Robust Load Balancing using Multi-Topology Routing

Amund Kvalbein and Olav Lysne

Simula Research Laboratory, Oslo, Norway

Abstract Current methods for traffic engineering with traditional link state routing protocols like OSPF and IS-IS are based on optimizing link weights to fit a given estimate of the traffic demands. These methods are not good at handling natural changes in the traffic over time. In this paper, we introduce new methods for IGP load balancing based on the upcoming standard for Multi-Topology routing. The main advantage of our approach is that it is far more robust to changes in the traffic demands. Our initial evaluations indicate that our method significantly reduces the chances of losing packets due to congestion.

1 Introduction

In connectionless intradomain routing protocols like OSPF or IS-IS, traffic engineering is done by carefully tuning the link weights that decide the shortest paths from each ingress to each egress node. Based on the network topology and the projected traffic demands, the link weights are set so as to minimize the cost of routing the demands through the network. The performance of such methods have been shown to be close to what can be achieved using connection oriented protocols like MPLS [1,2]. The main problem with traffic engineering approaches based on optimizing link weights is that they rely heavily on the available estimate of the traffic demands. These estimates can be based on traffic measurements and projections of customers needs. However, the demands vary significantly over time, and it is difficult to get an accurate network-wide view of the situation [3].

With a changed traffic matrix, we would like to run the optimization heuristic again, and install the new optimized link weights in the network to maintain the desired load balancing properties. However, changing link weights in an operational network is a bad thing. Not only does it lead to a period of routing instabilities as the routing protocol converges on the new topology [4], but it may also change the egress routers that are chosen in the BGP route-selection process, causing additional unwanted traffic shifts [5].

Several proposals have been made to mitigate the effects of traffic demand changes. In [6], a method is described that adapts the routing to the new traffic demands with as few weight changes as possible. This reduces the consequences, but it does not remove the problem. Other schemes try to find a link weight setting that performs well also in the presence of a link failure [7–9]. These proposals prepare for changes to the routing caused by failures, but do not handle natural changes in the traffic matrix caused by shift in user demands.

In this paper, we propose a new method for IGP traffic engineering that avoids the problems associated with link weight changes. Our method is based on Multi-Topology (MT) routing, which is currently being defined by the IETF [10,11]. MT routing allows the routers to maintain several independent logical topologies, with independent link weights, and hence independent routing, in each topology.

The main idea in our contribution is to construct the set of logical topologies in such a way that any congested link can be avoided in at least one topology. Traffic is then spread among the topologies in a way that gives good load balancing. We explore two different ways of utilizing this; one global method where ingress-egress flows are mapped to a topology at the ingress node, and one local method where traffic is dynamically moved to an alternate topology by the node experiencing congestion.

By looking at authentic traffic demands from the pan-European GEANT network, we show that the day to day variations of ingress-egress flows are significant. We then evaluate our two methods using simulations of this network, and compare the results to a well known method for traffic engineering based on IGP weight tuning [1]. We find that our local method significantly reduces the chances of packet loss in our simulated scenarios, while the global method performs as good as the weight tuning heuristic with a much simpler and more dynamic algorithm.

The rest of this paper is organized as follows. In Sec. 2 we illustrate that the temporal variations in traffic demands are indeed large. We then introduce

