Simple and Stable Dynamic Traffic Engineering for Provider Scale Ethernet

António Teixeira
CITI and Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Quinta da Torre, 2829-516 Caparica, Portugal

José Legatheaux Martins
CITI and Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Quinta da Torre, 2829-516 Caparica, Portugal

***Abstract*—Traffic engineering defines the set of engineering methods and techniques used to optimize the flow of network traffic. Static approaches enjoy widespread use in provider networks, but their performance is greatly penalized by sudden load variations. On the other hand, dynamic traffic engineering is tailored to adapt to load changes. However, providers are skeptical to adopt dynamic approaches as these induce problems such as routing instability, and as a result, network performance decreases.**

This paper presents a Simple and Stable Dynamic Traffic Engineering framework (SSD-TE), which addresses these concerns in a provider scale Ethernet scenario. In addition, the validation results show that SSD-TE achieves better or equal performance to static traffic engineering approaches, whilst remaining both stable and responsive to load variations.

***Index Terms*—Provider Scale Ethernet, Traffic Engineering, Routing Algorithms**

I. Introduction

The widespread availability of Ethernet solutions, coupled with their increasing speeds and decreasing costs, has motivated the development of several Ethernet standards and protocols targeted at provider scale networks [1].

Moreover, ISP customers such as large enterprises may desire to connect geographically separate Ethernet LAN segments transparently, relying on their ISPs ability to provide Ethernet services on their backbone network. What is more, multicast traffic is becoming of primary importance in provider networks due to IP-TV deployments, among other applications.

Legacy routing in layer 2 Ethernet networks and VLANs entirely relies on the well-known spanning-tree protocols (STP). However, due to the new context of usage, current standards incorporate all the traditional set of L3/L2 protocols (e.g. OSPF, IS-IS, MPLS, PIM-SM).

The naïve way to run a successful network mixes shortest path routing with overprovisioning. Unfortunately, in large networks this method is uneconomical and cannot be used by a competitive provider. A more sophisticated alternative, quite suitable when the network load is relatively stable and predictable, is to use traffic engineering methods to provision the network and optimize the load distribution.

Traffic engineering concerns the adaptation of routing to network conditions, allowing for performance enhancements and efficient use of network resources [2]. Furthermore, it delays the need for network capacity improvements, decreasing expansion costs.

Most traffic engineering approaches follow a static paradigm. That is to say, the distribution of traffic is computed beforehand having as input previous measurements of network statistics. Assuming that traffic matrices remain largely constant during significant periods of time, this type of approach is feasible and yields near-optimal results.

However, this assumption does not always hold true. Flash crowd phenomena (e.g. flooding a news website) lead to sudden changes in the traffic matrix. These events render static traffic engineering useless, as the changes occur in a smaller time frame than that required to run the traffic engineering process and reconfigure the network [3].

In light of the above, static approaches to traffic engineering have a drawback of slow reactivity in the presence of traffic matrix modifications. Furthermore, the most common implementation of these approaches relies on standards and protocols (e.g. MPLS) that increase the complexity of network management and increase costs. This is a clear contrast with the light management of traditional Ethernet solutions. Therefore, dynamic traffic engineering schemes attempt to provide solutions to both these problems, albeit still optimizing routing and efficient use of the network resources.

Dynamic traffic engineering adapts the routing of traffic on-the-fly as the network conditions change. This approach minimizes centralized computations, placing the burden of calculating the traffic distribution on the network nodes. Typically, this implies that nodes must acquire information about network load through a signaling mechanism, in order to load balance traffic appropriately.

Despite the apparent advantages, dynamic traffic engineering solutions have had limited practical acceptance [4], due to most dynamic approaches inducing network instability, which greatly penalises TCP traffic. Moreover, most solutions are quite disruptive from today's protocols and architectures, and suffer from scalability problems. Coupling the disruptive nature of solutions with failed practical applications makes providers skeptical about adopting entirely new network schemes and architectures for dynamic traffic engineering, notwithstanding how adequate they might be.

