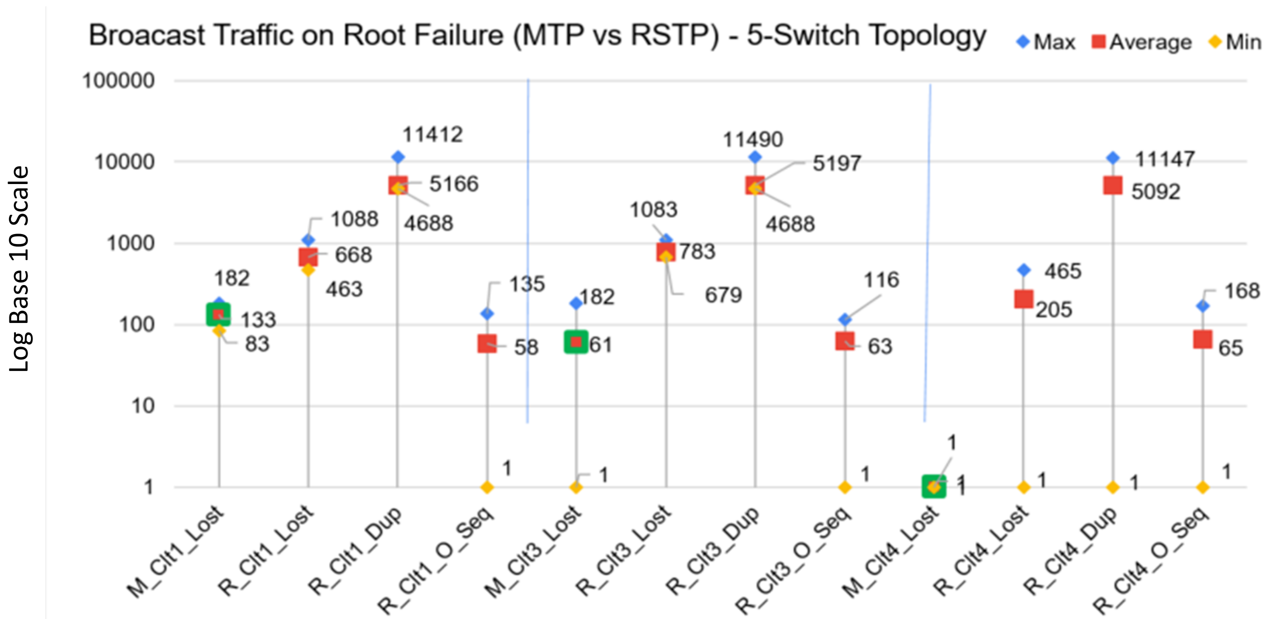
- 5 switches with 5 clients
- 10 switches with 10 clients
- 17 switches with 10 clients

- MTP Failure Detection Time
  - Switches take action on missing one hello
  - Hello intervals of 0.5 sec tested
  - MTP's hysteresis approach and the backup VIDs avoid flooding the network on false failure detection
- MTP has no root election
  - Required number of roots are pre-designated.
  - Depends on the downtime acceptable by the network
- MTP New Tree Shift Time
  - On the failure of the primary tree all switches shift to the meshed tree from the secondary root.

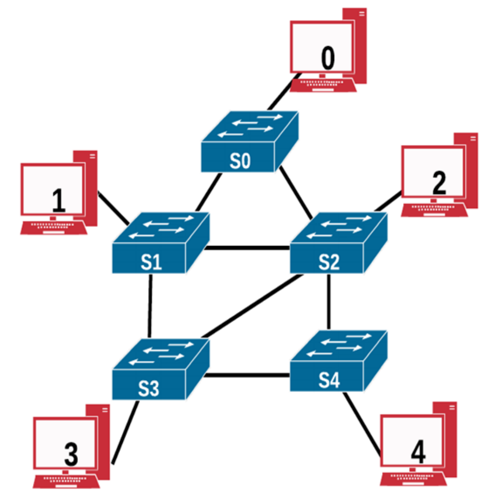
Broadcast Traffic Impact (MTP and RSTP):