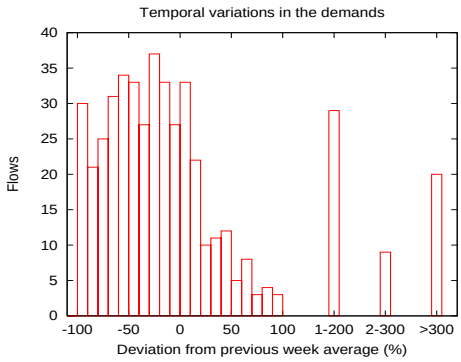

Fig. 1 Flow deviations from previous week average

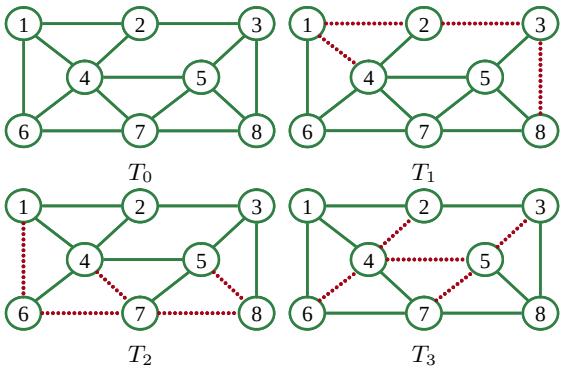

Fig. 2 Example topologies

MT routing and describe our algorithm for calculating logical topologies in Sec. 3. In Sec. 4 we explain the details of our global and local methods, before we evaluate our methods and compare them to an existing traffic engineering method in Sec. 5. Finally in Sec. 6, we conclude.

2 Temporal variations in backbone traffic

Traffic demands vary in both daily and weekly patterns, but they also show significant stochastic day-to-day variations. In this work, we look at real intradomain traffic matrixes from the GEANT network [12]. GEANT is a high capacity network connecting the national research networks in most European countries[1]. The network consists of 23 nodes connected by 37 bidirectional links. Most of the links have capacities ranging from 2.4 Gbps to 10 Gbps, with a few links with lower capacities from 155 Mbps to 1.5 Gbps. The traffic matrixes in our dataset tell us how much data was sent from each ingress node to each egress node in every 15 minutes interval over a four month period.

The demands in the GEANT traffic trace typically have their daily peak around lunch hours. To illustrate the stochastic variations in the demands, we have computed a traffic matrix that consists of the average peak hour demands in the GEANT network over 7 consecutive days. We then look at the peak hour demands on the 8th day (a Friday), and measure how much each ingress-egress flow deviates from the previous week average.

The results are shown in Fig. 1. We see that the variations are significant. For this particular day, about 7% of the flows were reduced by more than 90% compared to the previous week average. Over 13% of the demand flows were more than doubled, and almost 5% more than 4 times as large. This illustrates how difficult it is to find a single demand matrix that can be used as input to a load balancing heuristic based on link weight manipulation.

[1] A map of the GEANT topology is publicly available at http://www.geant.net

3 Multi-Topology routing

Multi-Topology routing allows the routers in an AS to maintain several logical views of the network topology. The routers exchange topology-specific link state advertisements describing the properties of each link. Conceptually, the routers build a separate routing table for each topology. Data traffic is associated with a specific topology, and is routed according to the corresponding routing table. This association might be in the form of a topology identifier in the packet header, or by using tunnelling in combination with a private address space [13].

3.1 Building logical topologies

We will now describe an algorithm for creating the alternate topologies. The algorithm is taken from [14], where we proposed the use of multiple topologies for fast recovery from link failures. Here, we only describe the rules that control the topology creation.

We define a topology T as a set of nodes N and a set of links L. Given a network topology T_0, we build n additional topologies $T_1, \ldots, T_n$. We refer to T_0 as the *original* topology, and $T_1, \ldots, T_n$ as the *alternate* topologies. The alternate topologies are copies of the original topology T_0, with the difference that a subset of the links are removed in each alternate topology. Importantly, links are removed in such a way that each alternate topology $T_1, \ldots, T_n$ is still connected, and thus all nodes are still reachable in all topologies. Figure 2 shows an example of how three alternate topologies can be built so that all links are removed in one of the topologies.

Input to our algorithm is the original topology T_0, and the number n of desired alternate topologies. The algorithm then iterates through all links and tries to

remove each of them in one of the topologies T_i. A link can only be removed from a topology if doing so does not disconnect the topology. If a link cannot be removed in T_i, we try again in topology $T_{(imodn)+1}$ until all alternate topologies have been tried. For each link we want to remove, a new topology is chosen as the first T_i we try, so that the number of removed links is approximately equal over the different T_i.

The algorithm results in n alternate topologies $T_1, \ldots, T_n$ with two important properties.

1. All topologies are connected, so that in each topology, there is a valid routing path between each pair of nodes.
2. All links are removed in exactly one of the alternate topologies. We denote by $T(l)$ the topology where link l is removed.

