Building an MPLS-TP Simulator

Vishal Negi, Umang Kumar, Tulika Pandey, Ashwin Gumaste
Dept. of Computer Science and Engineering, Indian Institute of Technology, Bombay, Mumbai, India 400076.
Ministry of Communication and Information Technology, New Delhi 110,003.
Email: {vishalng, umang}@cse.iitb.ac.in, ashwing@ieee.org

Abstract**:** Service providers are experiencing a challenge that is caused by the explosive growth in demand for high-speed connections to and across metropolitan, regional and core networks. Operators are under pressure to increase the capacity of their networks to meet these ever growing requirements. To improve service agility and network efficiency and reduce costs throughout their entire infrastructures, providers are evaluating ways to use packet transport everywhere. Carrier Ethernet is a new mechanism that is being proposed to consolidate the packet optical transport network. Multiprotocol Label Switching-Transport Profile (MPLS-TP) – a variant of Carrier Ethernet offers a potentially important addition within the packet transport toolset. Operations, Administration, and Maintenance (OAM) functionality is central to this new standard. This helps service creation and assurance thus providing an extension to packet transport capabilities within emerging multi-layer infrastructure architectures. This paper aims at exploring the performance of MPLS-TP network using an extensive simulation model.

I. Introduction

Carrier Ethernet is being proposed as the next generation of data transport for operator networks. Two variants of Carrier Ethernet are being considered – from the IEEE the 802.1Qay or PBB-TE and from the ITU/IETF the MPLS-Transport Profile or MPLS-TP. The Multiprotocol Label Switching – Transport Profile (MPLS-TP)[1-3] is a packet transport technology based on a subset of MPLS features [1] with additional transport functionalities such as comprehensive Operations, Administration and Maintenance (OAM)[5] capabilities, restoration and survivability, data-plane/control-plane separation, and static provisioning of bidirectional services, all of which makes MPLS-TP a carrier class solution. The rich installed base of MPLS and the key MPLS functionalities such as Quality of Service (QoS), scalability, traffic engineering and Layer 2 packet forwarding give MPLS-TP a head-start with operators. The result is the ability to provide network operators with full control over their packet networks meeting the demands of next generation services at unmatched price-points – especially when considered against SONET/SDH transport. MPLS-TP is somewhat backward compatible with MPLS, thus giving operators an excellent user/customer base that is already accustomed to provisioning such a technology in their networks. MPLS-TP allows the operator more control and monitoring facilities – both of which are quintessential to meeting the demands of a carrier-class network. Further, from the perspective of end-to-end communication, MPLS-TP, meets the user requirements at the edge, aggregation and core of the network, thus finding way to solve most customer requirements.

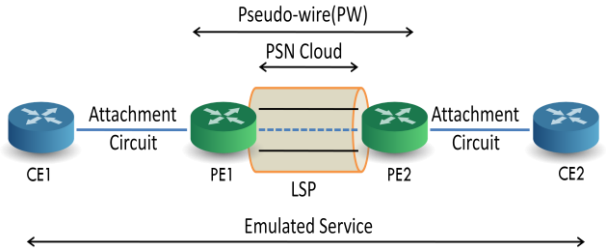

Fig. 1. Major Constructs: Pseudowires and LSPs.

MPLS-TP transport paths are statically provisioned via a Network Management System (NMS) or can be dynamically provisioned through a distributed control plane. The control plane is mainly used to provide restoration functions for improved network survivability in the presence of failures and it facilitates end-to-end path provisioning across network or operator domains. The operator has the choice to enable the control plane or to operate the network by means of an NMS.

In MPLS-TP ELINE services are set up as LSPs that facilitate Ethernet transport using the MPLS forwarding plane. Rather than use an automated control plane and the IP based control plane, the MPLS-TP implementation works with an independent control plane that is completely disassociated from the forwarding plane. MPLS-TP further allows creation of service attributes that facilitate the operator to identify a service instance, as well as protect, restore the monitor the service instance. Shown in Fig. 1 is an example of MPLS-TP providing a service to customer edges CE1 and CE2 via the provider edge LSRs (PE1 and PE2). The LSP created between the provider edges is considered as a service instance, and a control plane called the Generic Associated Channel (an in-band control plane) is used to set up service identifiers, and perform OAM features. MPLS-TP traffic can be provisioned over LSPs, or over pseudo-wires (PW) or over Multi-Segment-Pseudo-Wires (MS-PWs).