The main requirement that dynamic traffic engineering solutions must address is *optimality*. Optimal routing should be guaranteed in respect to a given optimality criteria. Other than that, these approaches must address the following concerns:

a) Stability: The lack of stability is largely responsible for the low adoption of dynamic traffic engineering approaches. The traffic flow distribution should not vary needlessly in the presence of high traffic volumes or small changes in the network load. All the same, excess routing stability may lead to low convergence times whenever the traffic distribution must change to optimize routing.

b) Implementation Compatibility: Dynamic traffic engineering schemes should not force providers to change their network architecture and use untested protocols. The compatibility of approaches with proven and robust network standards increases their practical applicability and success. Furthermore, the achieved compatibility should not result in costly and complex management or configuration procedures.

c) Scalability: Aside from these design requirements, dynamic traffic engineering frameworks must be scalable to the high traffic volume expected in provider networks. Low signaling and computational complexities are vital for the success of such solutions.

Considering these design concerns, this paper presents the Simple and Stable Dynamic Traffic Engineering Framework, or SSD-TE for short. SSD-TE takes into account these concerns and provides a traffic engineering framework as follows. SSD-TE comprises a provisioning method which assigns the link capacities of a network. The provisioning is such that a given amount of traffic is guaranteed to fit the network if routed by the shortest path. SSD-TE forwards traffic with two different approaches: the largest amount possible is forwarded by the shortest path, whereas surplus traffic is forwarded using the ELB algorithm (detailed in section II). The traffic rates of both approaches are adjusted dynamically with a distributed optimisation algorithm as the network load changes.

The remainder of the paper is structured as follows. First, section II presents the SSD-TE framework in detail, whereas its validation is shown in section III. Furthermore, section IV describes current state-of-the-art approaches in dynamic traffic engineering. Finally, section V presents the conclusions and future work.

II. SSD-TE Framework

A. Presentation

From the simple naïve alternative of overprovisioning to that of static traffic engineering, complexity of network operations has grown dramatically, which in turn increases its operational cost. Moreover, real provider networks are often not stable at all, due to flash crowds phenomena and variations related with the evolution of costumers habits and the popularity of applications and sites. Thus, in practice, complex networks operations go hand in hand with overprovision due to load variations, flash crowds and the difficulty to measure or quickly adapt to the expected load. Very small and very large providers live quite well at the two extremes of the presented spectrum. However, this state of affairs is cumbersome in medium backbone networks.

A simple ideal way to operate a network could consist in provisioning it in a way that shortest path routing of the "commonly expected load" brings no congestion or stress to the network. However, whenever needed, the network should automatically resort to some form of load distribution of any (un) expected surplus load, using not only an already provisioned spare capacity, but also using any available capacity in whatever path could be used to route packets to their destination.

With this phylosophy in mind, the Simple and Stable Dynamic Traffic Engineering framework was developed. As any other traffic engineering framework, SSD-TE comprises a provisioning method, routing algorithms and a network status monitoring mechanism.

With SSD-TE, a given network is provisioned in a two-step way. First, an estimated traffic matrix (ETM) should be defined in a way that all packets belonging to traffic fitting in ETM must be routed by shortest paths and with no losses. ETM can be seen as the fraction of guaranteed high quality traffic the network should be able to transport. The actual fraction of the real expected traffic that will fit in ETM is an operator option.

ETM allows a first computation of the ingress traffic in each network ingress node. In accordance to SSD-TE, the operator is then required to define an incoming traffic surplus for each ingress node. Given ETM and the surpluses per ingress node, SSD-TE provides tools allowing the operator to compute the capacity of all links in the network in a way that the routing and load distribution method used by a SSD-TE managed network guarantees the following two conditions.

First, at any moment, the rate of the packets flowing from any ingress node to any egress node by the shortest path and without loss, is greater or equal to the one that is deemed to occur according to ETM for the same ingress - egress pair. Second, if the packet incoming rate at any ingress node does not violate the rate foreseen in ETM plus the surplus rate anticipated for that ingress node, all packets entering the network will be routed to the egress node without loss, by one of the available paths (not necessarily the shortest one).