We have previously shown that a surprisingly small number of alternate topologies are needed to fulfill these two properties [14]. This way of generating the alternate topologies ensures that when a link becomes overloaded, there will always exist a topology where the congested link is not used to forward traffic.

4 Load balancing using Multi-Topology routing

We describe two different methods for load balancing based on MT routing. We refer to the two as the *global* and the *local* methods respectively.

4.1 Global load balancing

The global method takes as input the network graph, the link capacities, and the available traffic matrix estimate. Based on this, it assigns traffic flows to topologies in a way that tries to minimize the chances of congestion, by minimizing the maximum link utilization in the network. Note that we use the notion of a flow to describe the aggregate of traffic going from an ingress to an egress, and we to not distinguish traffic on a finer granularity. We foresee the global method being used in a centralized offline tool that calculates the consequences of different routings.

The global method starts out with routing all traffic demands according to the original topology T_0, and then moves some selected flows over to the alternate topologies in order to minimize the utilization of the most loaded link in the network. The load distribution in the network in the initial stage of the algorithm is determined by the link weights used in T_0. Our method is agnostic to the setting of these weights.

Pseudo-code for the algorithm that assigns ingress-egress flows to alternate topologies is given in Alg. 1. Each iteration in the outer while loop starts by identifying the most utilized link l_{max} in the network, and the flows $\mathcal{F}$ that are routed over this link. The inner for loop then iterates over $\mathcal{F}$ to evaluate the consequences of moving each flow. The flow that gives the lowest new max utilization is moved from T_0 to the topology $T(l_{max})$ where l_{max} is avoided. The algorithm terminates when no flow is found that gives a lower max utilization. The output of the algorithm is a mapping that is used by the ingress nodes to assign each flow to the correct topology.

Algorithm 1: Topology assignment algorithm

```
1  continue ⇐ true
2  while continue do
3      l_max ⇐ most loaded link in the network
4      u_max ⇐ utilization of l_max
5      F ⇐ set of flows routed through l_max in
       T_0
6      u' ⇐ ∞
7      forall f ∈ F do
8          Move f to T(l_max)
9          u_f ⇐ new max utilization in the
           network
10         if u_f < u' then
11             u' ⇐ u_f
12             f' ⇐ f
13         end
14         Move f to T_0
15     end
16     if u' < u_max then
17         Move f' to T(l_max)
18     else
19         continue ⇐ false
20     end
21 end
```

Since we operate only on coarse ingress-egress flows, the number of flows routed over any single link will be modest. Hence, we evaluate the consequences of moving each flow to $T(l_{max})$ before deciding which flow is actually moved. If we wanted to decrease the running time of our algorithm, this could be done by basing this selection on randomness or the size of the flows instead of testing for each flow.

4.1.1 Responding to changing traffic demands Network operators use various tools to monitor the state of their network, and to detect changes in the traffic demands [15]. When such changes are detected, we can use the above algorithm to calculate a new mapping from flows to topology. A key point here is that only this mapping needs to be changed to respond to changes in the demands. This gives two important

advantages over traffic engineering methods based on link weight tuning. First, the calculation of this mapping is fairly simple and takes considerably shorter time than a weight search heuristic. Second, and most important, changing the mapping does not trigger a new IGP convergence in the network, and it is transparent to the BGP route selection process.

Because our algorithm has low complexity and does not introduce instability in the routing, we argue that this recalculation can be done quite frequently. We believe it can be done on a time scale of hours, or perhaps even minutes, depending on how fast changes in the traffic demands can be measured.

4.2 Local load balancing

With the local method, no prior global calculations or setup is needed, except the construction of the alternate topologies. Instead, traffic is moved to an alternate topology that avoids the congested link by the upstream node local to the congestion. The local method does not require any knowledge of the traffic matrix, and the method responds immediately to unexpected bursts or changes in demands.