A. MPLS-TP OAM

MPLS-TP OAM [6, 9] is intended to reduce network operations complexity associated with network performance monitoring and management, fault management, and protection switching. These are required to operate without any IP layer functions. Dedicated OAM packets are interspersed into the associated user traffic flows. These OAM packets are created and processed by Maintenance End Point (MEPs). MEPs are service end-points that define an MPLS-TP tunnel ingress and egress nodes. In addition to MEPs that generate and sink OAM packets, Maintenance Intermediate Points or MIPs can also process these OAM packets and may collect data or raise alarms. The OAM features are as shown in Fig. 2.

Two important components of the OAM mechanisms are the G-ACh and the Generic Alert Label (GAL). They

allow an operator to send any type of control traffic into a PW or an LSP. In the context of transport networks, ACH was generalized to enable the same associated control channel mechanism to be used for Sections, LSPs, and PWs. The associated control channel, thus generalized is known as Generic Associated Channel (G-ACh)[4].

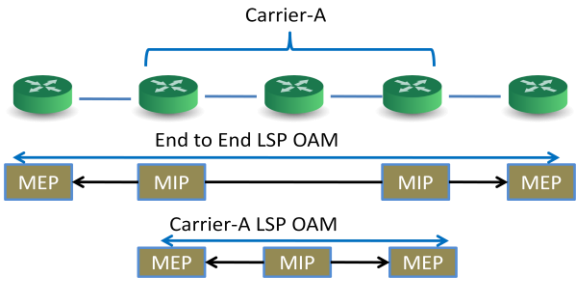

Fig. 2. LSP monitoring using OAM flows

This generic function is capable of carrying user traffic, OAM traffic, and management traffic over either a PW or an LSP. Using G-ACh, the same OAM mechanism can be unified for LSPs and PWE, enabling the same functionality for both and ease of implementation.

II. MPLS-TP Simulator

Given the interests that operators have shown in deploying MPLS-TP, we need tools that facilitate the statistical analysis of the MPLS-TP domains and their components as well as study their characteristics. A simulator will hence allow us to evaluate the performance of an MPLS-TP network without the need to setup an actual MPLS-TP network. What is intrinsic to such a simulator is the carrier-class MPLS-TP LSR architecture that facilitates the provisioning of MPLS-TP tunnels in a network.

The goal of this simulator is to model an MPLS-TP node and to understand the behavior of MPLS-TP network under different traffic conditions. The simulator is implemented in C++ on a Linux platform. It provides features that setup Label Switched Paths (LSPs) between various pairs of nodes based on a traffic connection (adjacency) matrix. These LSPs are Co-routed bidirectional transport paths traversing an MPLS-TP domain. The nodes that provide interface to an MPLS-TP network are called Terminating-Provider Edges (or T-PEs).

To emulate a service, a PW is setup between the CEs and data is transported on an MPLS-TP LSP. Switching-Provider Edges (or S-PEs) in the network allow for simulating the MS-PW. The MS-PW is thus the client of underlying MPLS-TP service layer. The connection matrix allows for setting up the transport paths based on Open Shortest Path First (OSPF) routing.

Customer traffic is first encapsulated within the transport-service layer network by the T-PE and then forwarded along the appropriate transport path. The requirement for MPLS-TP specifies that it must not modify the MPLS forwarding architecture and must be based on existing PW and LSP constructs, thus facilitating good backward compatibility with MPLS/PW definitions. Therefore the MPLS-TP forwarding is based on the Labels and Label operations (Push, Pop, or Swap) at each MPLS-TP node. The strategy used for Label distribution in our simulator is based on a variant of *Downstream on Demand*. We argue that using downstream unsolicited strategies would make the network non-carrier-class.

Quality-of-Service (QoS) mechanisms are required in the packet transport network to ensure the prioritization of critical services. A transport network must provide the means to meet the QoS objectives of its clients. The simulator provides 8 priority levels and performs WFQ scheduling to achieve this objective.

