Traffic Engineering using New VS Routing Scheme

Girish P. Saraph and Pushpraj Singh
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai – 400076, India
girishs@ee.iitb.ac.in

Abstract— **Goal of traffic engineering in packet networks is to improve the network performance by providing support for congestion management, higher bandwidth utilization (or throughput), and QoS or priority-based services. Open shortest path first with traffic engineering extensions (OSPF-TE) [1] and and Multiprotocol Label Switching (MPLS) protocols are commonly viewed as possible solutions. Both, the OSPF-TE with optimized link weights or MPLS with explicit optimal path set-ups, work ideally under static network conditions with a known demand-matrix. However, these methods do not provide scalability and flexibility to adapt to arbitrary, dynamic demand patterns in large networks. A novel traffic engineering scheme based on the Virtual Space (VS) routing [2] is proposed here, which has the scalability, flexibility, and robustness to rapidly adapt to arbitrary, dynamic load conditions in large networks. Simulations are carried out on randomly constructed 40, 80, and 200-node networks with arbitrary demand matrices, which demonstrate excellent capability of the VS routing in terms of load balancing, packet throughput, and congestion avoidance. The proposed VS scheme can be used for path selection in MPLS and integrated with the signaling protocols for path establishment, such as the constraint based routing (CR-LDP) [3] or resource reservation (RSVP-TE) [4] protocols.**

Keywords- Traffic engineering; packet routing; MPLS; QoS

I. INTRODUCTION

As the core IP networks grow in size and complexity, it is becoming increasingly important to have effective traffic engineering tools to optimize their performances. As the traffic grows and the demand for bandwidth increases, the network may develop several traffic hot-spots that lower the overall performance in terms of lower IP throughput, higher loss rates, and varying time delay. It is not cost-effective to undertake infrastructure build-outs or upgrades until the existing network resources have been optimally used. Hence, it is essential to have a traffic engineering tool that achieves good load balancing to reduce network congestion and improve bandwidth utilization. It also helps in supporting better service level guarantees to the end-users. The majority of the traffic on the Internet is inherently highly bursty in nature. The peak to average bandwidth requirements on the link can be very high and the traffic remains bursty even after aggregation or statistical multiplexing. Thus, the dynamic load conditions in the network can change rapidly. A good traffic engineering scheme should adapt to the dynamic load conditions quickly without adding excessive performance penalties in terms of resource overheads, convergence time delay, reduced stability, or poor scalability.

Commonly used routing protocols rely on information about the network conditions such as, a node or a link failure, to flow from the distant network node to all participating router nodes. This information flows either directly (in link-state type protocols, such as OSPF) or indirectly (in distance-vector type protocols, such as RIP or BGP). Thus, each router forms a consistent picture of the network, which is used to build and update the route look-up tables based on the dynamic network conditions [5]. Presently, a typical core IP router may have look-up table entries in excess of 150,000 [6], which makes the routing and forwarding tasks of the router rather complex and costly. In addition, if the individual link costs or weights are changed frequently to reflect the dynamic load conditions then, it would lead to excessive overheads in terms of routing information exchange, processing, memory storage, and I/O.

One of the main advantages of MPLS protocol is to support traffic engineering based on specific label-switched path set-ups [7]. A possible intra-domain routing algorithm that can be used for MPLS path selection is OSPF with traffic engineering extensions or OSPF-TE [1]. OSPF-TE uses weight assignment to individual links and uses Link State Advertisements (LSAs) to flood this information. This information exchange can be both periodic as well as event-driven such as, a change in the link load above a threshold level. After every LSA update the shortest-paths are recomputed in terms of cumulative link weights using the Dijkstra's algorithm and the routing tables are updated. As the forwarding engines get the updated information, the packet-flow traffic adjusts accordingly. As the traffic pattern changes it may trigger further LSAs due to load changes. Thus, every LSA update may trigger some cascaded updates as the network settles down to a new traffic pattern. A similar but more drastic effect is observed in case of a node or link failure. After such an event, flurry of routing information exchanges and processing updates take place leading to high processor utilization [8]. It has been observed that under multiple failures the processor utilization remains high for a few hundreds of seconds [8]. As the processor is pre-occupied with this information processing, it is not available for doing other tasks. This can overwhelm the internal queues in the router leading to packet drops. Furthermore, any missed control packets may lead to a false interpretation as a loss of adjacency and lead to additional flooding of information. This is potentially an unstable situation and would be more severe for larger networks. The frequency of the LSA updates and the average processing time after each LSA update increases with the size of the network. As the amount of control information exchanged and its frequency increases, a large network may be driven into unstable operating regimes. Such events, although rare, can be catastrophic for the end-users.

For every known demand matrix and network topology, there is a set of optimal link weights to give the best possible network performance. However, finding of these optimal weights is shown to be NP-hard [9] and many heuristic algorithms have been proposed. The simplest types of weight assignments are static, which ignore dynamic link loading completely, *e.g.* the weights can be all unity (*i.e.* simple OSPF with no traffic engineering capability), or inversely proportional to the link distances (*i.e.* based on the propagation delay), or inversely proportional to the link capacity (proposed by Cisco [10]). When link costs are related to actual delays on the links, it may lead to oscillatory behavior in traffic flows. Whereas, when the link weights are assigned based on inverse of link capacity, it ignores the actual link loading that governs the performance. Routing based on static or quasi-static traffic conditions ignores the dynamic nature of the load conditions. Heuristic traffic engineering algorithm based on approximate minimum delay routing has been presented in [11, 12]. The link weights based on the actual link utilization has been proposed in [9, 13], wherein a piece-wise linear, convex cost function is used to define the initial link weights. The network performance for a known demand matrix is optimized iteratively to determine a sub-optimal solution with only a few weight changes. The iterative method uses a heuristic approach of local search in the parameter space to march towards the solution. The demand matrix is generated from the traffic data aggregated over long time intervals (hours and days) [14] and quasi-static solution is determined. The two main reasons for employing the static or quasi-static approaches are: one, the computational complexities of the algorithms involved and two, the frequent cascaded updates can lead to excessive overheads and stability problems due to slow convergence of routing tables in the above mentioned approaches.

The VS-based traffic engineering scheme presented here, is a fully dynamic solution that is inherently stable and highly scalable. It aims at minimizing the routing information exchange, processing, memory storage, and all the associated costs. This scheme can be viewed as employing the local search heuristic in the virtual space topology. The detailed VS-based routing scheme has been presented in [2]. It is used here specifically for the traffic engineering study. The VS routing scheme separates the network information into static (or slowly changing) part related to the network topology and the dynamic part related to the network conditions. The static topology information is made available to each node in the most simplified form using the virtual space configuration. This information is exchanged only while initializing or in rare updates (say, in days and weeks). The dynamic information, including link loading, any node or link failure, is expressed in terms of the link and node costs. The dynamic link cost information is exchanged frequently (say, 25 ms intervals), but to the direct neighbors only. Since the VS link cost updates neither flood the whole network nor lead to any cascaded updates, this algorithm is highly scalable and stable under dynamic conditions. The VS forwarding path towards the destination is selected using the VS coordinates and link costs.

A concise version of the VS formulation based on [2] is included in Section II for the sake of completeness. This paper deals with the VS routing as an overlay tool that is compatible with other protocols (MPLS or ATM) and provides traffic engineering support. It is proposed that the VS based routing can be used for the initial selection of the label-switched paths in MPLS [7]. The path set-up process can be integrated with the MPLS signaling protocols (CR-LDP or RSVP-TE). The dynamic link cost formulation used for the present traffic engineering studies is described in Section II. In Section III, the results from the simulation studies have been presented, which demonstrate the efficacy of VS routing including the load balancing and congestion avoidance properties. These simulations are carried out on arbitrary networks of 80 and 200 nodes and a realistic network (BTnet backbone) with arbitrary demand patterns. The paper is concluded by summarizing the results and the advantages of the proposed scheme.

II. VS Routing Scheme

A. *Virtual Space Transformation*

The critical aspect of the VS routing is the automatic path directivity that is enabled by the virtual space embedding. This is achieved by making the distances between every node pair to closely match with the least number of hops between them. The VS configuration is such that a directed VS distance to any destination node indicates the available path options with the least (or low) number of hops through the network. It is equivalent to having a multi-dimensional road-map of the network topology for navigation of packets.

The detailed formulation of the VS embedding is given in [2]. Here it is summarized for the sake of completeness. The transformation of an initially planar (2-D) network into the virtual space representation of dimensionality N_d, is achieved by letting the network evolve under the influence of a set of virtual forces in the multi-dimensional virtual space. The forces are defined as follows:

1) Force on the directly connected nodes: This force, called F_{1ij}, acts like a spring force between the nodes n_i and n_j that are directly connected and tends to make the distance between the directly connected nodes close to unity.

2) Repulsive force on the nodes not connected directly: This force, called F_{2ik}, acts on the nodes n_i and n_k that are not directly connected to push them apart.

3) Random kick force: This is a deterministic, pseudo-random force F_{3i} used to kick individual nodes in the multi-dimensional VS configuration.

All such forces add vectorially, to give total force $\mathbf{F}_{ti}$ acting on each node n_i. For every iteration, the node n_i moves under the action of $\mathbf{F}_{ti}$ by a small distance Δ_i that is proportional to the force. The network is allowed to evolve in the multi-dimensional VS space under the influence of the forces specified above for a fixed number of (say, 40) iterations. The network slowly saturates to a final VS configuration. The final configuration has all the network topology information embedded in it and exhibits excellent directivity property for routing packets. It is envisaged that the VS configuration would have to be recomputed after a sufficiently long duration (days or weeks), if the network topology changes significantly. Whereas, any dynamic network conditions such as link or node failures and changes in the link or buffer utilizations are reflected in the respective link and node costs. These dynamic conditions do not require recomputation of the VS configuration.

B. VS-based Routing

The packet forwarding decisions in VS routing are based on the VS coordinates and the dynamic costs. The dynamic link (or node) failure or loading conditions are expressed in terms of link and node costs and advertised only to the direct neighbors. As an analogy, while traveling by road to a distant city, it is sufficient to head towards it without knowing the exact traffic conditions near the final destination. In a well-connected road network, a suitable detour would be found in the local vicinity of the problem area having blockage or congestion. Here, the directivity is based on the VS coordinates and the local vicinity refers to the direct connections and not the physical distances. Thus, this scheme reduces the routing information exchange significantly, including the overheads and the associated costs. It also makes the scheme highly dynamic, robust and scalable.

The VS forwarding at a starting or intermediate node, n_i, is based on the VS addresses of the final destination node, n_D, of the packet and the VS addresses of the forwarding node n_i and all its neighbors. A cost function, C_{ij}, is evaluated for each outgoing link from the node n_i to its neighbor n_j given by,

$$C_{ij} = C_1 * (1 - \cos\theta_j) + C_{2ij} + C_{3j} . \qquad (1)$$

Where C_1 is a constant and θ_j is an angle between the directed distance to the final VS destination $\mathbf{d}_{iD}$ from n_i to n_D and the outgoing link distance vector $\mathbf{d}_{ij}$ from n_i to n_j in the multi-dimensional virtual space. The second term in the cost function is a link cost function, C_{2ij}, for the link between n_i to n_j nodes and reflects the dynamic traffic conditions. The third term C_{3j} is a node cost term for node n_j and is an average of all the outgoing links from that node. This gives a "forward visibility" to avoid any network congestion further downstream.

The VS routing can be either single-path or multi-path, depending upon the channel requirements. In single path routing, each packet is forwarded along the link having the lowest total cost. The advantage of single path routing is that as long as the link costs are unchanged, a given data stream will follow the same path and the packet sequence remains unchanged. In multi-path routing, one or more low cost paths (with total costs within a fixed cost differential of +0.3 from the lowest available value) are chosen with equal probability. Rather than availing the multiple path options at the packet level, they can be chosen on a per flow basis using hash-tables [15], in order to avoid out-of-order packets in the TCP flows. Multi-path routing has a natural tendency to distribute any traffic load between the available paths. The VS-based routing scheme can incorporate the multi-path and stochastic routing without any additional resources in terms of time, cost, storage, or complexity. Thus, it provides a quick recovery or protection from any link or node failures.

C. Traffic Engineering using VS-based Routing

The dynamic link cost, C_{2ij} in (1), is dependent on the link and buffer capacity utilization levels. As the cost increases, the link becomes less attractive vis-à-vis other path choices, if available. As the links with lower capacity utilization are selected their utilization improves and congestion on the heavily loaded links is alleviated. Thus, the traffic pattern evolves dynamically to achieve efficient load balancing. In addition, the node cost, C_{3j} in (1), also gives a forward 2-hop visibility. A linear link cost function based on the link utilization is used here as given by,

$$C_{2ij} = C_{20} + C_{21} * (L_{ij} - L_{avg}) / \Delta_L . \qquad (2)$$

Where, constants C_{20} and C_{21} are fixed at 0.40 and 0.30 in the simulations presented in Section III. L_{ij} is the link usage or loading of the link from node n_i to n_j, and L_{avg} is the mean value for all the links. Δ_L is a scaling factor, which is fixed to the standard deviation of the link usage distribution (L_{ij} values) after the first iteration. These normalization constants bring the cost distribution for most of the links within the (0,1) interval. Certain more complex, linear and nonlinear link cost functions have also been tried and have shown some improvements. However, those cost functions are not discussed in this paper.

The loading level for each link is determined by counting how many packets traversed that link in the previous iteration. The simulations presented here, assume equal link capacity on all the links, otherwise the link usage would have to be scaled down by the capacity to get the link cost that reflects the utilization level. The link costs are modified after every iteration depending upon the link usage L_{ij} for each link. If the link costs are allowed to fluctuate with drastic amplitudes, it may lead to route flaps and load oscillations. Hence the cost is updated to a weighted average of the newly proposed link cost, $C'_{2ij\,(N)}$, and the previous cost, $C_{2ij\,(N-1)}$, as given by,

$$C_{2ij\,(N)} = \alpha \,.\, C'_{2ij\,(N)} + (1 - \alpha)\,.\, C_{2ij\,(N-1)} . \qquad (3)$$

Where, α is a constant set at 0.3 to avoid load oscillations and the iteration numbers are indicated in brackets by, (N) and (N–1). Each iteration corresponds to one link cost update interval in the VS protocol and its value may be fixed at 25 or 100 ms in the practical implementation. As the link costs change, the traffic pattern evolves dynamically under a given arbitrary demand matrix. The load balancing is achieved due to the feedback between the path selection process and the link cost updates.

Instead of packet switching, the VS routing may also be used in the control plane to select label-switched paths in MPLS (or virtual circuits in ATM). A possible implementation could be based on sending multiple path-selection packets that are routed through the network using the VS scheme. These control packets record the path traces and the link costs along the way. The selection of the single or multiple primary paths and back-up paths can be done at the destination. The MPLS signaling protocols can then be used to set-up the primary and back-up label-switched paths. The back-up paths may be triggered at the point of the path bifurcation if the primary path gets heavily congested. The efficacy of the VS based path selection, in terms of load balancing and congestion avoidance, is demonstrated by the simulation results in the next section.

III. Simulation Results

For simulation purposes arbitrary networks have been constructed using randomly chosen connectivity matrix and random node coordinates. The network is specified in terms of only the total number of nodes and the average connectivity per node. We have carried out simulations for randomly constructed networks with 25, 40, 80, and 200 nodes with average node connectivity of 4 and 6. An example of the transformation from a planar 25-node network topology with given connections into the 3-D VS configuration is shown in Fig. 1 from [2]. The figure shows how the nodes get rearranged to reflect the original network topology information in terms of the VS embedding. The nodes 21 and 4 that are originally located near get separated as per their least hop distance. Whereas, the nodes 15 and 16 that are directly connected remain close by and are separated by a distance of approximately one unit. The directed VS distance between distant nodes tends to indicate the available paths with the least or low number of hops, *e.g.* the directed VS distance from the node 18 to node 5 indicates routing paths given by 18–19–14–24–6–5 and 18–10–9–8–7–25–5.

Simulation results of VS based packet routing in randomly constructed 100 and 200-node networks with average node connectivity of 4 and 6 are presented in [2]. With properly evolved VS configuration with dimensionality of 5 or higher and using the 1-hop, 2-hop termination schemes the packet looping probability can be virtually eliminated. The packet throughputs of 99.9% have been observed in the simulations consistently. The VS routing also shows high packet throughputs under arbitrary link failures to give a self-healing property to the network. The studies presented in [2] demonstrate over 99.5% and 99% throughputs under 5% and 10% random link breakages for 100-node networks.

Traffic engineering simulations are performed on several randomly constructed 200-node networks and the sample results are presented here for one such network with an average connectivity of 4. The connectivity of individual nodes varies between 2 to 8 with a large majority of them having the connectivity of 3, 4, and 5. The routing is based on the 8-dimensional VS configuration that is constructed as per the steps outlined in section II. Out of the total 39800 source-destination pairs only 1600 pairs are selected randomly to represent active communication channels. A demand matrix is generated by randomly selecting a data rate for each pair from 5, 10, 15, 20, and 25 packets/iteration.

For the first iteration all the links are kept at a fixed cost level of 0.4 to generate the initial traffic pattern. The link loading is calculated in terms of number of packets traversing that link in each iteration. The distribution of links versus the loading is plotted as a histogram after each iteration and the results are shown in Fig. 2(a). After the first iteration, the distribution has a wide spread and a large tail on the high loading side. For several such random initial settings, the traffic pattern after the first iteration had a very similar structure with the average link usage, L_{avg} in (2), of about 320 and the standard deviation of about 127. At every successive iteration the link costs are defined based the link usage (L_{ij}) in the previous iteration. The scaling factor, Δ_L in (2), is fixed to 127, *i.e.* the standard deviation after the initial iteration.

As the links with higher usage have higher costs, they become less attractive for further traffic flow and the vice versa. Thus, the VS-based routing naturally leads to load balancing, so that link usage pattern moves closer towards the

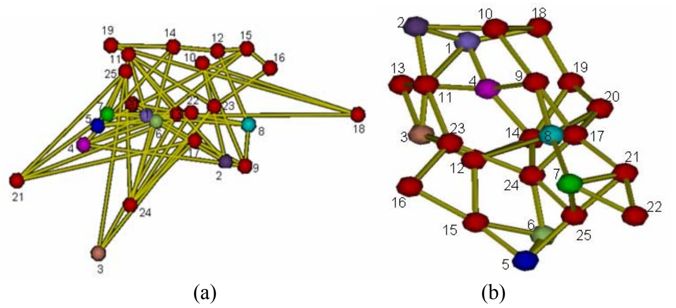
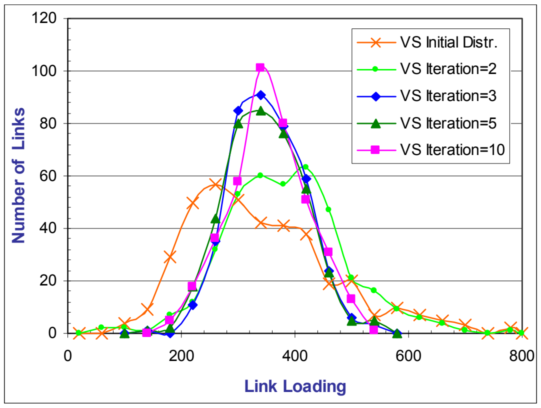


Figure 1. 25-node network topology showing (a) the original planar network and (b) the final 3-D virtual space configuration [2]

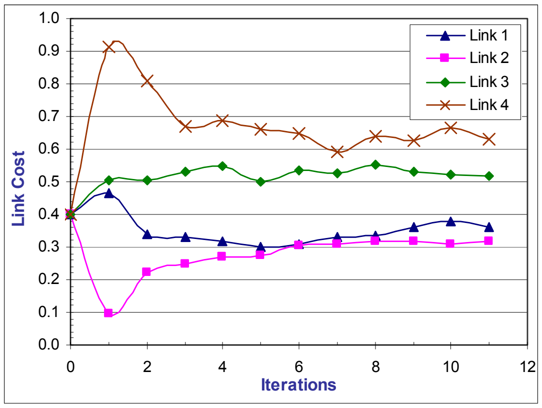
mean loading level. As the load balancing takes effect the heavy loading tail in the distribution is suppressed and the center peaks up in the successive iterations. This phenomenon is clearly demonstrated by the simulation results presented in Fig. 2(a). The traffic pattern stabilizes after about 4 to 5 iterations and the final pattern is plotted after 10 iterations. The standard deviation of the link usage distribution shrinks from 127 after iteration-1 to about 68 after iteration-10. Thus, close to a 50% reduction in the link usage variation has been achieved. The average link usage moves up only slightly from a 320 level to 330, which implies that there has been no performance penalty incurred in the load balancing process. Similar types of results are observed under different initial settings.

The variations of link costs for 4 sample links with iterations are plotted in Fig. 2(b). It shows that starting from 0.4 the costs spread apart (variation from 0.1 to 0.9) after the first iteration due to diverse loading levels, but the costs are brought close to the middle range (within 0.3 to 0.65) by load balancing. The link costs stabilize after 4-5 iterations, just like the link loading distribution. If the link cost variation is not damped based on (3), then certain links may exhibit load oscillations due to route flaps between the successive iterations. However, the overall link load distribution may not indicate these load oscillations. As mentioned before, each iteration corresponds to one link cost update interval in the VS protocol. Thus, the load distribution adjusts to any arbitrary demand pattern within 4 to 5 cost update intervals, *i.e.* the network can stabilize to a new load balanced traffic pattern in 100 to 500 ms depending upon the update interval. Since the link cost information does not flood the network nor does it lead to any cascaded updates, the update interval can be made arbitrarily small. This is a very useful property under dynamic load conditions or under bursty traffic patterns.

One important bench-mark for VS routing is its comparison with the OSPF routing. Simulations have been carried out using the OSPF routing for the same 200-node network under the same demand matrix. The optimal link weights calculation for the OSPF-TE is found to be not scalable for a 200-node network, so the comparison is not shown here. The comparison of the final VS and OSPF link loading distributions is shown in Fig. 3. The OSPF distribution has a heavy tail on the high link loading (>500) side indicating high level of traffic congestion. Whereas, the VS distribution shows a very low spread that indicates excellent load balancing.



(a)



(b)

Figure 2. Evolution of (a) VS link load distribution with iterations after 1, 2, 3, 5, & 10 iterations; and (b) VS link costs for 4 sample links; under an arbitrary demand matrix in a 200-node random network

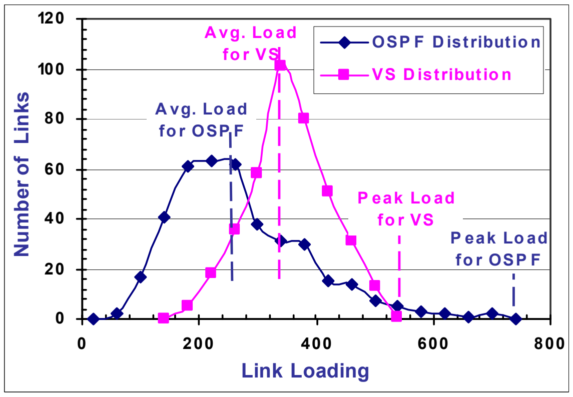


Figure 3. Link load distributions for the OSPF and VS routing under arbitrary demand matrix in a 200-node network

From Fig. 3, the OSPF distribution has the average link loading of 252 packets (for single iteration) that is about 25% lower than that of the VS routing. However, the RMS spread of the OSPF distribution is 115, which is about 70% higher. For fair comparison, we have carried out OSPF simulations with multi-path (more than one least-hop paths) routing just as the VS routing. If the single path OSPF routing is used then the average loading remains the same but the RMS spread increases even more. The average loading in OSPF indicates the lower bound for a given demand matrix and any scheme that improves the load balancing must have a higher value of average loading. For various 200-node network simulations the packet throughput remained at a level of 99.9%, which indicates very efficient VS routing with almost no looping in the network. If the VS configuration of lower dimensionality (6-D or 7-D) is used, the packet throughput remains at the same level except the average link loading goes up by 5–10% indicating slightly less efficient route selection.

The figure shows that the peak in OSPF loading extends about 35 to 40% higher than the VS peak. About 4 to 8% of the links in the heavy tail suffer from excessive loading and possible congestion. This congestion brings down the performance of the entire network in terms of lower throughput, available bandwidth, and quality. If this heavy tail is eliminated the network utilization levels can be increased by 25–30% without suffering any loss in its performance. The static link-weight assignment schemes (*i.e.* OSPF-TE) attempt to address the same problem. The VS routing scheme achieves this task of load balancing and provides solution that is fully dynamic and scalable.

Simulations are carried out for randomly constructed 25, 40, 80, and 400-node networks and the results from the VS routing and OSPF routing are very similar to the 200-node network presented in (3). As an example, 80-node network is simulated with 400 arbitrary source-destination pairs and with variable data rates of 5, 10, 15, 20, and 25 packets/iteration. The average and RMS spreads in the loading levels are 124 and 65 for the OSPF and 155 and 34 for the VS. Thus, the average OSPF link loading is about 20% less compared to the VS average, but the RMS spread is 90% higher. The peak value of the OSPF loading is again 30% higher, which again validates the congestion avoidance property of the VS scheme.

Although, these network topologies are randomly constructed, the realistic networks may have biases, which may alter their network performance. The practical networks tend to have higher probability of short-distance links (nodes that are physically located near by), which leads to clustering of nodes. A 90-node BTnet backbone network as seen from their website [16], has been used to carry out the VS and OSPF routing simulations. The average connectivity of each node in the network is about 5. As points of deviation, these simulations do not use the actual traffic demand matrix as would be seen on the BTnet and the link capacity on all links is assumed to be the same. Different link capacity values can easily be added by introducing a scaling factor in the link cost function, as described in section II. The demand matrix is created by choosing 200 source-destination pairs at random and the data rates are fixed randomly between 5, 10, 15, 20, and 25 packets/iteration. The link load distributions for the VS and OSPF routing schemes are plotted in Fig. 5. The average loading for OSPF and VS routing is 65 and 85, respectively.

One striking fact about the distributions in Fig. 4 is that, their characteristics are quite different from those for the 200-node random networks. The spread in the link loading levels for the OSPF routing is very high (at 79) for this network, in fact the RMS spread is higher than the average itself. If the high usage level links match with the higher link capacities, then the spread would be brought down and that would be valid for both the routing schemes. In spite of the differences in the nature of the distributions, the conclusions drawn from the comparison of the VS and OSPF routing schemes remain the

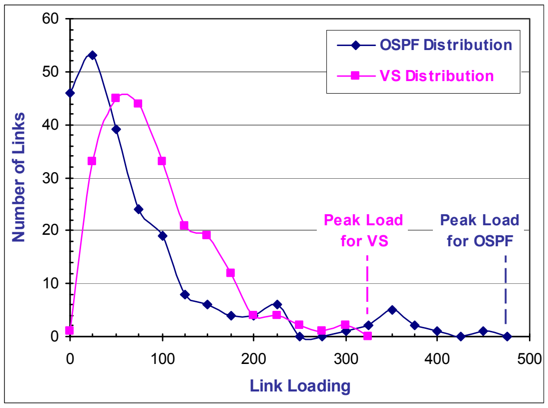


Figure 4. Link load distributions for the OSPF and VS routing for BTnet backbone network [17] for arbitrary demand matrix

same. The load balancing and congestion avoidance property of the VS routing is even more enhanced in these simulations. The peak loading level of the OSPF routing exceeds that of the VS routing by nearly 50%.

IV. Conclusions and Discussion

We have presented a novel traffic engineering technique using the new VS-based routing scheme. It uses virtual space embedding to express the network topology information in a very concise form. The dynamic link loading or failure information is expressed in terms of link and node costs and distributed to only the neighboring nodes. The final route selection is done by taking into account the load conditions along with the VS configuration. Since the need for broadcasting the link state information to distant nodes and the subsequent cascaded updates are eliminated, the VS routing avoids the flurry of information exchange under dynamic conditions. This makes the VS scheme a very stable, flexible, and highly scalable solution for traffic engineering. It is proposed that this technique be used for intra-domain path selection in MPLS networks.

The VS scheme takes into account the dynamic link costs which alleviate traffic congestion and achieve excellent load balancing. The simulation results for randomly constructed 80 and 200-node networks show that the VS-based traffic pattern adjusts to any arbitrary demand matrix in only 4 to 5 iterations of link cost updates to achieve excellent load balancing. The cost update interval can be made 25 ms or smaller to give fully dynamic adaptability to cater to any bursty or real-time traffic patterns. We are currently working on modeling the VS based routing using the Network Simulator (NS) to study the packet throughput, link failure, and buffer utilization.

As a bench-mark test, the link load distributions from the VS and OSPF routing schemes for the same networks are compared. The simulations results for randomly constructed 80 and 200-node networks show that the OSPF distribution has the average link loading that is 20–25% lower than that for the VS routing. However, the RMS spread of the OSPF distribution is 70–90% higher. The peak value of the OSPF loading is 30–40% higher, which validates the congestion avoidance property of the VS scheme. This property would improve the overall performance of the network in terms of higher packet throughput, lower loss rates, and stable time delay. The simulations are also carried out on the BTnet backbone network and the results demonstrate excellent load balancing and congestion avoidance properties of the VS routing scheme on the realistic networks.

The VS-based routing can be used as a stand-alone routing scheme. However, it can also be used along with the existing networking protocols (ATM and MPLS) as a control plane tool for traffic engineering. The VS-based routing can be used for path selection and integrated with the other signaling protocols to establish label-switched paths. A specific quality of service (QoS) request may be supported using the resource reservation protocols (RSVP-TE). As a possible implementation the path selection can be done by sending multiple path-request datagrams that keep track of their paths along with the associated costs and are forwarded using the VS routing. The destination node can select the primary and back-up paths or multiple primary paths for establishing the label-switched channels. As an alternative, all the path selections can be done at one central node using the VS routing, provided that all the dynamic link costs are available to the node.

In conclusion, the novel traffic engineering scheme based on the VS-routing appears to be very promising.

[1] J. Moy, "OSPF version 2," RFC 2328, IETF, Apr. 1998.

[2] G. P. Saraph and P. Singh, "New scheme for IP routing and traffic engineering," *Proceedings of HPSR 2003*, Torino, Italy, June 2003.

[3] J. Ash, M.Girish, E. Gray, *et al.*, "Applicability statement of CR-LDP," RFC 3213, *http://www.faqs.org/rfcs/rfc3213.html*, Jan. 2002.

[4] D. Awduche, L. Berger, *et al.*, "RSVP-TE: Extensions to RSVP for LSP tunnels," RFC 3209, *http://www.faqs.org/rfcs/rfc3209.html*, Dec. 2001.

[5] S. Keshav, "An Engineering Approach to Computer Networking," Addison Wesley Publ., Reading, MA, U.S.A., 1997.

[6] BGP Table of Telstra (Online): http://bgp.potaroo.net, Dec. 2003.

[7] B. Davie and Y. Rekhter, "MPLS Technology and Applications," Morgan Kaufman Publ., San Francisco, CA, U.S.A., 2000.

[8] A. Basu and J. G. Riecke, "Stability issues in OSPF routing," *Proc. of ACM SIGCOMM'01*, Aug. 2001, pp. 225-236.

[9] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," *Proc. 19th IEEE Conf. on Computer Communications (INFOCOM)*, 2000, pp. 519-528.

[10] Cisco [Online], "Configuring OSPF," *http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/np1_c/1cprt1/1cospf.htm*

[11] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," *Proc. of ACM SIGCOMM'99*, Sept. 1999.

[12] S. Vutukury and J.Garcia-Luna-Aceves, "A traffic engineering approach based on minimum-delay routing," *Proc. IEEE IC3N 2000*, Oct. 2000.

[13] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Select. Areas Commun.*, vol. 20, pp. 756-767, May 2002.

[14] A. Feldmann, A. Greenberg, *et.al.*, "Deriving traffic demands from operational IP networks: Methodology and experience," *IEEE/ACM Trans. Networking*, 2001, pp. 265-279.

[15] A. Feldmann, A. Greenberg, *et.al.*, "NetScope: Traffic Engineering for IP Netwprks," *IEEE Network Mag.*, Vol. 14, Mar./Apr. 2000, pp. 11-19.

[16] [Online]: http://www.ignite.com/internetservices/BTnet/network_backbone.htm, Mar. 2003.