We propose a simple mechanism inspired by Random Early Detection [16], where packets are moved to the topology where the congested link is avoided with a probability that increases as the buffer occupancy gets closer to 100%. When there is no congestion in the network, all traffic is routed according to the original topology T_0. Packets are moved to the alternative topology by an intermediate node with a probability p that is 0 if the buffer occupancy stays below a certain threshold t_{buf}, and then increases linearly to 1 when the buffer is full. To avoid looping, we never let a packet chance topology more than once.

This local moving of traffic from one topology to another increases the chances of packet reordering, with its adverse effects on TCP performance. Packet reordering can be reduced by a mechanism that tries to select packets from the same TCP flow when deciding what traffic should be moved to another topology, much like what is done today when traffic is split between several equal cost paths using hashing functions.

4.3 Discussion of the global and local methods

While the global method tries to *prevent* congestion by assigning flows to topologies that avoid the most heavily loaded links, the local method instead *resolves* congestion by routing the traffic on an alternative path from the point of congestion.

The difference between the two methods is clearly visible in Fig. 3. The figure shows the simulated load

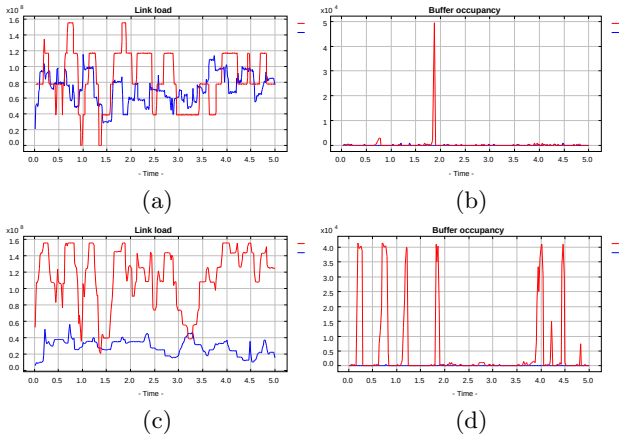

Fig. 3 Link load and buffer occupancy of the two links into a node with the global (a,b) and the local (c,d) method

and the corresponding buffer occupancy over a 5 seconds period for the two 155 Mbps links leading to a node n in the GEANT network (further description of the simulation environment and setup will be given in Sec. 5). Figure 3a) shows how the global method distributes the traffic towards the node relatively evenly between the two links. For most of the time, the link utilization is kept below the capacity, and the buffer occupancy stays close to 0, as seen in Fig. 3b). With the local method, the load on one of the incoming links is far greater than on the other, as seen in Fig. 3c). This results in a higher variation in the buffer occupancy (Fig. 3d). When the buffer occupancy approaches the capacity, the local method responds by moving traffic over to the topology that avoids the congested link. This traffic will follow the shortest path in the alternate topology. Traffic destined for node n will finally finds its way through the other link attached to n, where we can observe it as small spikes.

The main advantage of the local method is that no knowledge of the traffic demand is needed, and that it immediately adapts to shifts and short term fluctuations in the traffic. The price to pay for this flexibility is higher variation in buffer occupancy than with the global method, which might affect the delay experienced by the packets.

5 Evaluation

We have evaluated our proposed methods using simulations of the pan-European GEANT network, with real intradomain traffic matrixes [12]. We conduct simulations on the packet level, using the J-sim simulation framework [17]. Network traffic has been shown to have significant short term variations,

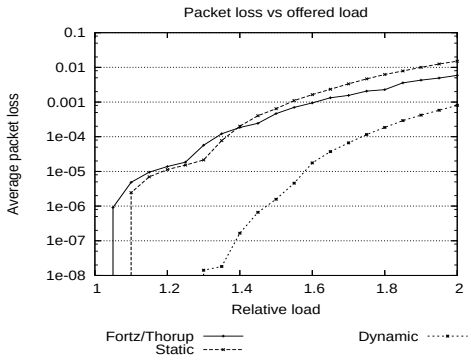

Fig. 4 Packet loss with increasing load

with self-similar characteristics [18]. To capture these effects, we let each ingress-egress flow be modelled as a self-similar packet source at the ingress node, producing the desired average load to the egress. Our self-similar packet sources are based on multiplexing several heavy-tailed ON/OFF sources, and are set to produce traffic with Hurst parameter 0.9. This traffic setup results in large short term variations in link utilization, as seen in Fig. 3. We use relatively small buffers (50 kBytes) in the routers. Combined with the large short term variations in link utilization, this setup gives a relatively high change of packet drops.

