Using MPLS-TP for Data-Center Interconnection

Ashwin Gumaste, Chirag Taunk, Sarvesh Bidkar, Deval Bhamare, and Tamal Das

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay, Mumbai, India - 400 076
ashwing@ieee.org,
{chiragtaunk,sarvesh,deva,tamaldas}@cse.iitb.ac.in

Abstract. Data center interconnectivity is particularly very important and essential for emerging applications such as cloud computing and financial trading. Current data center architectures are built using Ethernet switches or IP routers - both with significant cost and performance deficiencies. We propose for the first time, extending MPLS-TP into the data-center. To this end, a new look-up protocol for MPLS-TP is proposed. Autonomic communication within the data center is possible using our look-up protocol that enables fast creation and deletion of LSPs. The MPLS-TP based data-center that is architected in this paper leads to performance betterments over both IP and Ethernet. To this end, a comprehensive simulations model is also presented. Operations within the data-center using MPLS-TP are also extended to inter-data-center operations using LSP setup across a core network. OAM and performance issues are investigated.

1 Introduction

The growth of data traffic and web services has put excessive stress on current network infrastructure. Emerging applications are typically web-based and these imply the need for distributed storage and processing across the Internet – leading to the proliferation of the data-center. The data-center is becoming an important network infrastructure from the perspective of application virtualization as well as resource consolidation. A typical data-center consists of servers, storage elements and very fast switches. The latter is the key to making the data-center a success – interconnecting memory modules to servers as well as to the rest of the Internet is an efficient way. The interconnection must support the ability to provide virtualization and consolidation – amongst servers, amongst memories and across supported applications. Apart from being able to meet carrier class requirements, the switches should also be able to support high-bandwidth volume cross-connect, at low energy needs and provide low-latency guarantees. Mapping applications that reside in network attached storage (NAS) devices as well as those running at servers is daunting on account of the inability to migrate easily when one considers typical data-center tasks such as load balancing, etc. Typical network interconnection fabrics are based on complex IP-routers or MPLS LSRs. The use of high-end IP-MPLS equipment has been known to be an overkill [1]. Particularly from both cost and performance standpoints, the use of IP/MPLS routers is not necessarily justified. The closed domain of the data-centers along with the proximity of network elements

I. Tomkos et al. (Eds.): BROADNETS 2010, LNICST 66, pp. 398–410, 2012.

within the data-center implies that use of IP/MPLS routers leads to significant routing overhead, which can otherwise be achieved by simple switches, provided they can guarantee service differentiation (QoS support) and Carrier-Class features (OAM&P support). To alleviate this problem, there have been proposals [2,3] to use Ethernet switches [4] for data-center support. The strong emphasis on carrier-class support means using transport-oriented versions of Ethernet – Carrier-grade Ethernet transport platforms. Amongst the two version of carrier-grade Ethernet, the MPLS-TP (transport profile) has gathered significant attention recently, especially when compared with the Ethernet-bridging based PBB-TE (Provider Backbone Bridged Traffic Engineering) standard.

In this paper, we propose the use of MPLS-TP [6-14] as an architectural technology for use in the data-center. To this end, we use MPLS-TP in conjunction with our earlier proposed Ethernet transport technology called "Omnipresent Ethernet" [5] or OEthernet for short. The OEthernet technology uses network interconnection patterns as aids in creating a communication framework (for switching, routing and transport). In the OEthernet framework, any network graph is converted to a binary graph. This leads to binary routing and source routing – two well-known concepts in interconnection systems. However, to make these pragmatic, we propose the use of binary addresses as MPLS-TP labels. This leads to a system, whereby forwarding of MPLS-TP packets is instantaneous – without the need for a lookup table, as the corresponding bits of a label signify to which port a packet would be forwarded to. The binary forwarding mechanism when applied to an MPLS-TP framework solves the larger question of the control plane for the data-center. The resultant is an autonomic data-center, where LSP setup and tear down are accomplished through the OEthernet control plane, and forwarding is entirely based on binary addresses and source routing. The use of OEthernet mechanism with MPLS-TP makes MPLS-TP a plausible alternative for the data-center. Moreover, simulation results show significant cost, latency and energy efficiency improvements with our approach when compared with other solutions.

This paper is organized as follows: Section II is a primer on MPLS-TP while Section III describes OEthernet –for the data-center. Section IV describes the use of MPLS-TP using OEthernet from the perspective of the data-center. Section V focuses on numerical evaluation, while Section VI concludes the paper.