The above service guarantees depend on reliable and efficient operation of a Transport Network. Thus MPLS-TP requires OAM mechanism to monitor the integrity of the transport paths and provide other network monitoring and management features. Generic Associated Channel (G-Ach) is used to carry OAM traffic and distinguish it from normal data traffic thereby providing a full in-band control plane. The simulator implementation provides in-band OAM that shares fate with client traffic. Proactive Continuity Check Messages (CCM) are provided to monitor the integrity of the transport path. The CCMs generate triggers in case of failures or congestion. These triggers are then used to provide 1:1 restoration along the Protection path using Automatic Protection Switching.

A. *Simulation Methodology*

The simulator simulates the behavior of an MPLS-TP network if the native service traversing the network is a MS-PW [7]. The simulation model assumes the presence of T-PEs and S-PEs in a MPLS-TP network. The T-PE is at the Customer Edge (CE) of the network and sources or sinks customer traffic to or from an MPLS-TP network. The TPEs also act as Label Edge Routers (LERs) and provide as endpoints for LSPs and MS-PWs. These are also responsible for Label Distribution and Proactive Pseudowire monitoring. The S-PEs are capable of switching the tunnels of preceding and succeeding segments of a MS-PW [7].

The architecture of the MPLS-TP node used for the purpose of simulation is described in Fig. 3. Each individual block in the architecture represents a module and performs some functions. A node can be a T-PE or an S-PE, depending on its role in the MPLS-TP network.

The traffic arrival at each T-PE is modeled as a Poisson process. After a packet arrives at a node, it goes through a series of functions, before it is sent back into the network towards the next hop. This process continues at every node, until the packet reaches its final destination or is dropped in the process.

When the packet arrives at a node, it first goes through the *Edge Port Logic*. This module is responsible for frame classification, lookup and forwarding logic. It also is also responsible for requesting the *Memory Write Controller* to store the frame in the *Memory*. Once the forwarding decisions are made, the packet is forwarded to the *Switch Fabric*. The Switch Fabric then forwards the packet to appropriate Output port. At this point, the *Output Port Logic* is performed. At an individual output port, the packet is placed at the appropriate Priority Queue based on the priority of packet.

A WFQ scheduler is responsible for serving the Priority Queues and aggregates the resultant frames. A request is then placed to the *Memory Read Controller* to

read the appropriate frames from the *Memory*. Once the frame is removed from the memory, processing is done on the frame before it is forwarded to the output interface to be sent back into the network towards its intended destination. The delay experienced by the packet at a node is the summation of all the delays in each of the individual modules.

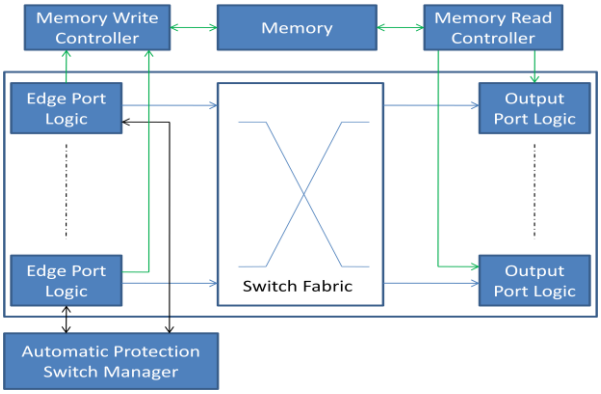

Fig. 3. Node Architecture.

B. Edge Port Logic

This module is described in Fig. 4. It is implemented at the input port. For the purpose of simulation, an input port queue is maintained. A packet is stored in the queue, until the previous packet is completely received. However, for an individual packet, processing starts as soon as the first bit arrives. Simultaneously, a request is placed to the *Memory Write Controller* to store the packet into *Memory*. Since the packet will be stored in Memory, to reduce packet delay, all the processing hence forth is done on the packet header. With the arrival of the first byte, UNI/NNI processing begins.

The *UNI-NNI Logic* is responsible for frame classification, link layer specific preprocessing and identification of Traffic Service Instance and client flow. After this step, if the packet is a data packet, the packet header is stored in Look-up Queue. If the packet is an OAM packet, it is forwarded to *Automatic Protection Switching (APS) Manager*.