We use destination-based shortest path hop-by-hop forwarding, but the paths taken by packets will depend on which logical topology the packet belongs to. We use $n = 5$ alternate topologies, and we use original GEANT IGP link weights in all topologies $T_0, \ldots, T_n$. These link weights are mainly based on the inverse of the link capacities, with some manual tuning [12]. For our local method, we use a buffer threshold t_{buf} of 0.75.

To evaluate the performance of our methods, we compare them to the load balancing method proposed by Fortz and Thorup [1]. They define a cost function that penalizes routing traffic over heavily utilized links, and use a heuristic to tune the link weights in order to minimize this cost function. We will refer to this method as FT.

5.1 Robustness to increase in demand

Figure 4 shows the packet loss with our global and local methods, and with the FT heuristic. The simulations are based on a single 15-minutes traffic matrix captured between 12:45 and 13:00 from our dataset. The offered load is increased along the x-axis by multiplying all the flows with a scaling factor. The plotted values are average values from 20 simulations with different seeds.

We see how the local method gives substantially less packet loss than the other methods, the improvement is of a factor 10-100 compared to the FT heuristic. The global method performs slightly better than FT when the load factor is relatively low, and slightly worse for higher loads. This effect is seen because our global method only takes the load on the most loaded link into account, while the FT heuristic also looks at other highly loaded links. Hence the global method performs better as long as packet loss is confined to a single link.

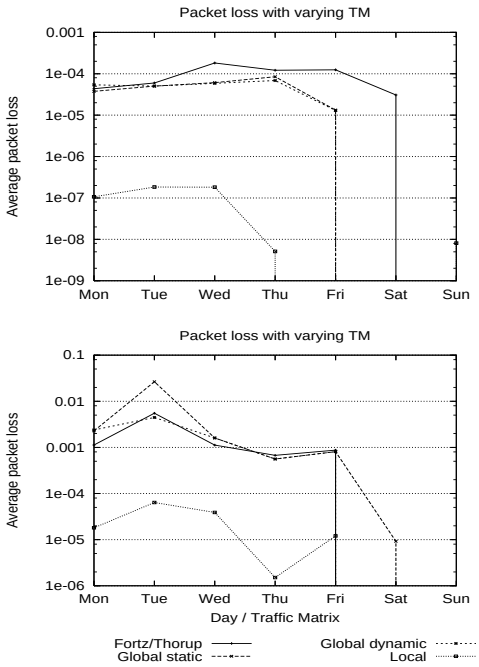

Fig. 5 Packet loss with demands that change over time.

The results show significant day-to-day variations, but due to space limitations we can only show results for a single demand matrix. Some days FT never overtakes the global method even for a high load factor. However, the tendencies that the local method performs significantly better than the other methods, and that the FT method has higher packet loss than the global method at low loads seem robust.

5.2 Robustness to changes in demand over time

We test how the different load balancing methods handle the day to day variations described in Sec. 2. Figure 5 shows the packet loss for 7 consecutive days. The FT method and our (static) global method are optimized for the average demands from the previous week. The dynamic global method is optimized for each day. In order to stress the network and get interesting results, the original GEANT traffic demands are multiplied with a factor 1.3. Again, the results shown are averages over 20 simulations with different seeds.

The traffic demands, and hence also the packet loss, varies significantly between weeks. Therefore, we plot results from two different weeks. Figure 5a) shows a week with relatively low demands, while Fig. 5b) shows a week with higher demands. Again, the main tendency is the same as in Fig. 4; the local method gives significantly less packet loss than the other methods. Also, the performance of the global methods relative to the FT method is better when the network is not severely overloaded. A final point to notice is the reduced traffic in the weekends (days 6 and 7), when we see almost no packet loss.

6 Conclusion

In this work, we have argued that current traffic engineering methods based on link weight tuning are not flexible enough when facing variations in the traffic demands. Using real traffic demands from the GEANT network, we have demonstrated that such variations are indeed significant.