2 MPLS-TP Primers

MPLS was developed to focus on improving the cost and performance issues associated with core IP routers. MPLS has proved successful in carriers' converged IP/MPLS core networks. However, MPLS is an expensive technology due to the higher cost of managing the routers in the core networks. Also, MPLS is not fully-optimized for transport functions such as guaranteed QoS, protection, deterministic end-to-end delay, leading to new extensions that meet the transport needs. MPLS-TP based networks are more deterministic with the addition of features like traffic engineering, end-to-end QoS, full protection switching and fast restoration. Being a packet-based technology, MPLS-TP simplifies the networks as well as reduces the CAPEX and OPEX.

MPLS-TP is aimed to provide fault management (fault identification, fault isolation, fault recovery, resiliency), configuration management (generating statistics of configuration of network resources, updating the records), accounting management, performance management (utilization, error rate) and security management.

MPLS-TP supports bi-directional label switched paths (LSP). MPLS-TP provides end-to-end connection-oriented transport. It also provides end-to-end path protection and QoS with operation administration and maintenance (OAM) support. MPLS-TP OAM functionality is independent of the dynamic control plane, offering the services using only the data plane.

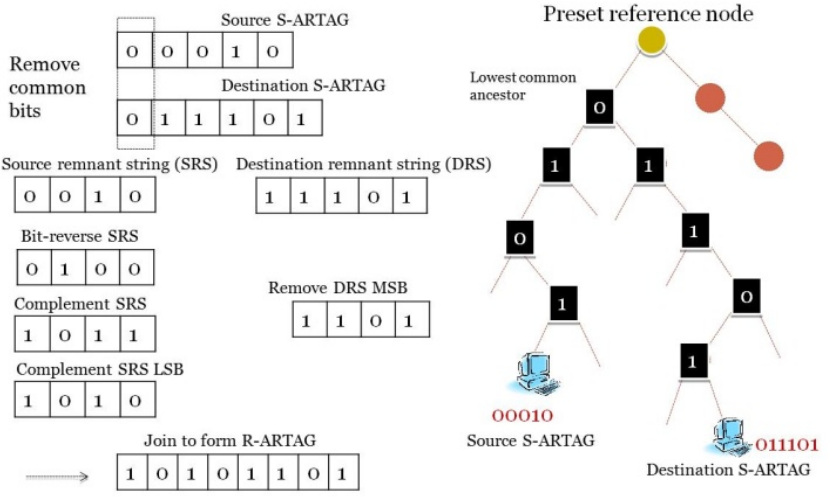


Fig. 1. Omnipresent Ethernet Architecture and Addressing

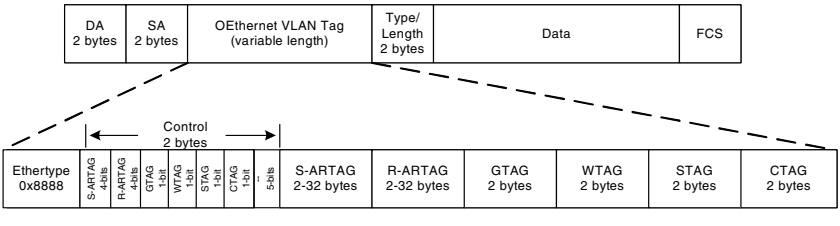


Fig. 2. OEthernet Frame Format

3 Omnipresent Ethernet in the Data-Center

Carrier Ethernet has been the new buzzword for metropolitan networks – with active worldwide deployments. Carrier Ethernet is currently available in two flavors – from the IEEE the 802.1ah/802.1Qay (PBB/PBB-TE) and from the IETF the MPLS-TP with 5+ separate drafts under circulation. We proposed the Omnipresent Ethernet [5]

or OEthernet technology as a solution to Carrier Ethernet showing superior performance. In particular, the OEthernet solution is more scalable, has lower latency, supports multipath service provisioning and consumes lower energy as compared to PBB-TE/MPLS-TP variants.

OEthernet was proposed as an end-to-end solution for communication without the use of IP or more blatantly with the sole use of Ethernet and the optical layer [5]. It has since then been investigated to provide multiple communication paradigms leading to benefits such as low latency, low cost, superior service support, lower energy requirement and resiliency [15]. Note that OEthernet has been proposed as a stand-alone communication paradigm, and we extend this to the domain of Carrier Ethernet technology, particularly from the perspective of MPLS-TP.

The OEthernet concept involves converting any network topology to a binary graph – by the addition of "dummy" nodes to smoothen-out all the nodes, resulting in a system where each node is a 1×2 interconnection element [5]. A binary graph leads to binary routing *i.e.* the facilitation of source routing with binary addresses. A single bit can now tell a node (1×2 switch) whether to go left or right. A node only has to process $(2\log N)$ bits in an $N \times N$ switch. Further, there is no need for any lookup. Each node is assumed to have its address – a binary value that determines its route from a preset node in the binary graph.

Communication in the OEthernet framework: A source node has access to the destination node's binary address through a procedure called the *Ethernet Nomenclature System* (ENS) that is central to the working of the OEthernet framework. Note that the ENS is not required in MPLS-TP – all one needs is to use OEthernet addresses for LSRs and binary values of the addresses are now embedded in MPLS-TP LSPs.

To compute the route from the source to the destination, the source node uses the procedure described next. The source node examines both binary addresses (of itself and the destination). This is done by aligning the two addresses MSB onwards. All the leading *common* bits (if any) are then discarded. What remains are the source and destination remnant binary strings. We have now isolated the lowest common ancestor for the source and the destination nodes. The next step is to do a 1's complement on the source remnant string – this enables us to obtain a string that would guide a frame *from* the source to the lowest common ancestor. Recall that the original source binary address enabled a frame to be guided from the preset reference node to the node under consideration – and we now want to go in the opposite direction, *i.e.* towards the reference node. But instead of going all the way, we desire to stop at the lowest common ancestor. The next step is to flip the LSB of the source remnant string. The LSB of the source remnant string is used to indicate to the 1×2 switch at the lowest common ancestor as to what it should do with the incoming frame. The source remnant string with its 1's complemented bits and the last bit further complemented is now conjoined to the destination remnant string to create a *route-binary-string* as shown in Fig. 1.

In the case of OEthernet without MPLS-TP, we make use of the IEEE 802.1Qay but with minor differences resulting in significant performance improvements. Four key features distinguish the IEEE 802.1Qay from LAN/switched **Ethernet:** (1) Absence of MAC learning. (2) Turning off of Spanning Tree Protocol (STP). (3) The

customer frames with or without VLAN tags are mapped to service provider tag (STAG and then the ITAG) – which is further mapped to the BTAG and BMAC, thereby allowing encapsulation of the entire frame in a service provider frame and (4) Frames are forwarded based on a service provider BTAG in conjunction with a BMAC address. In the OEthernet case [5], we adopt this methodology (of turning OFF STP, disabling MAC learning) and allowing the use of multiple stacked VLAN tags in an Ethernet frame as our basic protocol data unit (PDU) for OEthernet. Further, five kinds of tags are proprietarily defined as follows: Source Address-Route TAGs or S-ARTAGs, Route ARTAGs or R-ARTAGs, Granularity Tags or GTAGs, Type Tags or TTAGs and Window-TAGs or WTAGs [15]. The S-ARTAG is a series of VLAN tags stacked together and contains information pertaining to the address of the node (*i.e.* the route from the preset reference node to the node). A single S-ARTAG can have 12 entries as its VID and hence can support a binary tree of diameter 12. If however, the binary tree has a larger diameter, then multiple S-ARTAGs can be stacked together. The R-ARTAG is a series of VLAN tags that carries route information – a binary string indicating the route from the source node to the destination node. The R-ARTAG is the most important tag for forwarding the frame. Each 1×2 switch examines just 1 bit in the R-ARTAG (based on the method below) and forwards the frame to one of its output ports. The frame format for the OEthernet frame is shown in Fig. 2.

From the perspective of MPLS-TP, our approach is to induct binary addresses as labels. An MPLS-TP LSP is now defined by its route from a source node to the destination node. Labels can be stacked together similar to MPLS-TP with each label depicting a path from the corresponding intermediate source node to an intermediate destination node. Shown in the Fig. 3 is a set of MPLS-TP LSPs that are stacked to provide an integrated LSP from the source to the destination. Shown in Fig. 4 is corresponding label stack that provides for the LSPs.

Working: The creation of the MPLS-TP LSPs using OEthernet concepts is explained in Section IV. We now discuss how to use the created MPLS-TP LSPs in a network. At an intermediate node, the topmost label is examined. The 20-bit address is analyzed as follows: the first 4 bits are called *pointers* – they indicate which bit amongst the 16-remaining bits in the shim, should the switch begin to work on. If the value of the first 4-bits is 1111, then the label is popped; the next label would have the first 4-bits set to 0000 as it is "fresh" implying that is has yet not been considered upon. The MPLS-TP LSR would then consider the next $\log_2 N$ bits, for an N-port LSR. The LSR would also update the first 4-bits to mark the $(\log_2 N + 1)^{th}$ bit, so that that next switch can begin consideration of the binary label. The last label in the stack is pushed into the stack after "last-label-preprocessing". As part of last-label-preprocessing, the first 4-bits of the bottommost label are changed from 0000 to a value that determines the number port count of the last LSR – the one that is the destination node. This value $(15 - \log_2 N)$ will tell the LSR that it is the destination node, and to send the packet to the requisite destination port.

Switching: At an LSR, there is no requirement for a lookup table. All that the LSR does, is to receive the packet correctly (usually done through cyclic redundancy check at layer-2) and then isolate the correct log_2N bits corresponding to the LSR. Once these bits have been isolated, the LSR forwards the packets based on the value of the

$\log_2 N$ bits. There is no lookup required to compare the value of the label – thus saving energy and time, reducing latency and maintaining wire-speed operation.

Note: Label swapping in the MPLS sense is not supported – the MPLS packet is created at the source node with multiple labels. It is assumed that the source node has the global topology of LSPs. This assumption is valid when we consider that (1) the data-center is a relatively closed domain and (2) without the knowledge of global LSP topology, it is not possible to enable end-to-end QoS support at the transport layer.

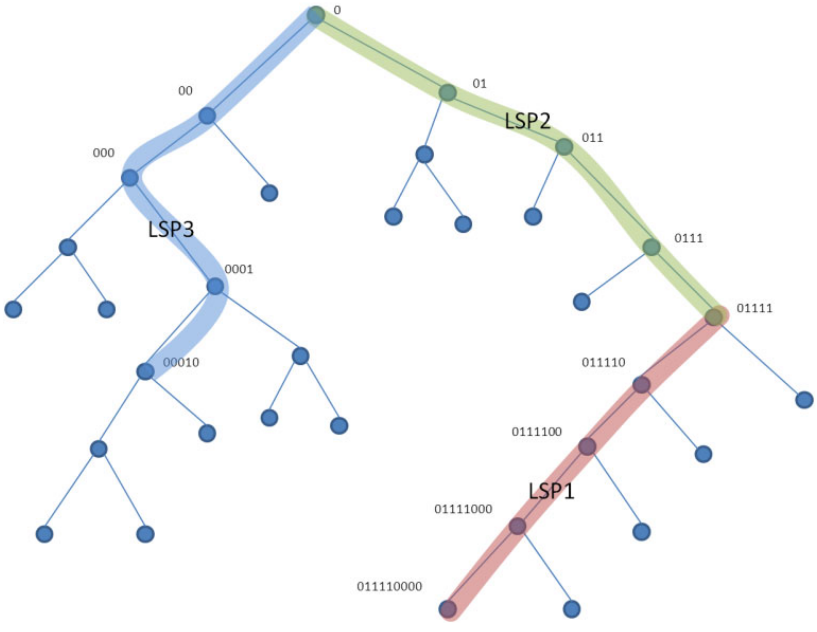


Fig. 3. LSPs in MPLS-TP using OEthernet addressing scheme

LSPs	S-ARTAG (Ingress)	S-ARTAG (Egress)	R-ARTAG
LSP1	011110000	01111	111
LSP2	01111	0	0001
LSP3	0	00010	1010

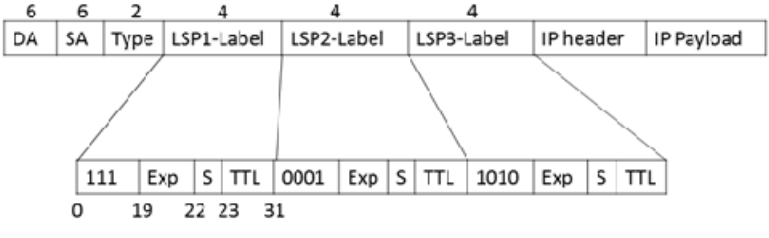


Fig. 4. Labels using MPLS-TP and OEthernet addressing scheme

4 Implementing MPLS-TP in the Data-Center

In this Section, we discuss the creation of MPLS-TP LSPs using OEthernet technology, as well as the implementation of MPLS-TP within the data-center. A generic data-center is shown in Fig. 5. Observe that the topology of the data-center resembles a star network – several servers and NAS units are interconnected to a gateway through the star topology. There are multiple methods to implement such a star network – as a hierarchical tree, or generic shuffle-exchange architecture [16]. We will present an approach to convert a given data-center interconnection model into a binary tree. Subsequently, we will show how to implement MPLS-TP within the data-center.

We first create a binary tree. For this purpose, the topology is discovered by the Network Management System (NMS) [17]. The gateway of the data-center that connects the data-center infrastructure to the rest of the Internet now becomes a default point of access.

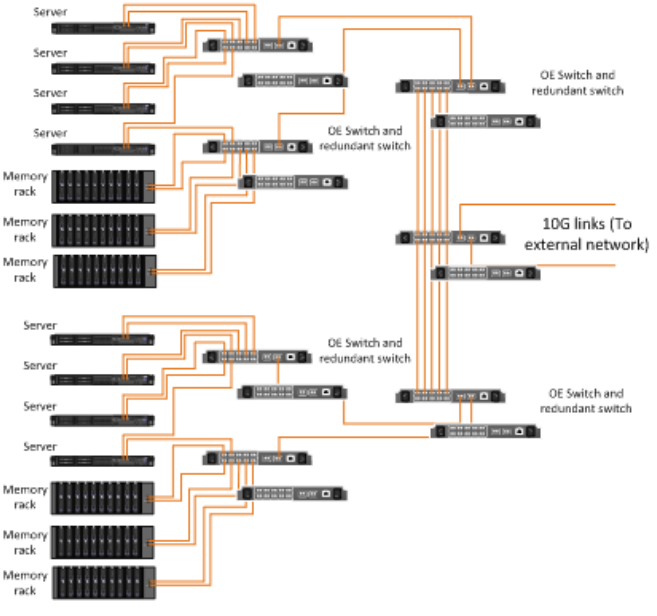


Fig. 5. Generic data-center using OEthernet switches

The following algorithm is run by the NMS in a breadth first search manner:

```
for ∀N
    if D(N)<1x2,
        replace N with N″
    end
end
```

The two variables D and N'' are defined as follows: $D(.)$ is an operator that denotes the degree of a node, while N'' converts any node with degree of connectivity greater than 1×2 to a set of nodes such that each node has a degree of connectivity 1×1 or 1×2. Upon converting every node in the tree to a binary-node, we apply the following algorithm to break cycles and create a tree.

```
is-cycle:=sort cycle()
     while is-cycle ≠ ∅
          break (cycle(i))
               is-cycle=is-cycle-i
               refresh is-cycle()
               increment (i)
     end
```

In the snippet above, we sort all the cycles in the graph according to their sizes in terms of number of edges and call this set as *is-cycle*. We disconnect the first cycle (the largest one), in the set *is-cycle*. We refresh the set *is-cycle()* to check if any cycles persist. If another cycle continues to persist despite the break of the previous cycle, we break the next largest cycle, and again check if any further cycles exist (after refreshing the set *is-cycle()*. The algorithm ensures that every cycle is broken with the disconnection of the minimal number of edges. To break a cycle, we choose the edge that causes *maximum damage* to the cycle. To do so, each edge in every cycle is given a weight. The weight corresponds to the number of cycles that the edge would break, if removed. In the break cycle statement, we hence break the edge with the highest weight.

The next step is to add binary addresses to a node. Giving the root an address of "0", we traverse to every leaf, appending a "0"bit to the existing address of the ancestor if the current node is right of its immediate ancestor, or append a "1" if the node is left of the immediate ancestor. Hence the two descendants of the root would have addresses of 00 and 01 respectively.

Subsequent to this, we create LSPs. For the creation of LSPs, each edge node has to perform binary-specific penultimate hop-popping operation – contrary to the generic MPLS-TP requirements. However, this is allowed as the Management End Points (MEPs) in the OEthernet case continue to be the source and destination node, thereby facilitating complete connectivity and fault tolerance operation.

At the edge nodes of the tree, i.e. the leaves we have to interface the edge LSRs with the servers and NASs. Likewise, at the root, we have to interface the data-center to the rest of the Internet. Since the binary addressing and routing are internal to the data-center, we need to facilitate a mechanism at both the gateway of the data-center as well as the leaves (edge LSRs) so that the global addresses can be mapped to local binary addresses. This translation between globally unique IP-addresses to locally relevant binary addresses is done by an edge logic structure that in the OEthernet parlance is called as the "Thin Ethernet Logical Layer" or TELL. The TELL is a table that contains mapping between a destination IP address (typically IPv6) to a corresponding LSP or R-ARTAG from the OEthernet perspective. In the case of the OEthernet network as shown in [15], the mapping is between the IP addresses/MAC

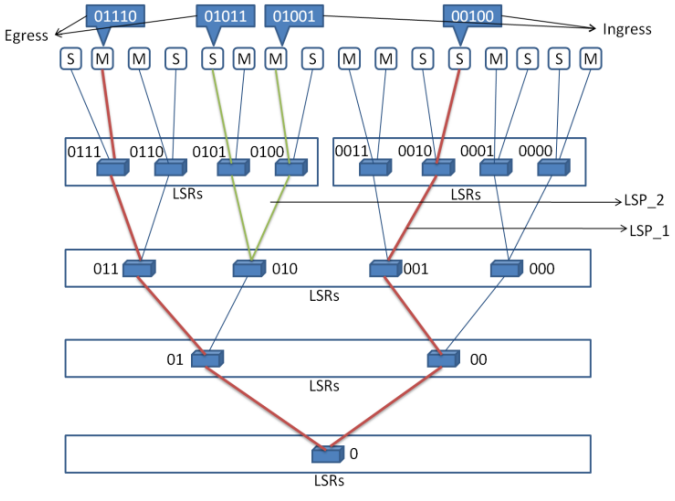
addresses or even HTTP URLS to S-ARTAGs, since the framework supports multi-layer communication. However, in the case of the MPLS-TP data-center, the mapping has to be only between the IP addresses and the S-ARTAGs. The edge LSRs create LSPs by examining the TELL table as shown in Fig. 6.

LSP Creation: We will consider how the LSPs are created inside an MPLS-TP supported data-center from the perspective of both communication from the edge nodes as well as from the core of the Internet (through the gateway).

At edge nodes: All the edge nodes are assumed to have at least one LSP to the gateway. An incoming packet whose prefix is beyond the scope of the data-center is encapsulated with labels that would enable it to reach the gateway (root) node. All such out-of-scope packets imply that the destination is outside the data-center.

IPv6	LSP ID	Label 1 (20 bit value)	Label 2 (20 bit value)	Label 3 (20 bit value)
2010:0db8:3c4d:0015:0000:0000:abcd:ef12	LSP1	0000110110 1000000000	0000011000 1000000000	0000111101 0110000000
2010:0db8:3c4d:0015:0000:0000:ad13:cd13	LSP2	0000100100 1000000000	0000011110 1000000000	0000100000 1000000000
2010:0db8:3c4d:0015:0000:0000:abc1:0011	LSP3	0000100010 1100000000	0000101100 0001100000	0000110010 1000100000

Fig. 6. An example of a populated TELL Table



LSP ID	Binary SRTAG	Binary R-ARTAG
LSP_1	00100	1100110
LSP_2	01001	001

Fig. 7. MPLS-TP data-center and label creation

At the gateway: The TELL table at the gateway implements the mapping between incoming IP requests and outgoing LSPs. While we assume at least one LSP to each leaf (from the gateway), this assumption is not always practical for very large sized networks. The gateway using global LSP information then selects a set of multiple stacked LSPs (and hence a stack of labels) that would guide the incoming packet to the requisite destination node (leaf). Multicasting is handled using the multicaster logic used for OEthernet in [5, 15]. For multicasting, the lowest common ancestor for all the multicast nodes creates LSPs to each destination and replicates packets to each such LSP. In a future work, we also consider the creation of a multicast LSP tree, though this requires intelligence at intermediate nodes for selective multiplication.

LSP Consolidation: As mentioned earlier, it is not always possible to have LSPs from the gateway to every leaf. Likewise, for inter-leaf communication, especially to support virtualization, virtual machine migration etc. it is not possible to have LSPs set up between every pair of leaves within the data-center. Hence, we use the concept of multiple LSPs within the data-center by using label stacking (see Fig. 3 for example).

5 Simulation Model and Results

We performed an extensive discrete event simulation (DES) to evaluate the performance of our proposed data-center interconnection mechanism using Omnipresent Ethernet encoded in MPLS-TP. In the model, we assume 70% of the leaves to be NAS and 30% to be servers (processors). All the leaves have Gigabit Ethernet interfaces. The number of leaves is varied from 1000 to 1 million. Traffic requests arrive at the gateway as service jobs and these are to be transported to the leaves or within the leaves (inter-leaf communication) or from the leaves to the gateway. LSPs are set up ahead in time for the major routes. There are 4-levels of QoS supported by the network. Each LSP also defines with it a granularity that can be implemented using a token bucket rate-limiter function. Our interest is to measure the performance of the MPLS-TP architected data-center using OEthernet concepts as compared to native data-centers using MPLS, MPLS-TP (standard) as well as IP routers. Requests arrive following a Poisson distribution and are characterized by a general holding time (since the data-center is a specialized part of the network – most requests are heavily granular, as opposed to regular arrivals that are exponentially distributed). Load is computed as the ratio of the total consumed bandwidth in the network, as opposed to the total bandwidth that the network can provide, resulting in a range of [0,1].

Shown in Fig. 8 is a viewgraph of latency versus load for a data-center of size 1000 nodes (leaves). For comparison, we measured the performance of data-centers with IP routers, conventional MPLS and MPLS-TP (without OEthernet technology) using the same traffic and the same topology. Observe the almost 3-orders of difference in the delay values between any of the other technologies and our proposed data-center architecture. The measurements are taken as average latency over all the source-destination pairs and at time-ensemble, (1000 runs at the same load value). The MPLS-TP architecture using OEthernet does not require any lookup table and the maintenance of a global LSP database further facilitates faster switching. As can be seen in Fig. 8,

the latency of our proposal is consistently better than all the other conventional technologies. This advantage is very important from the data-center perspective given that it is said that a 1-millisecond latency difference can cause a financial trading house over 100-million USD in a fiscal year.

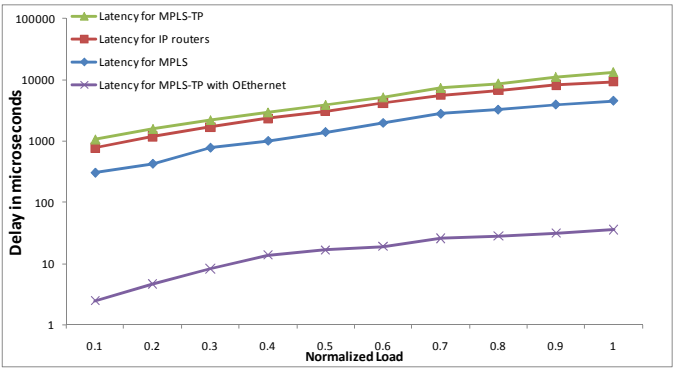


Fig. 8. Latency results for a 1000-node data-center

To demonstrate scale, the latency measurements are taken for a larger (1-million node) data-center. The measurements are consistent as can be seen in Fig. 9, whereby the latency difference between our proposal and existing technologies is easily 2-3 orders of magnitude. The superior performance of OEthernet when encapsulated within the MPLS-TP domain adds significant functionality flavor to MPLS-TP from the data-center perspective.

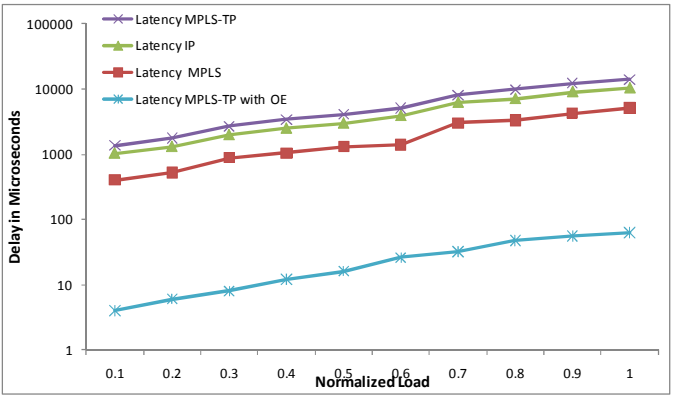


Fig. 9. Latency results for a 1000,000-node data-center

Shown in Fig. 10 is a viewgraph of energy consumption for an MPLS-TP network as compared to the energy consumption for our proposal. The MPLS-TP network requires more processing at each LSR. It can be concluded that the energy requirement is directly proportional to (1) the latency of the protocol – more the time spent at a node, more the energy consumed due to processing. and (2) lookup table size – larger

the table, more the energy required. Base values for energy consumption are assumed as shown in [5, 15]. On an average, there is a 72% energy saving using our proposal as opposed to a generic MPLS-TP scheme. It should also be noted that the energy consumption for IP-routers and MPLS LSRs is significantly more than that when we use MPLS-TP and hence not shown in the viewgraph.

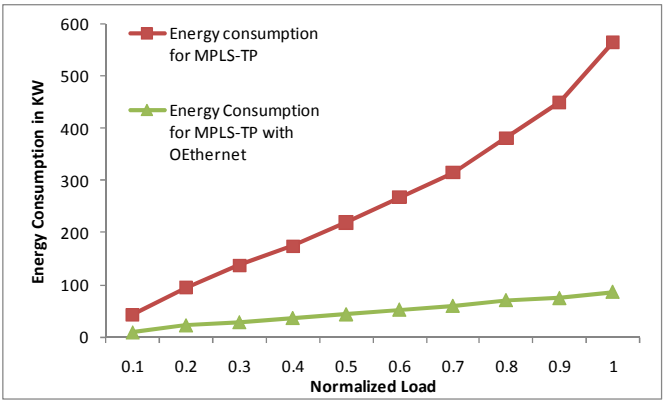


Fig. 10. Energy Efficiency comparison between MPLS

6 Conclusion

The fast proliferation of data-center technology has implied a need for a scalable, acceptable and economical protocol for interconnection between servers, NAS and switches. We propose the use of MPLS-TP to architect the data-center. However, instead of using native MPLS-TP, we propose the use of our earlier proposed Omnipresent Ethernet technology as an enabler for faster switching, lower energy consumption and better scalability within the data-center. The OEthernet technology, through the use of binary and source routing when plugged into MPLS-TP creates a very fast, efficiency and lower-energy consuming network – especially suited for the data-center. These performance results justify the use of OEthernet technology within an architected MPLS-TP data-center.

References

1. Gumaste, A., Antony, T.: Data Center Networking and Cloud Computing - A Networking Overview, Embedded Technology Brief (2009)
2. Shpiner, A., Keslassy, I.: A switch-based approach to throughput collapse and starvation in data centers. In: 18th International Workshop on Quality of Service (IWQoS), June 16-18 (2010)
3. Farrington, N., Rubow, E., Vahdat, A.: Data Center Switch Architecture in the Age of Merchant Silicon. In: 17th IEEE Symposium on High Performance Interconnects, August 25-27 (2009)

4. Ibanez, G., Carral, J.A., Garcia-Martinez, A., Arco, J.M., Rivera, D., Azcorra, A.: Fast Path Ethernet Switching - On-demand, efficient transparent bridges for data center and campus networks. In: IEEE Workshop on Local and Metropolitan Area Networks (LANMAN), May 5-7 (2010)
5. Gumaste, A., Mehta, S., Arora, I., Goyal, P., Rana, S., Ghani, N.: Omnipresent Ethernet—Technology Choices for Future End-to-End Networking. Journal of Lightwave Technology 28(8) (April 2010)
6. Niven-Jenkins, B., Brungard, D., Betts, M., Sprecher, N., Ueno, S.: Requirements of an MPLS Transport Profile. IETF RFC 5654 (2009)
7. Bocci, M., Bryant, S., Frost, D., Levrau, L., Berger, L.: A Framework for MPLS in Transport Networks. draft-ietf-mpls-tp-framework-12 (2010)
8. Busi, I., Allan, D.: Operations, Administration and Maintenance Framework for MPLS-TP based Transport Networks. draft-ietf-mpls-tp-oam-framework-08, September 17 (2010)
9. Koike, Y., Paul, M.: MPLS-TP OAM Maintenance Points. draft-koike-ietf-mpls-tp-oam-maintenance-points-01, March 8 (2010)
10. Sprecher, N., Farrel, A.: Multiprotocol Label Switching Transport Profile Survivability Framework. draft-ietf-mpls-tp-survive-fwk-06, June 20 (2010)
11. Bocci, M., Swallow, G.: MPLS-TP Identifiers. draft-ietf-mpls-tp-identifiers-0, March 8 (2010)
12. Takacs, A., Fedyk, D., He, J.: OAM Configuration Framework. draft-ietf-ccamp-oam-onfiguration-fwk[A1], January 28 (2010)
13. Zhang, F., Wu, B., Dai, X.: LDP Extensions for MPLS-TP PW OAM configuration. draft-zhang-mpls-tp-pw-oam-config-00, October 15 (2009)
14. Sprecher, N., Bellagamba, E., Weingarten, Y.: OAM Analysis. draft-ietf-mpls-tp-oam-analysis, July 04 (2010)
15. Gumaste, A., Mehta, S., Vaishampayan, R., Ghani, N.: Demonstration of Omnipresent Ethernet - A Novel Metro End-to-End Communication System Using Binary + Source Routing and Carrier Ethernet. Journal of Lightwave Technology 28(4), 596–607 (2010)
16. Mysore, R.N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: PortLand - A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In: Proceedings of the ACM SIGCOMM Conference, Barcelona, Spain (August 2009)
17. Gumaste, A.: Deciphering omnipresent ethernet: An all ethernet communication system - the control plane. In: 12th International Conference on Transparent Optical Networks (ICTON), June 27-July 1 (2010)