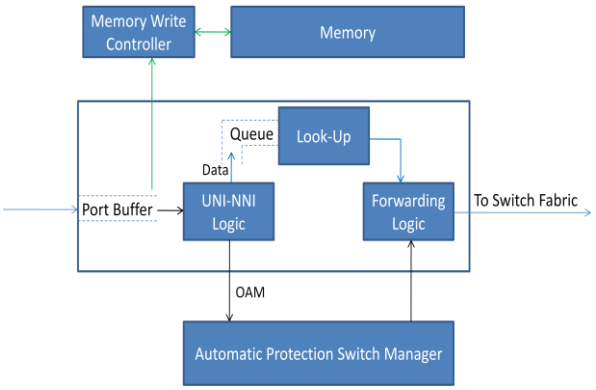

Fig. 4. Edge Port Logic

The *Look-Up module* then looks up Incoming Label Map (ILM) and FEC-To-NHLFE (FTN) table to extract Next-hop Label forwarding entries (NHLFE). The packet header is then forwarded to Forwarding Logic.

The *Forwarding Logic* then performs LSP label pop, PW label swap and then LSP label push on the packet header. From NHLFE, it identifies which output port the packet is destined to and forwards the packet header to Switching Fabric.

C. Switching Fabric

The *Switch Fabric* forwards the packet header from the input port to appropriate output port. An *arbiter* iterates over each input port and forwards the available frame to the destination port. The headers are put in the output queue at the output port.

D. Output Port Logic

This module is described in Fig. 5. When the packet header arrives at Output port, *Output Port Logic* begins. The header is placed in the appropriate *Priority Queue* according to the class of service defined for the packet. Each priority queue is assigned a weight. A *Weighted Fair Queuing Scheduler* empties each priority queues in proportion to the weight assigned to the queue. Thus available bandwidth is divided among the queues in ratio of their weights.

The scheduler then places the available headers into a packet aggregator. The aggregator then places a request to *Memory Read Controller* to read the corresponding frames from the *Memory*. Now the entire packets read from the memory are forwarded for UNI-NNI processing.

The *UNI-NNI logic* as its counterpart in the Edge port logic performs link layer specific post processing and associates an appropriate data link encapsulation with the packet. The packet is then forwarded to output interface, from where it is sent out in the MPLS-TP network towards its intended destination.

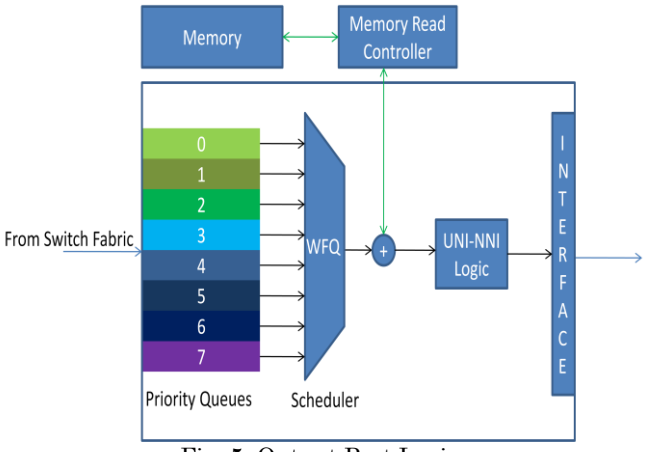

Fig. 5. Output Port Logic.

E. Automatic Protection Switch (APS) Manager

APS manager is responsible for OAM mechanisms needed for fault detection, diagnostics, maintenance, and other functions on a PW and LSP. It is an important component that is required for the 50 ms restoration time in event of failure. The T-PEs for a MS-PW form the Maintenance End Points (MEPs). The intermediate S-PEs in the Transport path form the Maintenance Intermediate Points (MIPs). Proactive OAM monitoring is done for checking the integrity and status of the transport path.

The simulator implements this by the means of CCM packets and *G-ACh*. The G-ACh acts as a container or channel that runs on the PW and carries OAM messages. The MEPs generate In-band CCM packets. These CCM packets share fate with the data-packets. The MIPs do not generate OAM packets. They simply forward these packets towards their destination MEPs. A MEP sends 3 successive CCM packets towards other MEP of the Transport path. The MPLS-TP OAM requirement specifies an interval of 3.3 ms between successive CCM packets. The MEP destined as receiver has a *Hold-off timer*. If the 3 packets are not received when the timer expires, a trigger is raised indicating the *Path-Failure*. The Transport path is then switched to the Protection path, until the Primary work path is restored again.