- Lost frames,
- Duplicated frames,
- Out-of-Sequence frames

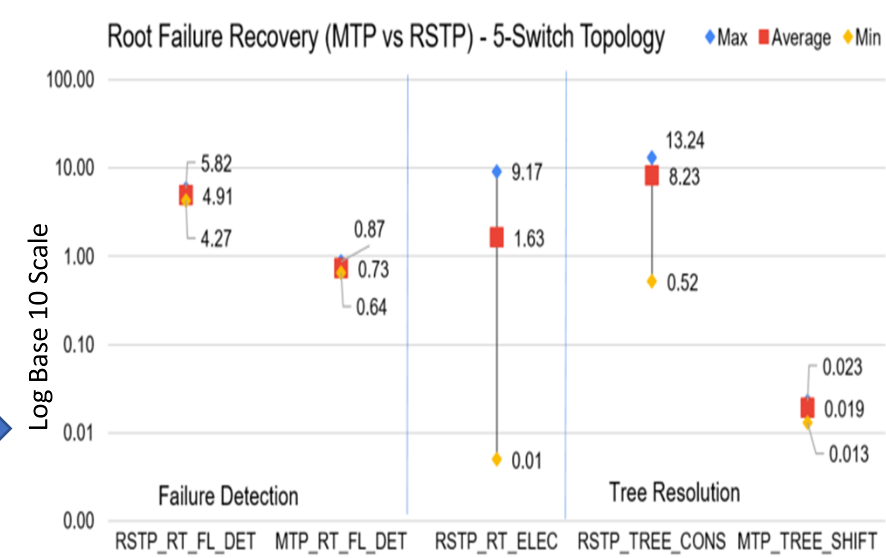
The 5-Switch 5-Clients Topology - Performance RSTP vs MTP



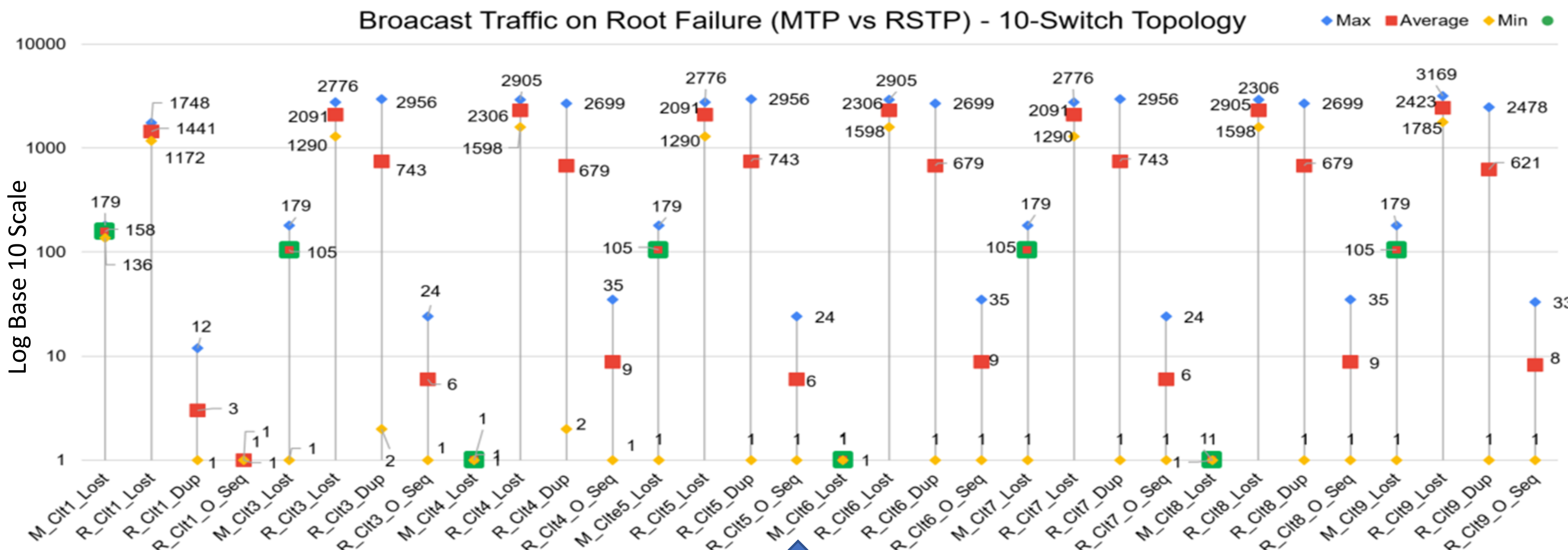
M_Cltx_Lost – Frames lost by client 'x' in MTP
R_Cltx_Lost – Frames lost by client 'x' in RSTP
R_Cltx_Dup – Duplicate frames received by client 'x' in RSTP
R_Cltx_O_Seq– Out of Sequence frames received by client 'x' in RSTP

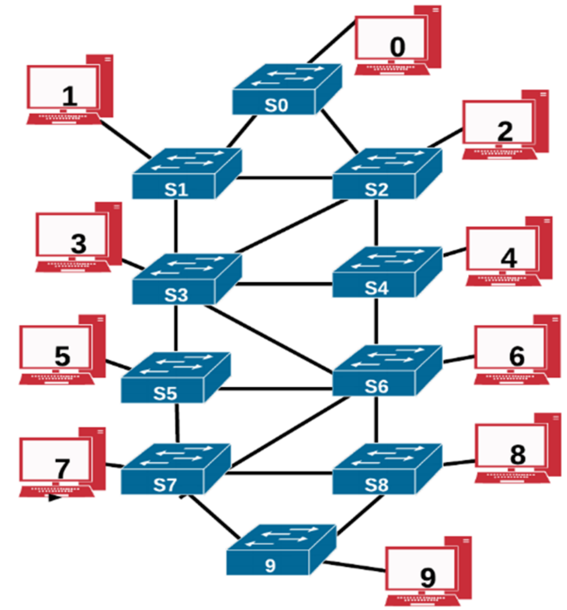


RSTP_RT_FL_DET – RSTP Root Failure Detection Time
MTP_RT_FL_DET – MTP Root Failure Detection Time
RSTP_RT_ELEC – RSTP Root Election Time
RSTP_TREE_CONS – RSTP New Tree Construction Time
MTP_TREE_SHIFT – MTP Shift Time from Primary to Secondary Tree



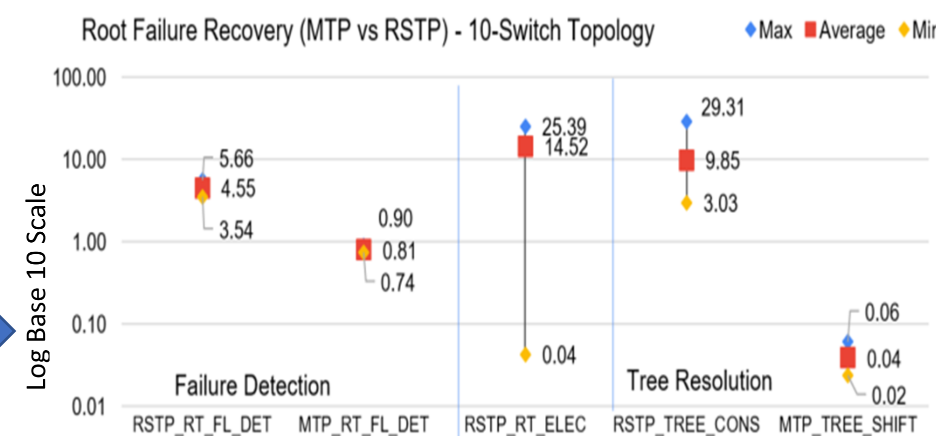
The 10-Switch 10-Clients Topology



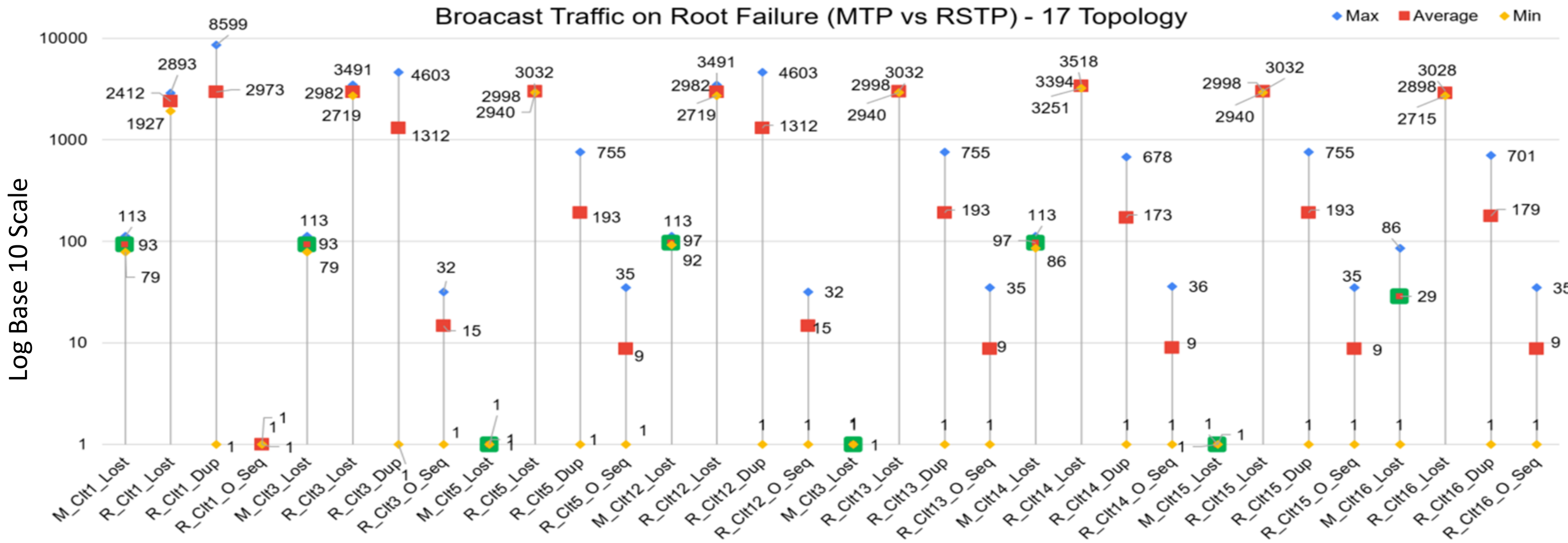


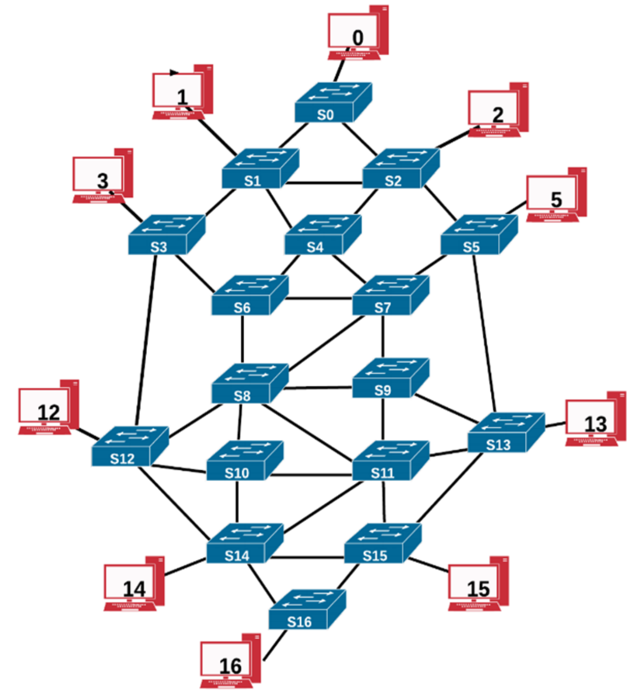
M_Cltx_Lost – Frames lost by client ‘x’ in MTP
R_Cltx_Lost – Frames lost by client ‘x’ in RSTP
R_Cltx_Dup – Duplicate frames received by client ‘x’ in RSTP
R_Cltx_O_Seq– Out of Sequence frames received by client ‘x’ in RSTP

RSTP_RT_FL_DET – RSTP Root Failure Detection Time
MTP_RT_FL_DET – MTP Root Failure Detection Time
RSTP_RT_ELEC – RSTP Root Election Time
RSTP_TREE_CONS – RSTP New Tree Construction Time
MTP_TREE_SHIFT – MTP Shift Time from Primary to Secondary Tree

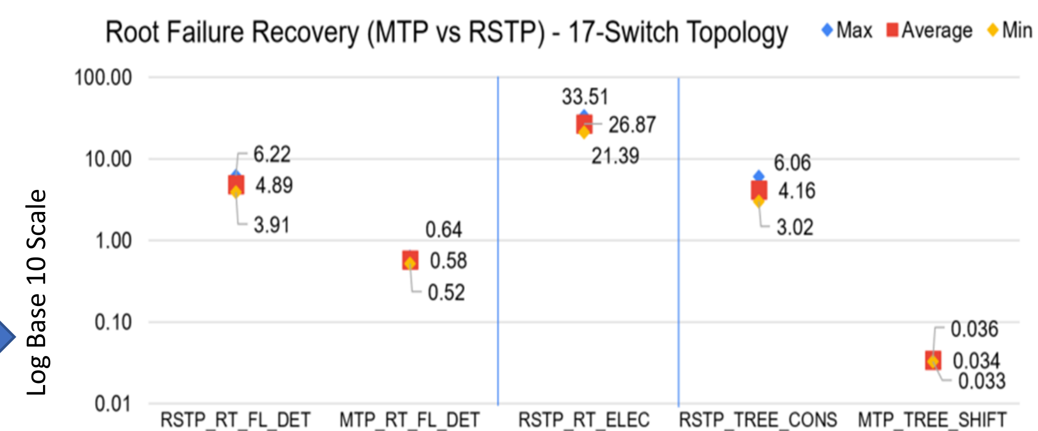


The 17-Switch 10-Clients Topology





M_Cltx_Lost – Frames lost by client ‘x’ in MTP
R_Cltx_Lost – Frames lost by client ‘x’ in RSTP
R_Cltx_Dup – Duplicate frames received by client ‘x’ in RSTP
R_Cltx_O_Seq– Out of Sequence frames received by client ‘x’ in RSTP

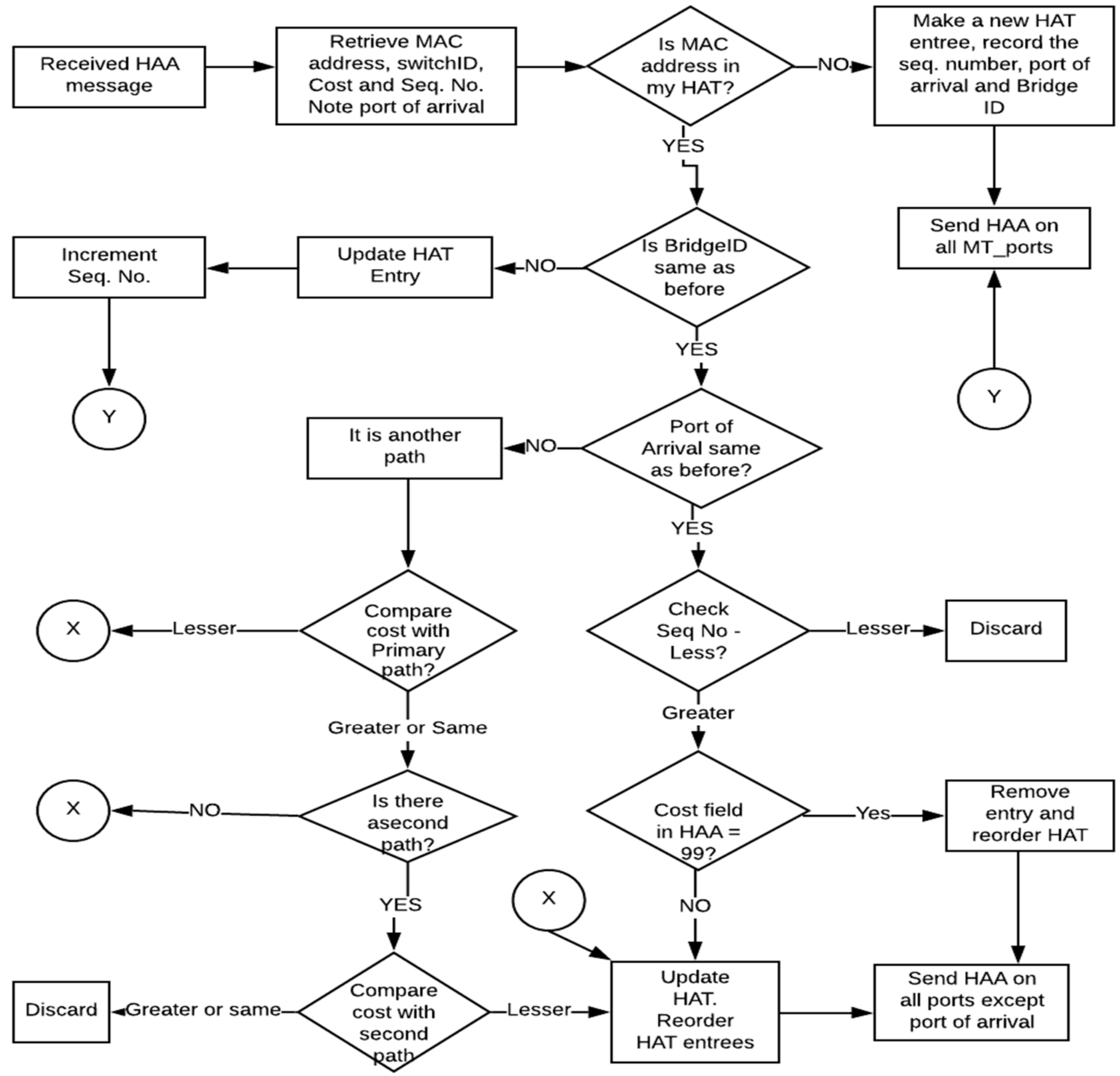


RSTP_RT_FL_DET – RSTP Root Failure Detection Time
MTP_RT_FL_DET – MTP Root Failure Detection Time
RSTP_RT_ELEC – RSTP Root Election Time
RSTP_TREE_CONS – RSTP New Tree Construction Time
MTP_TREE_SHIFT – MTP Shift Time from Primary to Secondary Tree