We have introduced the use of Multi-Topology routing to improve the load balancing in intradomain IP networks. This is a much more flexible approach than link weigh tuning, since it avoids the IGP reconvergence and the adverse traffic effects that are triggered by link weight changes. We have described two different mechanisms based on using MT routing. The *global* method uses the available estimates of the traffic demands to assign ingress-egress flows to different topologies, in order to avoid heavily loaded links. The *local* method needs no prior knowledge of the traffic demands, and responds to congestion by locally moving traffic to an alternate topology that avoids the congested link. Our evaluations show that our global method performs as good as previous methods based on link weight tuning with respect to packet loss, while our local method performs substantially better.

In our opinion, this work adds yet another argument for introducing a mechanism in the MT routing standards that allows packets to change from one topology to another in flight.

Acknowledgements: We would like to thank Steve Uhlig for providing help with the traffic matrixes for the GEANT network.

References

1. Fortz, B., Thorup, M.: Internet traffic engineering by optimizing OSPF weights. In: Proceedings INFOCOM. (2000) 519–528
2. Sridharan, A., Guirin, R., Diot, C.: Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. IEEE/ACM Transactions on Networking **13**(2) (2005) 234–247
3. Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., True, F.: Deriving traffic demands for operational IP networks: methodology and experience. IEEE/ACM Transactions on Networking **9**(3) (2001) 265–280
4. Basu, A., Riecke, J.G.: Stability Issues in OSPF Routing. In: Proceedings of SIGCOMM. (2001) 225–236
5. Teixeira, R., Shaikh, A., Griffin, T., Voelker, G.M.: Network sensitivity to hot-potato disruptions. In: Proceedings of SIGCOMM, Portland, Oregon, USA (2004) 231–244
6. Fortz, B., Thorup, M.: Optimizing OSPF/IS-IS weights in a changing world. IEEE Journal on Selected Areas in Communications **20**(4) (2002) 756 – 767
7. Nucci, A., Schroeder, B., Bhattacharyya, S., Taft, N., Diot, C.: IGP Link Weight Assignment for Transient Link Failures. In: 18th International Teletraffic Congress, Berlin, Germany (2003)
8. Fortz, B., Thorup, M.: Robust optimization of OSPF/IS-IS weights. In: INOC. (2003) 225–230
9. Sridharan, A., Guerin, R.: Making IGP routing robust to link failures. In: Proceedings of Networking, Waterloo, Canada (2005)
10. Psenak, P., Mirtorabi, S., Roy, A., Nguen, L., Pillay-Esnault, P.: MT-OSPF: Multi topology (MT) routing in OSPF. IETF Internet Draft (work in progress) (2006) draft-ietf-ospf-mt-06.txt.
11. Przygienda, T., Shen, N., Sheth, N.: M-ISIS: Multi topology (MT) routing in IS-IS. Internet Draft (work in progress) (2005) draft-ietf-isis-wg-multi-topology-11.txt.
12. Uhlig, S., Quoitin, B., Lepropre, J., Balon, S.: Providing public intradomain traffic matrices to the research community. ACM SIGCOMM Computer Communication Review **36**(1) (2006) 83–86
13. Bryant, S., Shand, M., Previdi, S.: IP fast reroute using not-via addresses. Internet Draft (work in progress) (2005) draft-bryant-shand-IPFRR-notvia-addresses-01.txt.
14. Kvalbein, A., Hansen, A.F., Čičić, T., Gjessing, S., Lysne, O.: Fast recovery from link failures using resilient routing layers. In: Proceedings 10th IEEE Symposium on Computers and Communications (ISCC). (2005)
15. Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J.: Netscope: Traffic engineering for IP networks. IEEE Network Magazine **14**(2) (2000) 11–19
16. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking **1**(4) (1993) 397–413
17. Tyan, H.Y.: Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation. PhD thesis, Ohio State University (2002)
18. Paxson, V., Floyd, S.: Wide-area traffic: The failure of poisson modeling. IEEE/ACM Transactions on Networking **3**(3) (1995) 226–244