To section in the simulator config file can be used to simulate *Path-Failures*. The elements of this section specify which link to fail and at what time.

F. Network Topology and Traffic generation

The Network topology is built using the *Adjacency Matrix* in the config file. A section in the config file specifies what nodes are the S-PEs. Rest all nodes are designated as T-PEs.

The *Traffic Matrix* specifies how much traffic is to be generated between the specified nodes. Such nodes are T-PEs. The granularity of the links between nodes is specified by *Granularity Matrix*. The *Traffic generation module* models the packet arrival at each T-PE as a *Poisson process*, while being considerate about the constraints specified by Traffic and Granularity Matrices.

For every Transport path in the network, a Protection path (1:1) is also provisioned. These paths are generated by modifying the Adjacency Matrix and using Dijkstra's algorithm to compute the shortest path on the modified matrix.

III. Simulation Results

Using the simulation model described in the previous section, simulation was performed to evaluate the behavior and performance of MPLS-TP under various traffic loads for various packet sizes and priorities (class of traffic). The topology used for performing simulations is shown in Fig. 6. The Traffic and Granularity matrices in the config file were used to specify the T-PE pairs amongst which the LSPs will be setup. The LSPs for this experiment are all co-routed bidirectional LSPs. It should be noted that the packet arrival at each T-PE is modeled as a Poisson process.

The implementation of simulator provides for 8 priorities (classes of traffic). The lowest priority is 0 and the highest being 8. The highest priority is used for OAM. The simulation experiment was run for a simulation period of 1 second for each traffic load (10% to 90%). For each traffic load, the following metrics were measured:

1. Load vs. Delay (for various packet sizes)
2. Load vs. Jitter (for various packet sizes)
3. Load vs. Delay (for various priorities)
4. Load vs. Jitter (for various priorities)

a. Load vs. Throughput/Packet Drop

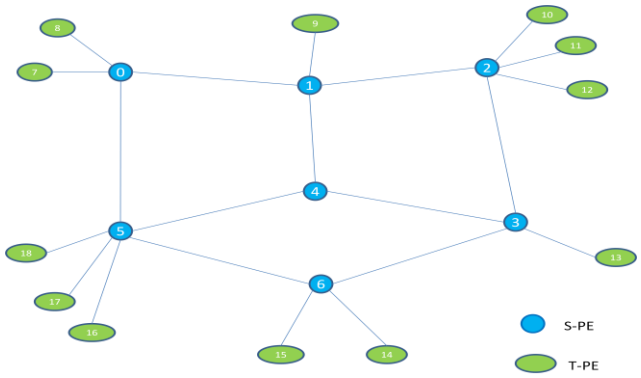

Fig. 6. Topology

A. *Load vs. Delay*

Average packet delay as a function of load, for packet sizes 64-100, 801-900 and 1401-1500 is shown in Fig. 7. It is observed that there is a steep rise in average packet delays for 70% traffic load and above.

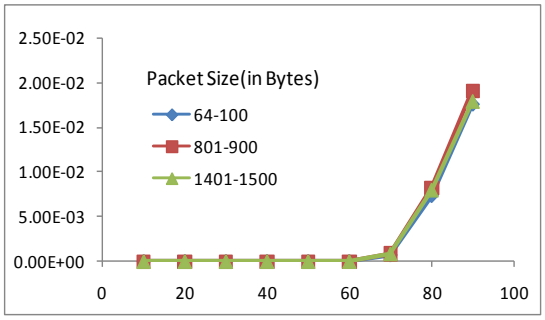

Fig. 7. Load vs. Delay (different packet sizes)

For various priorities also, it is observed that the delay increases with the traffic load and there is a steep rise in delay for traffic load of 70% and above. This is shown is Fig. 8.