Unicast Frame Forwarding with MTP

- MTP does not block any ports from forwarding frames
- Unicast frames can use paths independent of broadcast tree paths
- Multiple paths to reach host devices are stored in the switches
    - On the failure of the first path, next path is ready for use
    - Very low failure recovery latency
    - Low loss in unicast frames
    - Improved link utilization as unicast frames use paths not used by broadcast frames
    - Unicast frames take shorter paths
- Current Implementation stores two paths for every host device
- Switches populate a host address table (HAT)
- Switches advertise any changes to their HAT

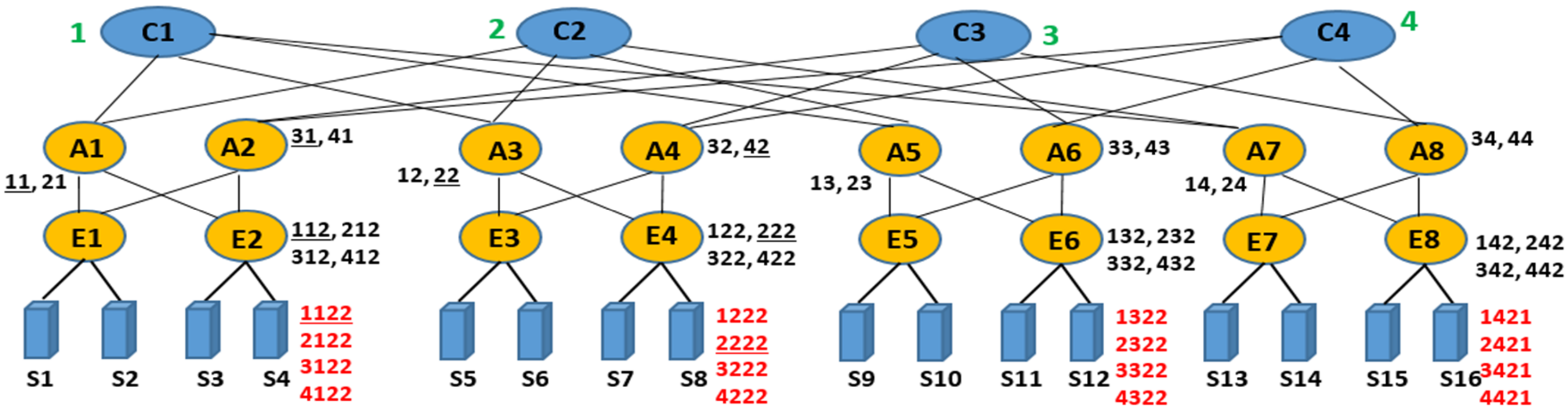
Flow chart for updating HAT



MTP in Tree Structured Data Center Networks

- Data Center Networks (DCNs) support multiple clusters, where a cluster supports hundreds of racks, and each rack supports tens of servers
- Servers communicate with each other
  - Desired high rates with minimum hops
- DCN should behave as huge non-blocking switch
- Current Direction – high redundancy topologies and use of existing routing protocols, equal cost multi path routing ….
- Meshed Tree Protocols can improve the performance of tree structured DCNs while reducing the operational complexity and redundancy

Multi Rooted Meshed Trees on FAT Tree Architectures



MTP VIDs can be used to provide the routing addresses for Core (C) switches, aggregate (A) switches, Edge (E) switches and servers in the FAT Tree architecture – this voids the need for routing protocols and IP addresses

- Core switches are assigned unique VIDs 1, 2, 3 etc.
- All other switches automatically get their routable addresses (VIDs)
- Each device has multiple VIDs and thus multiple paths

Using a single protocol - the MTP, the following functions can be achieved

- Address assignments to all devices
- Caching of VIDs to server addresses
- Unicast traffic forwarding between servers
- Broadcast traffic forwarding
- Load balancing using the multiple VID paths

Multi Rooted Meshed Trees for Tree based DCNs

- Meshed Trees can be adapted for any tree based DNC architecture
- Significant performance improvements can be achieved by connecting the core switches, as MTP VIDs will not allow for looping of frames
- Study – ongoing