To achieve these guarantees, SSD-TE uses shortest path routing at the base, complemented with load distribution according to the Ethernet Load Balancing (ELB) algorithm. ELB is based on an earlier proposal, the Valiant Load Balancing (VLB) routing architecture [5], which is described as follows. The goal of VLB is to provision the minimum amount of link capacity while assuring that all valid traffic matrices fit the network. For this purpose, VLB assumes the network comprises a full mesh of logical links between all n nodes, and that each node has a maximum ingress traffic rate of r. VLB works by using a two-hop routing scheme: the ingress node forwards a flow to an intermediate node, chosen in a round-robin fashion from the set of all nodes, and then the intermediate node forwards it to the destination node. If each logical link is dimensioned with a capacity equal to $\frac{2*r}{n}$, then all traffic matrices fit the network.

The ELB architecture further builds upon VLB, splitting traffic in a round-robin fashion to spanning-trees rooted at the different network nodes. What is more, this approach still guarantees maximum throughput for valid traffic matrices.

An SSD-TE engineered network, by default, routes packets using the shortest path, and continuously evaluates the state of packet transport in each ingress / egress nodes pair, e.g. (ni, nj). Whenever some form of congestion is deemed to occur from ni to nj, "surplus" packets are forwarded using ELB. These monitoring and forwarding methods, complemented with the SSD-TE provided provisioning tools, guarantee that the above conditions are respected.

B. Architecture

The SSD-TE framework comprises two main components: a provisioning method and a traffic distribution adjustment algorithm.

1) Provisioning Method: This method works by assigning enough capacity to the links in order to ensure that any traffic matrix which maximum ingress rate per node i does not exceed $ETM_i + \eta_i$ fits the network. Assume that for each flow[1] s, an ingress rate of r_s is guaranteed to be forwarded by the shortest path.

Additionally, an added ingress rate of η_i is tolerated per node, which is to be distributed using ELB with a gravity full mesh approach. To this end, the capacity c_{ij} of the ELB logical link between physical nodes i and j is given by:

$$c_{ij} = \frac{2\,\eta_i\,\eta_j}{R} \tag{1}$$

All in all, the total provisioning of a physical network link is given by the sum of two different amounts. First, the aggregate estimated ingress rate of all flows which shortest path traverse the link. This guarantees that the estimated traffic matrix fits the network using shortest path forwarding. Second, the total ELB logical link rate of all logical links that traverse the physical link. In conclusion, equation 2 gives the total capacity of physical link l.

$$c_l = \sum_{l \in shortestPath(s)} r_s + \sum_{l \in logicalPath(i,j)} \frac{2\,\eta_i\,\eta_j}{R} \tag{2}$$

2) Distribution Adjustment Algorithm: The forwarding scheme of the framework specifies two distinct rates per flow, one for shortest path forwarding and the other for ELB distribution. These can be set statically based on the ETM used in the provisioning method. However, this approach might not be optimal depending on the network current traffic matrix. In other words, the amount of traffic routed by the shortest path can be superior to the rates set by the provisioning method, without causing the traffic matrix not to fit the network.

[1]Note that in the context of SSD-TE, a flow designates a sequence of packets with an ingress node and an egress node in the backbone network (as opposed to a TCP flow).

The optimal traffic distribution can be calculated statically using a linear programming approach. For a dynamic approach, the gradient projection algorithm is used. This method works by iteratively modifying the traffic rates per flow. These modifications are processed in the opposite direction of the gradient of a chosen cost function, and are then projected onto the feasible space (which is defined by the model's constraints).

In SSD-TE, the approach used to model the gradient projection problem is an adaptation of that used in the MATE proposal [6], calculating the traffic distribution distributedly for each network flow.

The selected cost function C_l of the model has the main objective of steering the traffic distribution towards a desired state by attributing a cost value to a given link l. With this goal in mind, consider the expected waiting time of a M/M/1 queue, where the service rate is c_l and the arrival rate is x_l, is given by $\frac{1}{c_l - x_l}$. If c_l is taken to be the capacity of link l and x_l the current traffic rate at the said link, this waiting time represents an approximation of the queueing delay at link l. Consider that the estimate of the propagation delay p_l of link l further influences the link delay. We can derive a convex cost function C_l for link l as follows:

$$C_l(x_l) = t * p_l = \frac{p_l}{c_l - x_l} \tag{3}$$

This approach yields the lowest values of $C_l(x_l)$ for short (in terms of propagation delay) and uncongested links. In turn, this means the algorithm steers the traffic distribution towards shortest path forwarding, whenever the network is not congested. However, links approaching their capacity return increasing values of $C_l(x_l)$, which makes sure that no link is overrun.

This cost function is shown to conform to the conditions required for the convergence of the gradient projection algorithm to an optimal traffic distribution, provided that a *stepsize* parameter value is chosen sufficiently small. In practice, this parameter controls the trade off between the algorithm's convergence speed and stability.

C. Implementation

The provisioning method can be calculated in a centralised machine prior to using the rest of the framework. This calculation is processed by an algorithm that receives as input the network topology with link weights (e.g. latencies), the estimated traffic matrix, and the desired traffic margin. The output is the set of all link capacities.

Furthermore, the network must handle the two routing types used by the framework: shortest path routing and ELB. The former is implemented in an Ethernet scenario by forwarding frames onto their destination sink tree, i.e., a shortest path spanning tree with the egress node as the root. It follows that each network node needs to maintain state regarding all distinct node-rooted trees. In this case, no forwarding table per tree is necessary: all that needs to be kept is the path towards the tree root.

On the other hand, ELB requires that traffic is split among all network shortest path spanning trees, each having a different node as the tree root. Thus, the requirement is similar to that of shortest path routing: each network node must maintain state per network spanning tree. However, nodes must have appropriate forwarding tables for each tree. This happens because frames traversing a network tree may not be directed towards its root, but to one of its leaves.

Since at provider scale networks, MAC learning and spanning tree protocols are undesired, the topology of these VLAN trees along with their respective forwarding tables should be pre-calculated and then stored onto the nodes. A possible implementation mapping for this approach is the Provider Backbone Bridges – Traffic Engineering (PBB-TE) [1] architecture.

Nevertheless, signaling the network load presents a more delicate situation. This challenge arises from the fact that there is no signaling protocol in the network which can be used for piggybacking load information. Furthermore, there is no possibility of inferring network load using the network traffic, such as by inspecting TCP connections. This happens because frames in symmetric flows (with reverse ingress-egress node pairs) may not follow the same network path using this framework, which in turn makes it impossible to calculate statistics, e.g. end-to-end delay.

A possibility to implement load signaling relies on the sending of end-to-end messages as the payload of Ethernet frames. The payload can be modified by the nodes to change the appropriate load values as it traverses a network path. It follows that this approach is relatively simple to implement and maintain. However, it comes at the expense of injecting further traffic into the network.

III. Validation

A. Model

1) Network Model: The network model to be used to validate SSD-TE resembles a provider backbone packet-switched network. A network flow can follow one or more paths in the network between the ingress and egress nodes. Each path comprises the ordered set of links which it traverses from the ingress to egress nodes. The stated model assumes that of failures in nodes or links and multicast traffic are outside of scope.

2) Network Topologies and Link Capacities: The network topologies used for the framework validation were based on the Rocketfuel ISP maps [7]. For simulation purposes, the POPs of a given ISP were considered as the simulation nodes. In addition, the Rocketfuel latencies between POPs were used as the respective link latencies.

On the other hand, the link capacities were assigned by the means of the provisioning method described in the previous section.

3) Traffic Model: The traffic generation is driven by a traffic matrix and the traffic type. Two traffic types are defined as follows, constant bit rate (CBR) and variable bit rate (VBR) traffic. The frame inter-arrival time in CBR constant. On the other hand, VBR traffic maps frame inter-arrival time using an exponential distribution.

4) Matrix Modification Model: A model of dynamic matrix modification was used to test a traffic engineering scheme's response to unpredictable changes in the traffic matrix. This model implies the change from one traffic matrix m to a traffic matrix m' where the rate of a single flow from ingress node i to egress node j has been increased by a given fraction f. The change is done progressively, following a linear pattern over a time period of duration t. This means that at any instant t_k between the modification start time t_i and end time t_f (where $t = t_f - t_i$):

$$m'_{ij} = m_{ij} * (1 + f * (t_k - t_i))$$

5) Performance Metrics and Optimality Criteria: A performance metric was necessary to evaluate the SSD-TE approach. Let $u(f_{ij})$ designate the utility value for the flow f between ingress node i and egress node j. In addition, the previous section has stated that the traffic types that will be tested are inelastic.

The utility value combines the packet loss probability p_f and queuing delay q_f for flow f, which assuming a best-effort QoS model (all packets have the same service class) can be calculated as:

$$u(f_{ij}) = 1 - \alpha_p * p_f + \alpha_q * \frac{q_f - q_r ef}{q_r ef}$$

$q_r ef$ is the reference queing delay for flow f whereas $\alpha_p = 10$ and $\alpha_q = 1$. The value for α_p makes the value of the utility function become zero for a 10% packet loss and the value for α_q guarantees that the flow utility is equal to zero when the queuing delay doubles the reference value.

In order to provide a comparision standard for SSD-TE regarding the utility metric, the utility results for a static traffic engineering scheme are considered as the optimal solution for a given traffic matrix. The static traffic engineering results are calculated resorting to the optimization of the multipath path selection model presented in [3]. The objective function minimizes the network's congestion costs metric for each link, selecting a set of network paths for each flow in the process.

B. Tools

The validation of SSD-TE using the model specified in the previous section was done using the OMNeT++ [8] simulator. This choice relies on the fact that the model required significant changes to existing protocol implementations and OMNeT++ provides the most sound alternative for extension, due to its modular structure and open source code. Furthermore, a SSD-TE Java tool was built during the development of the framework, which was used for tasks such as the generation of static traffic distributions and the implementation of the provisioning method.

C. Test Cases

Several test cases were defined in order to create simulation instances to be run by OMNeT++. The test cases were built by assigning values to the variables of the validation model, namely: network topology, provisioning type, traffic type (CBR and VBR) and traffic matrix.

a) Network Topology: The AboveNet and SprintLink were the selected topologies from the Rocketfuel database. The considered topologies belonged to United States ISPs, each having significantly distinct size (in number of nodes/links) and average node degree.

b) Provisioning Type: To adequately provision the network with the traffic engineering framework, a traffic surplus factor γ was used. The tested γ values were 0.2 and 0.5.

c) Traffic Matrix: A random ETM matrix was generated for each test case, in order to provision the network, with an average ingress rate per node of 100Mbps. The actual traffic matrix of the test run, M, was built from the ETM matrix by increasing the ingress rate per node:

$$M_i = ETM_i \times (1 + \gamma) \tag{4}$$

Depending on the increasing variability of the flow rates per node, the real matrix is charaterised as symmetric, mixed or asymmetric.

D. Results

This section presents and analyses the results obtained for the test cases specified in the previous section. The results were generated by OMNeT++ simulation runs. Each run spanned 90 seconds of simulation time, with two flow modifications (with $f = 2.0$ occuring between $t_i = 30s$ and $t_f = 60s$).

Due to space constraints, only two test results are presented. These results correspond respectively to the worst and best case scenario for the SSD-TE framework. The complete set of results, as well as a more thorough analysis of the validation results, is presented in [9].

Figure 1 shows the average utility values for two network configurations: $\gamma = 0.2$, VBR traffic and symmetric matrix; $\gamma = 0.5$, CBR traffic and asymmetric matrix.

In general, results show that the dynamic framework reacts to flow modification, increasing its utility values after a period of time.

Figure 1.a) shows the worst test case: SSD-TE slightly underperforms on symmetric traffic matrices, specially until flow modification changes the network traffic pattern. This is an expected scenario, since symmetric matrices have an even distribution of traffic between flows, and as a result the pre-calculated static traffic distributions remain suitable. However, this type of even matrix distribution is not the most common in practice and, on top of that, the framework results are still reasonably high.

On the other hand, figure 1.b) presents the best case scenario. SSD-TE manages to significantly increase performance for asymmetric matrices, particularly when the provisioning factor γ is large. This is explained by two reasons. First, the higher variability in the distribution of surplus traffic. Second, the large provisioning factor leading to a high margin of adapatability for the dynamic framework, which is even more noticeable after the flow ingress rates are modified.

The global validation results show that both types of traffic to do not cause significant differences between the framework and the static approach. However, VBR tends to yield lower average utility results than CBR, which can be explained by the occurence of sudden traffic bursts due to the exponential distribution of frame arrival times.

Finally, the traffic distribution achieved by SSD-TE was not shown to exhibit sudden changes or oscillations, even when considering significantly different values for the stepsize algorithm parameter. Specifically, the modifications done by SSD-TE to the traffic rates iteratively become smaller as simulation time progresses, provided the traffic matrix remains itself stable.

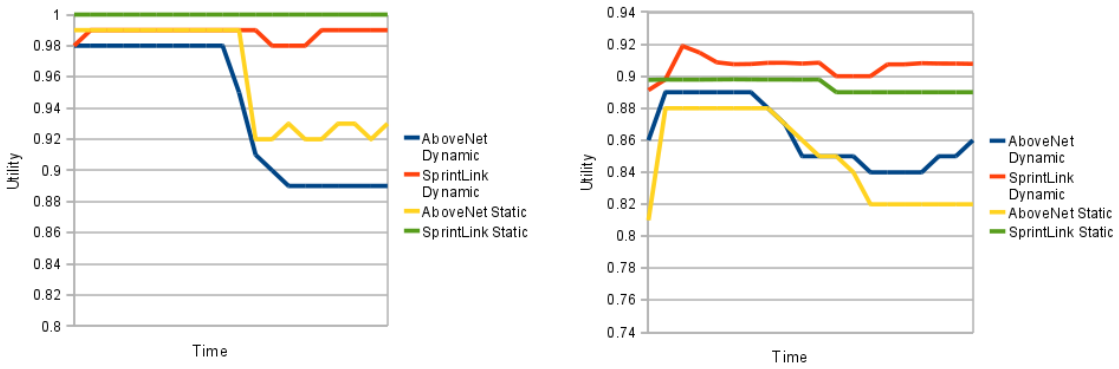
IV. Related Work

Current state-of-the-art dynamic traffic engineering approaches have as their main focus optimality for unicast traffic, scalability and routing stability. Some of these approaches are listed in the following paragraphs.

The Distributed Adaptive Traffic Engineering (DATE) [10] approach takes into consideration the aforementioned requirements, paying specific attention to the congestion control behaviour of TCP. Although dynamically adapting the percentage of traffic that follows each network path, DATE still requires that all paths are previously computed in the network. Finally, congestion-aware routing is performed by keeping track of TCP sessions at the network edge nodes and adjusting their routing accordingly.

The REPLEX traffic engineering scheme [4] works by adjusting the weights of available outgoing links based on collected network statistics. The algorithm works by using a game theory model, observing the concept of Wardrop Equilibrium. At the Wardrop equilibrium, all agents (which can be seen as flows) do not gain performance by picking any other path than their current one. The statistics, namely the expected latency for a link, are gathered using message exchanges between neighbour nodes in a distance-vector fashion. Network oscillation is avoided, both by increasing flow shifting probability as network load changes increase and by fine tuning algorithm parameters.

The MPLS adaptive traffic engineering (MATE) [6] method requires the existence of multiple virtual paths (specifically, MPLS label switched paths) between all pairs of nodes. These paths are used to define an optimisation problem similar to those used in static traffic engineering solutions. The resulting constrained optimization problem is solved using gradient projection. The gradient projection method iteratively adjusts routing in the opposite direction of the defined gradient. Additionally, MATE uses probe packets in order to measure the delay and packet loss ratio of a given label switched path, which influence the path cost in the said optimization problem and therefore the traffic distribution among the available paths.



(a) Symmetric matrix, $\gamma = 0.2$, VBR traffic

(b) Asymmetric matrix, $\gamma = 0.5$, CBR traffic

Fig. 1. Utility results

Other promising proposals exist, such as TeXCP [11] or the Potentials routing scheme [12]. However, most of these approaches are disruptive and, consequently, difficult to implement in practice. SSD-TE provides this implementation compatibility, whilst still remaining optimal and stable.

V. Conclusions and Future Work

This paper presented a dynamic traffic engineering framework, the Simple and Stable Dynamic Traffic Engineering (SSD-TE), which targeted the design requirements of unicast traffic optimality, stability, implementation compatibility and scalability.

The previous sections showed that SSD-TE met the goals stated in the introduction. The framework not only achieved resuls similar to those of static traffic engineering solutions for provisioned traffic matrices, but also outperformed the latter when the network load dynamically changes. This claim is based on two validation results.

First, SSD-TE converged to a traffic distribution that yielded flow utility values similar to those obtained by the static traffic engineering approach (using link congestion costs as the optimisation metric). Second, SSD-TE surpassed static traffic engineering utility values after load modifications were induced, in the majority of the test case runs.

Other than the optimality results, the validation section showed SSD-TE's capability to remain sufficiently stable in various load instability scenarios. Moreover, SSD-TE was considered to be both scalable and compatible with existing provider Ethernet standards.

The concerns of multicast traffic optimality and tolerance to node and link failures were not considered in SSD-TE. Undertaking work on these design requirements remains as future work. However, there are possible directions to tackle these concerns.

The MPLS multicast tree (MMT) [13] approach for multicast traffic engineering relies on integrating explicit path forwarding architectures, such as MPLS or PBB-TE, with multicast traffic engineering trees. This type of solution might cope well with the SSD-TE framework, which is also targeted at an explicit routing architecture and deals with forwarding trees.

Furthermore, the VLB network architecture can be easily adapted to deal with node or link failures, provided that extra capacity is provisioned. In order to tolerate k node failures in a network of N nodes, with r being the maximum ingress rate per node, each VLB logical link requires a capacity of $\frac{2r}{N-k}$.

Finally, the framework requires several improvements to its validation model, namely by adding more realistic traffic models and fully simulating a PBB-TE architecture.

References

[1] P. Bottorff and P. Saltsidis, "Scaling provider ethernet," *Communications Magazine, IEEE*, vol. 46, no. 9, pp. 104–109, September 2008.

[2] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional ip routing protocols," *Communications Magazine, IEEE*, vol. 40, no. 10, pp. 118–124, Oct 2002.

[3] O. M. Heckmann, *The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design*. John Wiley & Sons, 2006.

[4] S. Fischer, N. Kammenhuber, and A. Feldmann, "Replex: dynamic traffic engineering based on wardrop routing policies," in *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*. New York, NY, USA: ACM, 2006, pp. 1–12.

[5] R. Zhang-Shen, "Designing a predictable backbone network using valiant load-balancing," Ph.D. dissertation, Stanford, CA, USA, 2007, adviser-Mckeown, Nick.

[6] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "Mate: Mpls adaptive traffic engineering," vol. 3, 2001, pp. 1300–1309 vol.3.

[7] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 133–145, 2002.

[8] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," ICST, Brussels, Belgium, Belgium, 2008, pp. 1–10.

[9] A. Teixeira, "Simple and stable dynamic traffic engineering for provider scale ethernet," *MSc Dissertation, Universidade Nova de Lisboa*, 2010.

[10] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards robust multi-layer traffic engineering: Optimization of congestion control and routing," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 5, pp. 868–880, June 2007.

[11] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in *ACM SIGCOMM*, Philadelphia, PA, August 2005.

[12] A. Basu, A. Lin, and S. Ramanathan, "Routing using potentials: a dynamic traffic-aware routing algorithm." New York, NY, USA: ACM, 2003, pp. 37–48.

[13] A. Boudani and B. Cousin, "Mpls multicast traffic engineering," in *In IEEE ROCC*, 2003.