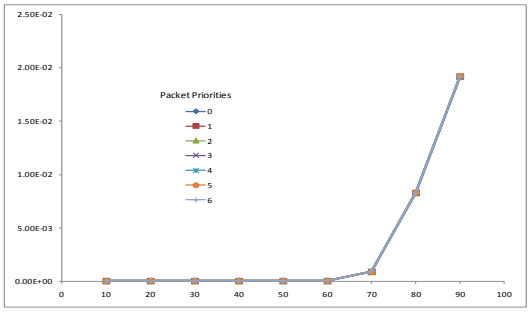

Fig. 8. Load vs. Delay (for different priorities)

B. *Load vs. Jitter*

Jitter is calculated as the average of the difference in the latencies of the packets at each node, i.e.

$Diff_n = | Laency_n - Latency_{n-1} |$, where, n is the current packet.

Thus Jitter is given by

$Jitter = \sum Diff_n/(n - 1)$, where n is the total number of packets.

It is observed that jitter as a function of traffic load depicts a saw-tooth pattern as shown in Fig. 9. This is because of router introducing alternate latencies because of packets being queued in different priority queues.

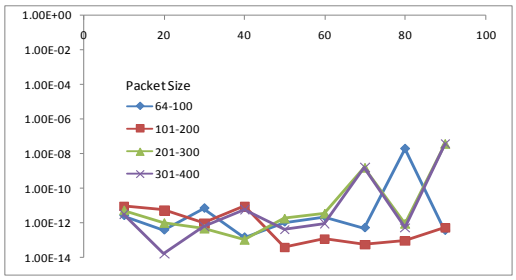

Fig. 9. Load vs. jitter (for various packet sizes)

A similar pattern can also be observed for various priorities as shown in Fig. 10.

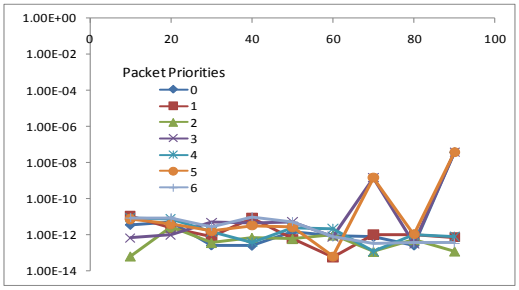

Fig. 10. Load vs. Jitter (for various priorities).

C. Load vs. Throughput/Packet-Drop

Based on the previous two results, it can be easily concluded that, as the packet delay increases with increase in load, the packet drop must also happen. The results corroborate this fact. For traffic loads of 70% and above, packets start dropping. This is shown in Fig. 11.

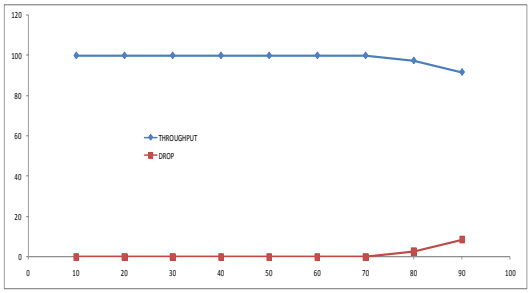

Fig. 11. Load vs. Throughput/Packet Drop

IV. Conclusion

MPLS-TP represents a new development in the larger MPLS protocol suite. MPLS-TP based packet transport networks take advantage of the cost-effectiveness and ease-of-use of pseudo wire over IP/MPLS architecture, and carefully preserves the OAM and management characteristics of legacy transport networks. Major advantages are consistent operations and OAM functions across the different network layers and the seamless interworking with IP/MPLS networks. By using IP/MPLS and MPLS-TP, service providers will have a consistent way of provisioning, troubleshooting, and managing their networks from edge to edge.

The simulator simulates the behavior of an MPLS-TP network, when the native service used is MS-PW. The results benchmark important metrics as a function of traffic load, for various packet sizes and priorities.

V. References:

[1]RFC 3031: Multiprotocol Label Switching Architecture

[2]RFC 5317: Joint Working Team (JWT) Report on MPLS Architectural Considerations for a Transport Profile

[3]RFC 5654: Requirements of an MPLS Transport Profile

[4]RFC 5586: MPLS Generic Associated Channel

[5]RFC 5860 Requirements for OAM in MPLS Transport Networks

[6]RFC 5921: A Framework for MPLS in Transport Networks

[7]RFC 5659: An architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge

[8]RFC 5960: MPLS Transport Profile Data Plane Architecture

[9]RFC 6371: Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks