Traffic Engineering in Metro Ethernet

Padmaraj M. V. Nair mpadmara@lyle.smu.edu

Suku V. S. Nair nair@lyle.smu.edu

F. Marco Marchetti marco@lyle.smu.edu
HACNet Lab, Southern Methodist University,
Dallas, TX, USA

Girish Chiruvolu
Maher Ali
Alcatel-Lucent
Plano TX, USA

Abstract

Traffic engineering is one of the major issues that has to be addressed in Metro Ethernet networks for quality of service and efficient resource utilization. This paper aims at understanding the relevant issues and outlines novel algorithms for multipoint traffic engineering in Metro Ethernet. We present an algorithmic solution for traffic engineering in Metro Ethernet using optimal multiple spanning trees. This iterative approach distributes traffic across the network uniformly without overloading network resources. We also introduce a new traffic specification model for Metro Ethernet, which is a hybrid of two widely used traffic specification models, the pipe and hose models.

Keywords: Metro Ethernet, Traffic Engineering, Multiple Spanning Trees

1. INTRODUCTION

Ethernet has evolved over the past decade from a simple CSMA/CD shared medium access protocol to a full-duplex, switched network. Ethernet dominates the current LAN realizations and it has been estimated that more than 90% of IP traffic originates from an Ethernet LAN. Inherent simplicity of Ethernet brings nice properties like cost effectiveness, rapid provisioning on demand, ease of interworking, simple packet based technology and ubiquitous adoption [6]. It promises relatively inexpensive high-speed access, which can then be combined with new networking services from the enterprise domain. With the latest enhancements, it has turned out to be a feasible technology even beyond the LAN. Switching, Fast Ethernet and Gigabit Ethernet have brought more bandwidth to the technology.

However Ethernet lacks end-to-end QoS guarantees, protection mechanisms and service performance monitoring [6]. Present Ethernet connectivity is largely restricted to the fast-growing but relatively small niche of the Internet access. In order to enter the metro domain, Ethernet must prove that it is a carrier-grade technology. Deploying Ethernet in the MAN will require many different upgrades. First, it requires the port-folio of Ethernet physical layers to be extended for the local loop and for interoffice connection within the MAN. Second, there is a need to upgrade

the Ethernet switching layer. It should be reliable and secure enough to handle mission critical corporate data.

Current Ethernet protocol relies on the IEEE spanning tree protocol (STP) [2], which provides a loop-free connectivity across various network nodes. But this protocol has slow re-convergence time in the case of a failure that typically is 50 seconds. Time critical applications on Ethernet can only afford a reconfiguration time less than few tens of milliseconds. Further, STP uses a single spanning tree to carry the entire network traffic, resulting in congestion and resource underutilization. To improve upon reconstruction time 802.1W [3] standardized its Rapid Spanning Tree Protocol (RSTP). In both RSTP and STP, after a re-convergence the MAC address might get associated with an altogether different switch port and makes it difficult to predetermine the path between a given pair of nodes after a spanning re-convergence.
The Multiple Spanning Tree Protocol (MSTP) [4] defined in IEEE 802.1s standard provides alternate paths between two nodes within an administrative region such that basic traffic engineering can be enabled. This standard provides guidelines for better resource utilization, localization of failures, and faster recovery. MSTP uses multiple spanning trees and VLANs [1] are mapped onto these trees.

Customers are demanding end-to-end Ethernet solutions in which geographically distant LANs are connected as if they were one single LAN. This customer virtual LAN(C-VLAN) provides full connectivity between customer sites (LANs), and allows a station in one LAN to engage in unicast, multicast, and broad-cast communication with any other station belonging to the C-VLAN [14]. In this paper we present an algorithmic approach for traffic engineering based on construction of multiple spanning trees in the metro do-main using customer traffic demands and given network topology.

Traffic specification is another important factor in traffic engineering. Current Metro Ethernet uses pipe and hose models. In our multiple spanning tree algorithm we use the pipe model. The pipe model provides the most efficient interface for the provider in terms of bandwidth allocation. However, it suffers from being complex and it requires pair-wise traffic information from the customer. The hose model, on the other hand, does not require knowledge of pair-wise traffic, but only aggregates. In addition, it is easy to specify by the customer. However, the lack of exact traffic distributions makes it least efficient in terms of bandwidth allocation. To take advantage of both these methods we introduce a hybrid model (Augmented Hose) in the future directions. The idea is to enhance our multiple spanning tree construction method with this new traffic specification model in the future.

The rest of the paper is organized as follows. Next section provides an overview of the IEEE Ethernet standards and the kind of network in consideration is described in section 3. Section 4 discusses the related work in this area. Section 5 proposes an algorithmic approach to multiple spanning tree construction. In section 6 we provide examples showing the performance of our approach. Future directions are discussed in section 7. Finally, section 8 concludes with a summary of our work.

2. OVERVIEW OF THE ETHERNET

This section elaborates on some of the relevant IEEE standards for Ethernet.

2.1 IEEE Spanning Tree

Ethernet traditionally relies on spanning tree defined in IEEE 802.1D, which inherently provides loop-free communication among all the nodes. Basically the construction of the spanning tree through the exchange of node ids and link costs (a.k.a. Port vectors). The algorithm terminates upon the discovery of the root node with the lowest id number and the lowest cost paths between every node and the root node. Each port of a switch assumes one of the three roles, namely, a)

Root port b) Designated port c) Disabled port. The port that leads to the Root Bridge with least path cost is the Root port for that non-root bridge. All the other ports (links) are either designated ports which forward frames on behalf of a LAN segment or blocked ports. Blocked ports lead to the underutilization of the links. The blocked ports/links serve as backups in the event of a change in topology due to link/node failure. However, the spanning tree reconstruction after a link failure has slow convergence as the default timers associated with STP are based on worst-case behavior of the network in terms of size (diameter) and response. Typically the worst-case reconvergence of the STP for a stable spanning tree after an event of link failure is around 50 seconds which is not acceptable for real time mission critical applications.

2.2 The Rapid Spanning Tree

The blocking port role in STP is split into two in RSTP, backup and alternate port roles. These ports do not forward user data. If the port is considered for immediate forwarding state in the event of root port failure then it is an alternate port. A port is backup port if it is for immediate forwarding in the case of a designated port failure. This is due to the fact that the alternate and the backup ports also do the learning even before they are promoted to the forwarding state. This significantly reduces the convergence time of RSTP, but still in the order of few seconds (~1-2 seconds) as the flush message (broadcast by the switch that gains in the subtree, rooted by itself, topology) takes time to propagate across the nodes as a result of (logical) topology change due to change in port status. Moreover, RSTP suffers from the same drawback as STP: both standards use a single tree to carry all the network traffic causing resource overloading and resource under utilization.

2.3 The Multiple Spanning Trees

IEEE 802.1s deals with defining Multiple Spanning Trees (MSTs) wherein each VLAN can be uniquely mapped onto a single spanning tree among the set of spanning trees. One can have a spanning tree per VLAN or multiple VLANs can be mapped onto a single spanning tree. Collection of MSTs in a VLAN-aware network is defined as a Spanning Forest (SF). The SF allows the utilization of all links that would otherwise be idled by the standard ST and thereby eliminating the wastage of bandwidth. However, the complexity of management increases with the number of spanning trees, switches must maintain port state information for each spanning tree of the SF, and the set of VLANs mapped on to them. The MSTP also allows a set of regions to be defined whose union spans the entire network. Within a region one or more MSTs can be defined. MSTP ensures that the frames of a given VLAN are assigned to only one of the spanning tree instances within a region. A common spanning tree (CST) spans across all the regions. The key advantage of such segmentation of the network into regions is the localization of any failure while having minimum/no impact on the non-local VLAN traffic. However, in order to have alternate paths between any pair of nodes with different VLAN segments attached, multiple regions have to be defined. This is due to the fact that each of the regions binds a given VLAN traffic with a unique instance of spanning tree, which in turn translates into a single but unique path. Moreover, spanning trees constructed in a region essentially follow RSTP and thus inherently carry the weakness of RSTP.

3. SYSTEM DESCRIPTION

Several architectural options have been proposed and deployed to carry Ethernet frames across metro area.

- Extending the native Ethernet protocol which is standardized under 802.1 committee.
- Using MPLS as the transport technology.
- Using SONET/SDH as the transport technology via General Frame via Generic Framing Procedure (GFP) encapsulation and Link Capacity Adjustment Scheme (LCAS) rate adaptation.
- Using generalized MPLS (GMPLS) to control Ethernet switches in the metro network.

Our research is based on the first architecture in conformance with the 802.1 standards. Fig 1 depicts the typical architecture for Metro Ethernet [14]. The metro domain consists of core switches/nodes and edge nodes (ingress/egress nodes). The goal is to transport end user Ethernet frames across the metro domain in a transparent manner. Since the Ethernet MAC addresses are flat, ingress nodes encapsulate the user frames with the proposed extension fields. The edge node identifiers are inserted appropriately such that the core nodes learn only about the edge nodes of the metro domain and not the end user MAC addresses. The edge nodes in turn learn about the MAC addresses of the end user frames belonging to the VLAN attached to them [10]. Another approach would be to use the labels that uniquely identify the edge nodes. Thus, encapsulation of end user frames can address the MAC address table explosion issue in the Metro domain.

The edge nodes of a metro region are access points (AP). Customer virtual LANs (C-VLAN) connect to different APs. A C-VLAN route is basically a tree that spans all locations of a given customer. A C-VLAN must use exactly one spanning tree for communication. A 12-bit VLAN identifier (VID), inserted into the end-user Ethernet frames, identifies each of the VLANs. The tagged frames belonging to a given VLAN will be forwarded across the network only to the end-hosts of the VLAN. This provides an implicit VLAN segregation and optimal network resource utilization. The traffic requirement of a customer is represented as a demand matrix (or access point matrix), which shows the traffic between AP pairs of a particular C-VLAN.

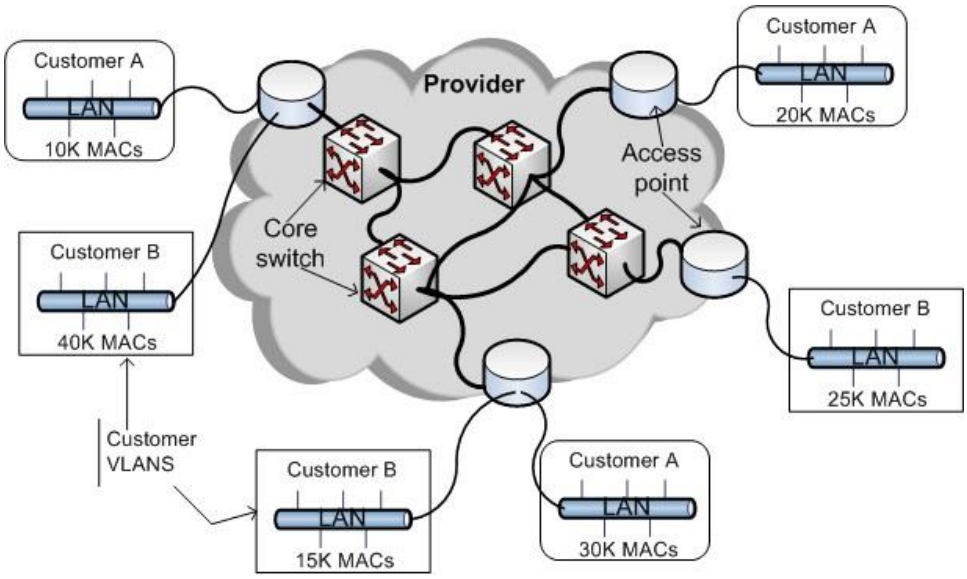

FIGURE 1: Metro Network.

4. RELATED WORK

Traffic engineering of Ethernet using spanning tree is a widely researched topic because of performance issues. QoS-aware Multiple Spanning Tree Mechanism [9] was proposed to address problems related to IEEE 802.1 standards and its extensions to the Spanning Tree protocol. They have proposed a simple and highly effective enhancement to the MST protocol to achieve high degree of QoS by keeping in perspective the different characteristics of the various traffic types. They use the traffic priorities defined in 802.1d standard for mapping VLANs to STs. For example the STs for best effort traffic is built based on the inter-face speed and end-to-end delay is the most important factor for building STs for multimedia traffic.

A fault tolerant multiple spanning tree protocol is proposed in Viking [12]. The Viking system provides at least two switching paths between any pair of end-nodes in two different STs, primary and backup switching paths. The path selection is based on cost assigned to each link. These

links are combined to form STs using path aggregation algorithm. The aggregation algorithm starts with an empty spanning tree set and select each path in the list to add to a spanning tree which does not form a loop. If there is no such tree then form a new spanning tree. One issue with this loop avoiding path selection procedure is that avoiding loops might leave out some paths which could have resulted in a better optimal tree. Viking relies on per-VLAN-spanning tree implementation of Cisco where there is a separate spanning tree running on every switch for every VLAN. This has the limitation on the number of VLANs the metro Ethernet can support due to the maximum VLAN tag size and the number of spanning trees a switch can support.

The article in [11] looks into the basic problems of traffic engineering in Metro Ethernet, such as load balancing, QoS-based protection, label-space management, evolving trends in traffic management at standard bodies and their implications. This proposed a grouping scheme that extends the current label space in the provider domain and allows for a large number of VLANs to be provisioned efficiently. Further, the issues of load balancing, multiple spanning trees, and interaction between grouping and bandwidth provisioning are discussed. It also addressed differentiated survivability in next-generation Ethernet and provided a novel scheme based on multiple spanning trees. The traffic engineering method we are proposing in this paper can use the C-VLAN grouping scheme described in this article to save VLAN tag space.

DiffPause [5] is a scheme for congestion control that is highly scalable, robust, and compatible with the assured forwarding of the differentiated services model for high-speed Metro Ethernet. The DiffPause introduces an Early Warning Threshold such that the upstream nodes can reduce the outgoing rate and throttle the aggressive traffic aggregates, thereby providing fair bandwidth allocation. This scheme takes advantage of per link-based back-pressure mechanism without any bandwidth reservation and is independent of the number of ongoing individual sessions, thus leading to high scalability.

In [13] we introduced a multiple spanning tree region construction algorithm which provides basic traffic engineering of Ethernet in the Enterprise domain. The proposed method supports dynamic nature of VLANs where the VLAN nodes can be added, moved, or removed without much effort. This method provides enough resources in the region to support protection from at least single failures. These multiple spanning tree regions are carefully engineered to provide better convergence time, reusability of VLAN tags, protection from failures, and optimal broadcast domain size.

A distributed scheme for fast restoration of Metro Ethernets, a fast failure recovery spanning tree scheme is proposed in [7]. This method restores lost facilities within 50 milliseconds irrespective of the network size.

5. TRAFFIC ENGINEERING USING OPTIMAL MULTIPLE SPANNING TREES

This optimal multiple spanning tree (OMST) method primarily uses customer traffic demands and network capacity to generate spanning trees. The approach complies with IEEE 802.1s MSTP and addresses six major components in constructing spanning trees. These are:

- How to assign weights to links in the network
- Root node selection for a spanning tree
- Parameters determining weight of the spanning tree
- How to map customers on to the tree
- Effectively distributing traffic across the network
- Avoid resource overloading

A C-VLAN is mapped to a spanning tree and frames forwarded to a tree can be identified by the VLAN ID. The C-VLAN grouping method suggested in [11] can used to save VLAN tags and limit the number of trees. Then the demand matrix for a group of C-VLANs will be the sum of all demand matrices. Thus, when spanning trees are constructed using our method, each customer

group is mapped to a single spanning tree. However, in this proposal we are not considering the grouping method and spanning trees are constructed for individual C-VLANs. The spanning trees are constructed such that the traffic is well distributed and no link or node is overloaded. Links for the spanning tree needs to be selected in such a way that it can handle the traffic demand generated by end nodes.

Dijkstra's algorithm can be a simple solution to this problem: build a spanning tree for each C-VLAN and then update weight on the links to reflect the new traffic. Since a customer is not using all the APs, we may not get the best path between C-VLAN's APs for the given traffic, even though the spanning tree is minimum weight. Further, the order in which links are selected, location of the root node and the order of spanning tree construction can affect the final solution. So our proposed solution considers all these requirements to build spanning trees. Though our approach might not deal with sudden load changes in the network, constant monitoring and simple reconfigurations can increase performance.

5.1 Dynamic Link Weight

Link weight is a function of the given link capacity and other parameters such as delay. To simplify the problem we are assuming that capacity and delay are the only parameters used in finding the link weights. However, this linear function can be extended with more parameters in the future. The delay itself can be divided into propagation delay and queuing delay. Queuing delay would increase as the traffic increases. Here we are using propagation delay which depends on the distance.

- Link weight is directly proportional to Delay.
- Link weight is inversely proportional to Capacity.

So a simple function is,

$W_{ij} = D_{ij}/C_{ij}$

Where W_{ij} is weight of the link between nodes *i & j*, C_{ij} is capacity of the link between node *i & j*, and D_{ij} is the delay between nodes *i & j*. Since we are trying to find the shortest possible paths (tree) between APs and equally utilize available resources (such as capacity), delay and link capacity are valid parameters. When the algorithm progresses available capacity on links will change, thus the link weight would also change. Hence adding more parameters to the function might need updating more variables during the course of algorithm execution.

5.2 Algorithm

The algorithm first sorts C-VLANs based on their traffic demand and builds a spanning tree for each customer starting from the customer with the highest demand. We use the highest-demand-first technique because inserting lower traffic demand into the network can be done without loss of much performance. Construction of each spanning tree is followed by calculating its aggregated weight. Aggregated weight is calculated using the formula given below, which is similar to the method suggested in [8].

$ST_{AW} = \sum W_{i,j} \times T_{i,j}$.

Where, ST_{AW} = Aggregated weight of the spanning tree. $T_{i,j}$ = Traffic flowing through the link connecting nodes *i* and *j*. If this link is used in more than one path (that is more than one pair of access points are using the link) then $T_{i,j}$ is the sum of traffic on the link. $W_{i,j}$ = Weight of the link between nodes *i* & *j*.

Once the tree is constructed for a C-VLAN, the link weights on the graph are updated to reflect the new traffic. This process is continued for each C-VLAN in descending order of their demands. The solution steps are described in Fig. 2 and a detailed description follows it.

```
1. Sort C-VLANs in descending order, based on traffic demand.
2. For each C-VLAN in the sorted list do {
   a) Select an AP as the root node from the list of APs that this C-VLAN is using.
   b) Create the parent tree with the root.
   c) Calculate aggregate weight of the parent tree.
   d) For all APs the C-VLAN is using {
        i. Find the path with minimum weight between each pair of APs in the graph
       ii. Sort these paths in descending order of their aggregate weights
      iii. For each path do {
             1) Select the unprocessed path with highest weight and map on to the tree
             2) If (loops exist), then break loops
             }
       iv. Calculate the aggregated weight of the tree and we now have the final tree.
        v. Update the weights of links in master graph.
    }
 }
```

FIGURE 2: Algorithm for constructing spanning trees

Sort the C-VLANs based on the demand and set the index to highest demand C-VLAN: C-VLANs are sorted in descending order of their demand. The aim is to construct spanning trees for C-VLANs in the order of their traffic demands. This way more demand would be satisfied using best optimal trees and the traffic would be well distributed on the network resulting in better performance.

Take next C-VLAN with the AP matrix and find a root node for the tree: The next selected C-VLAN will have the next highest demand requirement. After selecting a C-VLAN an access point is selected as the root for the tree. This can be done in three different ways:

- Select an AP from the list of APs the C-VLAN is using, which carries the lowest traffic.
- Select an AP from the list of APs the C-VLAN is using, which carries the highest traffic.
- Select a random AP.

We prefer the first option since there is a higher possibility that more traffic will flow through the root node of a spanning tree.

Create a spanning tree rooted at the root node: Create a minimum spanning tree rooted at the above selected node from the given graph. Let us call this the parent tree.

Calculate the aggregated weight of the tree: Aggregated weight of the tree is calculated using the previously defined function which considers both link weight and the traffic flowing on the link.

Find the shortest path between all APs of the C-VLAN and sort them in descending order of weights: Find the shortest path between each pair of APs (only for APs the C-VLAN is using) which satisfies the demand and assign weights to the path. A simple method like Dijkstra's algorithm can be used. Assign weights to each link in the path using the formula,

$W_{i,j} \times T_{i,j}$.

Where $T_{i,j}$ is the traffic flowing through the link and $W_{i,j}$ is the weight of the link. These paths represent the best routes the C-VLAN can have based on the traffic demand. Now list these paths in the descending order of their weights. For example:

AP1 ← → AP2, AP3 ← → AP1, AP2 ← → AP4

Take next path & map to the tree: Selection of the next path from the list can be done in two different ways:

- Select the path with the lowest weight first.
- Select the path with the highest weight first. This is a better choice because shorter paths would be applied towards the end of mapping.

Once we select a path, add its links to the tree. Since adding new links to the tree generate loops, the next task is to break loops.

Break loops: This step involves detecting the loop which is a very complex problem when looking for multiple loops. So to make this method simple, after adding each link from the above given path, algorithm breaks the loop. Adding one link to the spanning tree means a loop and this loop can be easily located. Finding multiple loops and breaking them in an optimal way is part of the future work. Now breaking loop can be done in two different ways. A simple approach would be to delete one of the links other than the newly added link. However this may not result in an optimal solution because some links may cause an increase in weight of the tree killing the whole purpose of mapping paths onto the tree. An optimal solution is described in Fig.3.

1) Delete the link which contributes more weight to the tree.
  a) For the loop {
    i) Determine the links which formed the loop.
    ii) For each link check {
      (1) If this link is removed, do other links in the loop have enough capacity to handle the excess traffic? Then this link is a candidate, otherwise this link cannot be removed.
      }
    iii) Once we get a list of candidate links for removal, remove the link which contributes the highest weight to the tree. In other words, after breaking each link in the loop we will get a different tree. Now keep the tree with the lowest weight.
}

FIGURE 3: Algorithm for breaking loops.

Update weights on the graph: Once all the paths mapped on to the tree, the C-VLAN gets the final tree. Next step is to update weights on the links of the graph based on the traffic through the links of the tree. Now the capacity of each link in the graph will change to,

$C_{ij} = C_{ij} – T_{ij}$

Where, C_{ij} is the capacity of link (i, j) in the graph and T_{ij} is the traffic of link (i, j) in the tree. This update will occur only for the links shared between the tree and graph. Once the capacity is updated, we need to recalculate the weight on each link in the graph using the weight formula defined in subsection 5.1.

This algorithm completes with optimal multiple spanning trees for C-VLANs. Now the spanning tree assigned to each C-VLAN can be further tuned by blocking ports which are leading to access points that are not used by this C-VLAN. This avoids traffic leakage and thus prevents unnecessary wastage of resources.

6. PERFORMANCE ANALYSIS

Performance evaluation of this algorithm is based on complexity analysis and simulation studies. This section describes results of our experiments.

6.1 Complexity Analysis

Since Dijkstra's algorithm is used in OMST, the complexity analysis is basically based on the complexity of Dijkstra. The complexity of sorting C-VLANs is simply the complexity of sorting algorithm we use, which is *O(n logn)* for n C-VLANs. The part which generates the initial spanning tree, the parent tree, is the first complex operation on the graph. The source based spanning tree algorithm on graph with V vertices and E edge has complexity *O(E + V log V)* using

the Fibonacci heap. Calculating aggregate weight of the spanning tree is simple and involves a smaller part of the graph and hence does not contribute much to the complexity.

Finding shortest paths between access points of a C-VLAN in the original graph is the next complex part of the algorithm. Since mapping of paths onto a spanning tree is done one link at a time, this process is less complex compared to finding shortest paths. Shortest paths from one node to all other nodes in the graph can be derived using Dijkstra's method. This algorithm has the same complexity as the spanning tree construction method, which is $O(E + V \log V)$ using Fibonacci heap. Though we need only paths between access points, the algorithm may end up finding shortest paths to all nodes in the graph. Since we need to find the shortest path from each access point to all other access points, the total running time is $O((A-1)(E + V \log V))$, which is $O(A(E + V \log V))$. The number of access points used by the client is represented using the variable A. So the overall complexity of this algorithm to generate a spanning tree can be represented using $O(A (E + V \log V))$.

6.2 Experimental Results

Inputs to the simulation are the network topology in matrix format and traffic demands for each C-VLAN. Network topologies and traffic demands are randomly generated based on probability functions. For every topology we used, the bandwidth of links was set to 100 Mbps. Propagation delay is randomly selected between 0.2 and 1.0 milliseconds. Output of the simulation is in the form of spanning trees. Then we analyze the performance of these trees with trees constructed using Dijkstra's method.

Fig. 4 shows the performance of a 22 node network. Two parameters are selected to show the performance, aggregated weight (Ag_W) of spanning tree and average weighted length (Av_WL) of paths within the tree. To calculate average weighted length, we first find the weighted length of all paths between AP pairs for this particular C-VLAN. The average of these values gives us the final average weighted length. This parameter gives us a clear idea about the delay involved and broadcast domain size. These parameters are compared with spanning trees generated using simple tree construction methods. The resulting network has 13 access points and a performance matrix is collected for 5 C-VLANs. Fig. 5 shows the performance matrix for different networks. Parameters are selected as the average for each network type.

Fig. 6 shows the traffic distribution on a heavily connected 12 node (40 links) network with 100 Mbps bandwidth based on the following approaches:

1. Single spanning tree for connecting all the nodes in the network. This spanning tree will be used by all C-VLANs in the network.
2. MSTs for C-VLANs. However the link capacities are not updated after constructing trees, resulting in almost similar trees.
3. MSTs are constructed and link capacities are updated after building a tree for the C-VLAN. This is very much similar to our approach, except that finding shortest paths and mapping onto the tree is not applied. Even though we haven't seen this kind of an approach in any literature for Metro Ethernet, we used this to prove the efficiency of our technique.
4. The optimal multiple spanning tree approach.

From the graph it is clear that algorithm 1 and 2 caused some links to overload and used less than 30% of links. Even though we used multiple spanning trees in the second approach, the result is close to the single spanning tree. Since traffic is not reflected in the graph after building spanning trees, choosing different root nodes for each C-VLAN will not make much difference to the structure of spanning trees. However, the third technique and our approach are very much similar. In our algorithm the maximum traffic on a link was less than 50% of link capacity, and less than 10% of links with no traffic. In the third technique (MST with traffic updates) 25% of links were not used and up to 80% capacity of some links was used. In conclusion our approach

can indeed increase the performance of spanning trees and keep traffic distributed efficiently in a metro domain.

C-VLANs	% of decrease in	
	Ag_W	Av_WL
1	32	20
2	46	39
3	32	29
4	39	32
5	38	39
Average	**37.4**	**31.8**

FIGURE 4: Performance of a 22 node network.

# of nodes	# of access points	# of C-VLANs		
			Ag_W	Av_WL
10	6	4	24.5	21.5
15	10	6	32.8	27.2
20	12	8	30.3	24.4
25	14	10	32.1	26.9
30	17	13	34.4	27.9

FIGURE 5: Performance matrix for 5 different networks.

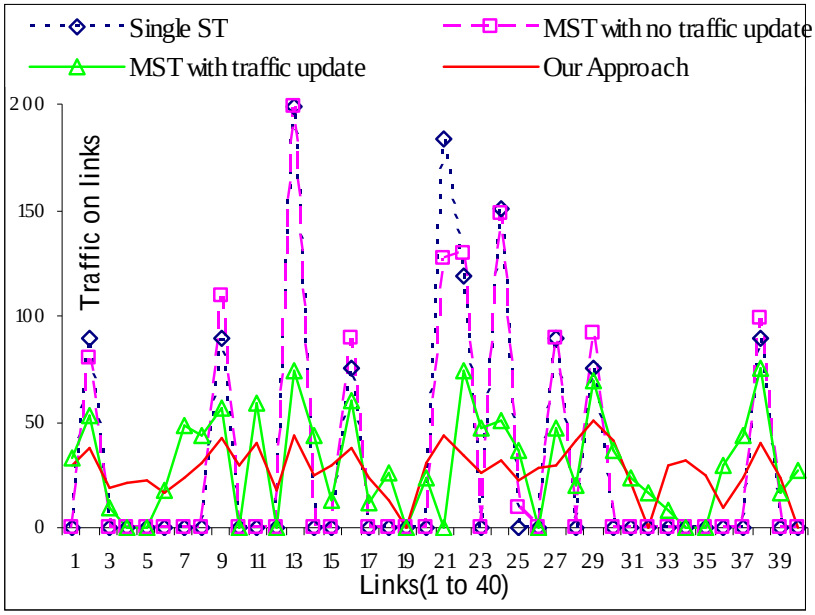

FIGURE 6: Traffic distribution in a 40 link network.

7. FUTURE DIRECTIONS

In this section we introduce a new traffic demand specification model that will be used in the future to evaluate the method presented in the previous section and other traffic engineering models for Metro Ethernet.

7.1 The Augmented Hose Model

This new model for traffic specification in Metro Ethernet is based on the observation that the customer sites for some of C-VLANs are usually clustered into components. As a result, the traffic can be classified as inter and/or intra-component traffic. The cases where the distribution of traffic is dominated by intra-component traffic are of interest for further optimization. In the case that there is no inter-component traffic, these components can be implemented as separate C-VLANs. However, if there is a percentage of the traffic that is to be delivered in the native technology (e.g., Ethernet) between these components, the hose model suffers from over-provisioning the bandwidth for the C-VLAN. Augmented hose model uses measurements obtained from the customer to greatly enhance bandwidth utilization in the provider network.

Consider for example the following hose model specification for the network in Fig. 7.Consider the bandwidth reservation for directed link (1, 2). Using the hose model, the bandwidth reserved is minimum value of: 1) the aggregate ingress bandwidth from A, B, and C, and 2) the aggregate egress bandwidth for D, E, and F. This value is computed to be min (222, 211) = 211.

Suppose now that measurements at customer points A, B, and C indicate that:

- At site A 85% of traffic goes to {B, C}
- At site B 80% of traffic goes to {A, C}
- At site C 65% of traffic goes to {A, B}

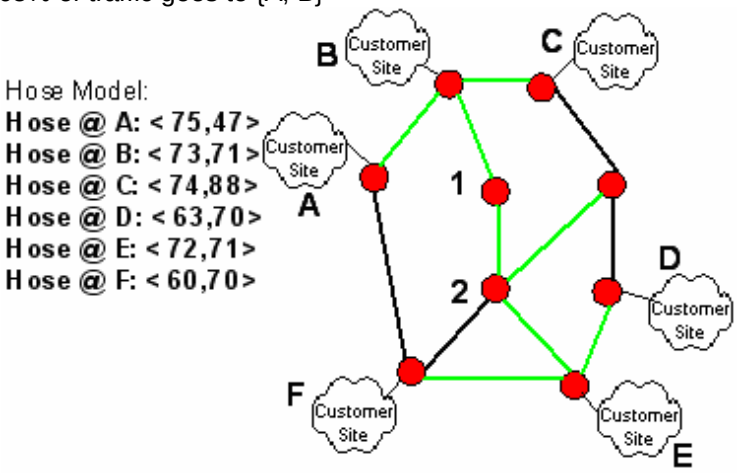

FIGURE 7: Sample network illustrating the benefits of hose augmentation.

This information can be communicated to the provider at the time of provisioning bandwidth for the C-VLAN in order to reduce the bandwidth reservation. For the directed link (1, 2), the bandwidth reserved is now equal to (1-.85)*75+ (1-.8)* 73 + (1-.65)*74 = 51.75.

Consider Fig. 8, where we show a C-VLAN A = {0, 1, 2, 3, 4, 5, 15, 16, 17, 18, 19, 20}. The C-VLAN is partitioned into two components: North and South.

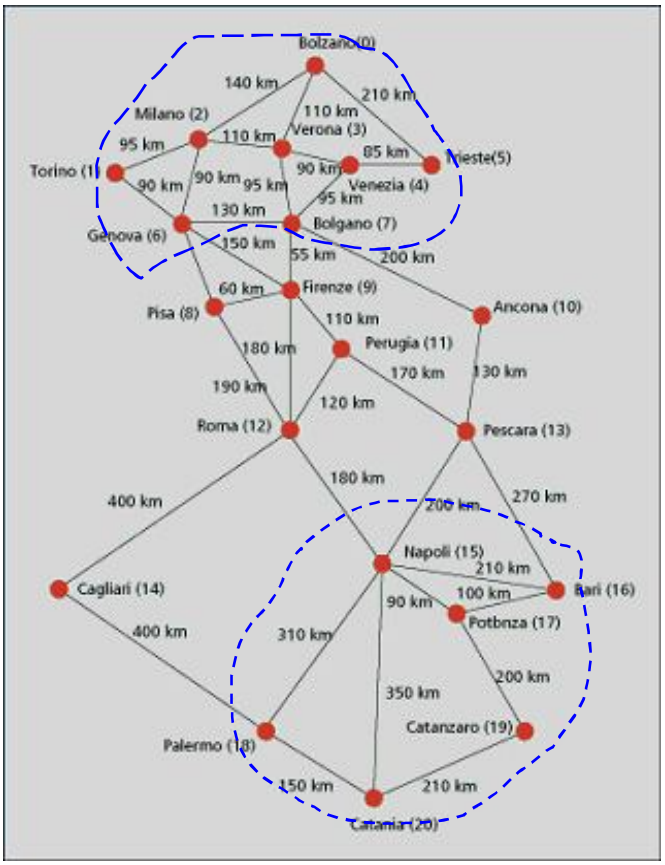

FIGURE 8: Network with a C-VALN composed of two components.

Fig. 9 shows the benefits of hose augmentation as a function of the inter-component traffic and using one ST. We notice that as the traffic increases between these components, the bandwidth savings using hose augmentation decreases. As expected, when the inter-component traffic reaches 50% of the traffic, the augmented hose model performance approaches to that of a regular hose model. However, under our assumption of non-zero inter-component traffic, we observe gains of about 25% in the case of 20% inter-component traffic. As Fig. 10 shows, these savings even increase when the number of STs is increased. The figure shows for example, that we have 36% savings in bandwidth in the case of 20% inter-component traffic and using 21STs.

The multiple spanning tree construction algorithm presented in section 5 will be analyzed in the future using this new traffic specification model. Further, in our subsequent papers we will study the performance of Augmented Hose and use this method for analysis of other traffic engineering methods.

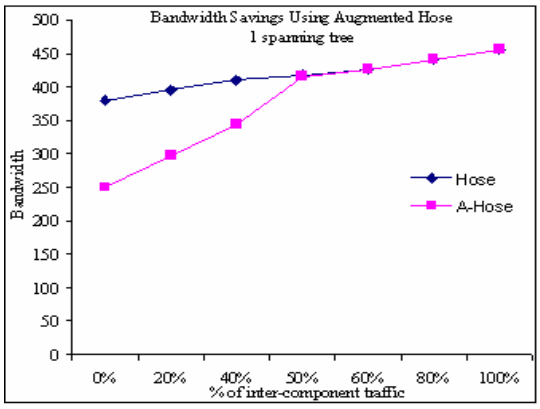

FIGURE 9: Enhancements to Bandwidth using 1 tree.

Fig. 9.

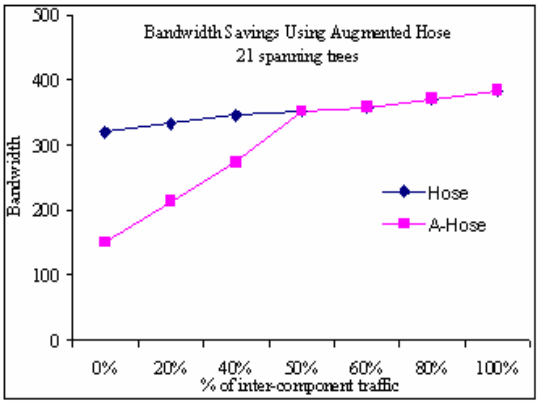

FIGURE 10: Enhancements to Bandwidth using 21 trees

8. CONCLUSIONS

Construction of spanning trees using well-known schemes such as Dijkstra's algorithm do not specifically address issues such as load sharing, efficient utilization of resources, QoS and reliability. For example, using Dijkstra's method could result in trees sharing a common pool of links. These links could get overloaded at the same time leaving some links idle. When considering a network topology such as Metro Ethernet, QoS is an important factor which depends on customers, their traffic demand and how well the given traffic can be distributed. In the paper we developed an iterative approach which uses traffic demand and dynamic link weight characteristics to fine tune the initial spanning tree to ensure that the resulting tree would satisfy the demand requirements and offer the best possible performance. Through a number of simulation experiments, we concluded that this approach can indeed increase the performance of spanning trees and keep traffic distributed efficiently in a metro domain. Further, we introduced a new traffic specification model, augmented hose, which uses measurements obtained from the customer to properly utilize bandwidth in the provider network. Through experimental studies we observed that this method performs better than hose model when inter-component traffic is below 50% of the traffic. In the future we will use this model to study performance of traffic engineering methods including the algorithm we suggested in this paper.

9. REFERENCES

1. IEEE 802.1q, "Virtual Bridged Local Area Networks", 1998.

2. IEEE 802.1d, “Media access control bridges”, 1998.

3. IEEE 802.1w, “Rapid spanning tree configuration”, 2002.

4. IEEE 802.1s, “Standards for Local and metropolitan area networks”, 2002.

5. A. Ge and G. Chiruvolu: “DiffServ-Compatible Fair Congestion Control through Extended Pause (DiffPause) for Metro-Ethernet”, IEEE ICC, 2004.

6. “Metro Ethernet networks - A technical overview”. Metro Ethernet Forum white paper, December 2003.

7. M. Padmaraj, S. Nair, M. Marchetti, G. Chiruvolu, and M. Ali, “Bandwidth sensitive fast failure recovery scheme for Metro Ethernet”, Computer Networks, Volume 52, Issue 8, 12 June 2008, pp. 1603-1616

8. Y. Li , Y. Bouchebaba, “A New Genetic Algorithm for the Optimal Communication Spanning Tree Problem”, Selected Papers from the 4th European Conference on Artificial Evolution, November, 1999, pp.162-173

9. Y. Lim, H. Yu, S. Das, S. Lee, M. Gerla. “QoS-aware multiple spanning tree mechanism over a bridged LAN environment”. IEEE Global Telecommunications Conference, 2003. pp. 3068-3072.

10. I. Hadzic,”Hierarchical MAC address Space in Public Ethernet Networks”, IEEE Globecom, 2001, pp.1563-1569.

11. M. Ali, G. Chiruvolu, and A. Ge, "Traffic Engineering in Metro Ethernet", IEEE Network, vol. 19, no. 2, March 2005.

12. S. Sharama, K. Gopalan, S. Nanda, and T. Chiueh, “Viking: A multi-spanning-tree Ethernet architecture for metropolitan area and cluster networks”, IEEE INFOCOM 2004, March 2004.

13. M. Padmaraj, S. Nair, M. Marchetti, G. Chiruvolu, and M. Ali, “Traffic Engineering in Enterprise Ethernet with Multiple Spanning Tree Regions”, In proceedings of IEEE/IEE International Conference on High Speed Networks, August 2005.

14. M. Padmaraj, “Quality of Service in Metro Ethernet”, PhD Thesis, Southern Methodist University, Dallas, Texas, May 2006