Link Weight Optimisation with Quality of Service Support

Jonas Griem

University College London

Abstract: This paper presents a new method to perform QoS enabled traffic engineering based mostly on conventional IP routing techniques. By combining the flexibility of layer 3 routing protocols like OSPF with a management plane link weight traffic engineering technique, this paper makes the proposition that more recent technologies such as MPLS are not essential to enable services that require QoS support.

1 Introduction

The success of multimedia applications on the Internet in the recent past clearly indicates that despite the Internet's principal design as a simple datagram service, performance sensitive applications run quite well by adapting to the variations in network conditions. Yet the calls for quality of service are justified, if the Internet is to replace, rather than complement any of the existing telephony and television services. In order to do so, routing algorithms have to become QoS aware either by reserving capacity on the network for each flow as proposed by (among others) the IntServ working group [1] or by arranging the aggregate traffic flows inside the network in such a way that statistical quality guarantees can be given for individual flows. The second option is potentially more scalable, but it requires several components: the DiffServ per-hop-behaviour (PHB) at the queue level as well as network management components that keep up to date information on traffic flows and available capacity via monitoring as well as optimise the traffic flow by performing traffic engineering. This paper outlines ideas and initial algorithms for an IP based traffic engineering mechanism.

2. Why IP Traffic Engineering

Shortest path routing algorithms such as used in the OSPF routing protocol have a significant limitation for performing traffic engineering: the choice of the route is essentially that of the algorithm, only indirect control over route selection is provided to the network operator via the link weight settings. There is no method to redirect a particular stream of packets to onto a different path, without potentially modifying the paths of many other streams of packets at the same time. The reason for this shortcoming is related to the distributed and autonomous nature of routing, forcing simple algorithms with low processing requirements and little information interchange. A lot of recent work has therefore focused on solving the problem using non-IP forwarding techniques such as Multiprotocol Label Switching (MPLS). With MPLS, label switched paths are explicitly installed for each traffic trunk (that is traffic on the same route requiring the same treatment). Each label switched router on the path needs to have a label translation table entry per such flow and routers without corresponding entries (typically routers outside the path) cannot forward packets belonging to the flow. Keeping explicit state information for each traffic trunk is a potential scalability problem of the MPLS technique. A more important shortcoming is the lack of flexibility in the MPLS solution compared to IP routing. Without routing software such as the shortest path algorithm in OSPF, MPLS solutions cannot recover from link or node failures or badly configured label switched paths without higher layer intervention (some work has been done in this area, e.g. [2]). The mechanisms provided by MPLS are thus useful, especially for small-scale intermediate solutions. However, for large scale Internet wide deployment, it would be preferable to keep the inherent advantages of routing and so MPLS-only solutions should be avoided unless they are extended with a layer 3 routing algorithm. In addition the scalability problems of MPLS have to be addressed.

The challenge to be met by an IP based traffic engineering system is twofold: meeting the traffic engineering (and QoS) requirements for traffic injected into the network and doing so without crippling the IP Interior Gateway Protocols (IP-IGPs) routing capabilities. In this paper, the proposition is made that this goal can be achieved. The algorithms proposed, are based upon the link weight optimisation techniques first described by [3]. The work shows that traffic engineering with near MPLS performance without QoS constraints can be achieved by manipulating the OSPF link weights using a heuristic search algorithm that balances the flows more efficiently. The link weight

optimisation is carried out offline (i.e. outside the layer 3 routing algorithm execution time frame) as a network management task and is based on real and estimated traffic matrices. The actual OSPF algorithm routing remains unmodified; all traffic engineering information is contained in the link weights. In addition to [3] the algorithms outlined in this paper have to meet the QoS constraints for individual flows, also without modifying the routing algorithms in the IGP (OSPF). As a modification to current IGP routing, this paper proposes to operate parallel *routing planes* for each DiffServ code point (DSCP), i.e. a maximum of 64 individual routing planes in order to diversify the traffic engineering choices. *Figure 1* outlines the process in which a small network has three independent sets of link weights (small numbers next to nodes). Three instances of OSFP calculate three different paths from the source *s* to destination *d* using the different sets of weights.

3. Traffic Engineering in the Management Plane

While applications have changed dramatically during the short existence of the Internet, network layer functionality has evolved very little. There are two reasons for this: the maturity of the technology (it works) and the inertia caused by the need to potentially update *every* router of the Internet. A point was made in [3] that functionality that is subject to frequent change, such as the link weight optimisation algorithms and policies controlling the algorithms, should not be located in the network layer because of this inertia. By contrast, if the functionality was located in the management plane it can be modified more easily to represent changes in business objectives as well as improvements in technology without the need for updates to router operating software, which is often outside the operators control. The approach outlined in this paper is a combination of conventional distributed routing and management based traffic engineering functionality. There are some good arguments for splitting the functionality in such a way. The most important advantage is that the approach bares the opportunity for graceful migration that causes little disruption to existing layer 3 operations. With the exception of the introduction of routing planes, most functionality is added at the management layer. Equally important is that the complexity introduced by QoS awareness remains outside the network layer, leaving the distributed best-effort Internet untouched. Moving the complexity caused by QoS awareness into the management plane also has the advantage of avoiding the large increase in state information in the network.

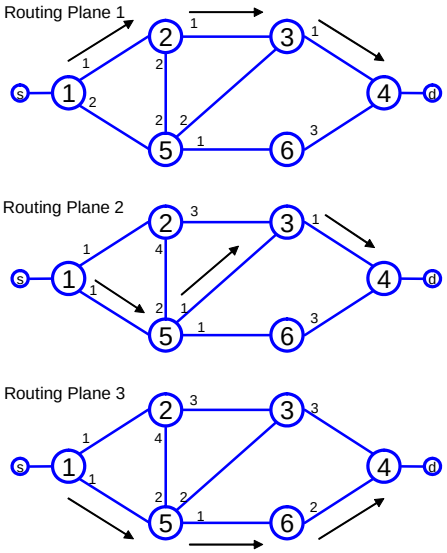

Figure 1 – "Routing Planes"

4. A Link Weight Modification Algorithm

Distributing traffic demands on OSPF networks is a challenge, because of the "indirectness" of the traffic engineering problem. Finding optimal paths for each traffic demand as for MPLS is only half of the problem, the other half is the implementation of these paths with OSPF link weights. While MPLS allows the explicit pinning of a route between any two nodes and effectively switches packets according to this configuration, OSPF relies on individual routing decisions taken at each node. These are based solely on destination IP address and the networks link weight metrics that determine the shortest path towards the destination address. Mapping an "optimal" demand distribution as calculated for an MPLS network onto an OSPF network is thus not an achievable goal in most practical cases. Instead, the algorithms for finding good paths and for translating these paths into link weights have to go hand in hand, iteratively searching for better link weights to spread the load across the network. *Figure 2* shows an activity diagram of a link weight traffic engineering algorithm [4].

4.1. Link Load

Before beginning the optimisation process, the network has to be initialised with a set of link weights. Preferably at this stage the weights should be chosen to provide a reasonable initial condition to the

optimisation. Inverse link weights, such as recommended by Cisco [5] are a better starting point than unit weights for instance, since inverse capacity weights reflect on the network topology. From the activity diagram in *figure 2,* the next step is the calculation of shortest path trees from each ingress/egress pair in the network that is expected to carry traffic. Using a matrix of predicted traffic patterns, the link load on each link inside the network can now be calculated. Note that the steps outlined have to be performed for each routing plane and in order to arrive at the total link load on each link, the results of the load calculations have to be summed. Quality characteristics for traffic on different routing planes are likely to be different and as a result, summation of the planes bandwidth consumption is not a straight forward process. A concept called *equivalent* bandwidth is employed to make the bandwidth comparable; however the details of this process and that of QoS parameter translation into traffic engineering parameters are out of scope of this discussion for reasons of space.

4.2 Link Cost

Once the utilisation of each link is known, an optimisation algorithm has to choose one (or some) links to optimise during the current iteration. This choice is made based on a cost function, which summarises the objectives of optimisation criteria. As in [6] the network is modelled as a directed graph, $G=(N,E)$ where the nodes $n \in N$ and links $l \in E$ represent routers and links between routers. A link l has capacity $c(l)$ indicating the amount of traffic that l can accommodate. The traffic is given in form of a demand matrix D, representing the amount of traffic flowing on a path between any nodes (s,d). This demand matrix is derived from projections based on historic data and current traffic demands. It can be expected that many of the demands (s,d) are zero, as not every ingress/egress pair has a non zero traffic flow. So the traffic engineering problem is to distribute the traffic from non zero $D(s,d)$ across the network evenly. The load on a link is given as x_l, this is the sum of all demands $D(s,d)$ using the link l. The utilisation of l is then given by $x_l/c(l)$. Link loads from each routing plane and the physical capacity of each link are used to give the total free capacity of each link. Each routing plane had different QoS characteristics, which has to be expressed by associating an *equivalent* bandwidth to PHBs rather than the physical bandwidth. For each PHB h of a set H_l of PHBs on a link with bandwidth allocation $x_{l,h}$, the equivalent bandwidth can be expressed as a function $f_{l,h}(x_{l,h})$ increasing in $x_{l,h}$ and greater for any given $x_{l,h}$ with higher priority h. The total equivalent load of each link is then $\sum_{h \in H_l} f_{l,h}(x_{l,h})$. Assuming that $c(l)$ is not the same for all links,

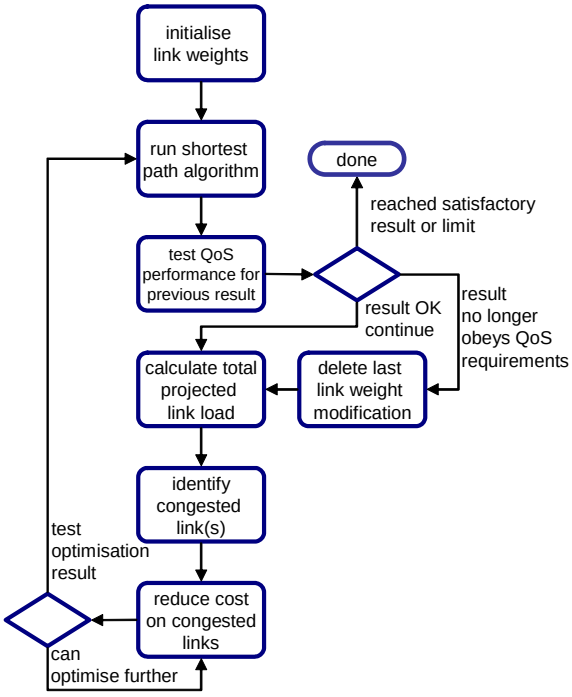

Figure 2 – Activity diagram outlining the stages of a link weight based traffic engineering algorithm.

$$L_e = \sum_{h \in H_l} f_{l,h}(x_{l,h}) \Big/ c(l)$$

is the normalised utilisation of the link. In order to arrive at an overall cost function, the statement has to be extended to reflect a cost per link, which can then be summed over $l \in E$.

$$\Phi = \sum_{l \in E} \Phi_l(L_e) = \sum_{l \in E} \Phi_l \left(\frac{\sum_{h \in H_l} (f_{l,h}(x_{l,h}))}{c(l)} \right)$$

4.3 Reducing the Load of Congested Links

In order to minimise the cost function, those links with the largest contribution to the total cost have to be identified. Hence $l_c = \max_{l \in E} \Phi_l(L_e)$ identifies such a link. Before modifying any weights, it is necessary to choose a candidate traffic flow passing though l_c which to modify. Depending on the type of modification and the amount of load, a choice of flows to modify has to be made. Parameters of the flows are bandwidth and routing plane membership. Since each routing plane has different QoS parameters, it is possible that flows of least one plane have sufficient leeway on relevant QoS parameters (available QoS parameter budget). Once a choice has been made, the flow is redirected:

Let R be the origin node of link l_c, and let R_l be the destination node of l_c. Let *W(R)* be the sum of the weights on the shortest path from node R to some egress node R_E. Also let R_n be a neighbouring node to R of a set of neighbours $R_n \in B$. Let $\omega_{x,y}$ be the link weight on link x for routing plane y. Then,

$$W(R) = W(R_l) + \omega_{l,h} \leq W(R_n) + \omega_{(R,R_n),h} \text{ for all } R \in B$$

The objective now is the reduction of the load on l_c, by redistribution of its load onto other neighbouring links. For a single weight modification, the neighbourhood of node R is searched in order to locate a neighbouring node R_n such that

$$W(R_n) + \omega_{(R,R_n),h} + \varphi < W(R_l) + \omega_h \text{, where } \varphi \text{ is a weight adjustment}$$

It is sensible to choose the neighbour where φ is the smallest value in all of B. The reason is that the adjustment of a weight may cause other routes on R_n to change. By choosing the smallest weight change, the probability of such unwanted changes is kept low.

5. Conclusions and Future Work

This paper has outlined some ideas and initial algorithms for performing traffic engineering for QoS enabled networks without the need for extensive modifications to existing routing and avoiding techniques like MPLS. The approach is based on a two layer traffic engineering with conventional IGP routing and centralised management plane functionality. It carries the advantages of both layer 3 and management plane traffic engineering techniques, avoiding complexity in the routing plane as well as leaving the best-effort Internet intact. Simulation is expected to take place within the next months to validate if the approach can rival MPLS based solutions, while staying within acceptable limits of computational complexity.

Acknowledgements

The work on IP traffic engineering is being carried out as part of the IST-MESCAL project. The author would especially like to thank David Griffin, Professor Chris Todd and Jason Spencer for their valuable input to the ideas presented here.

References

[1] R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, Request for Comments, IETF Network Working Group, 1994

[2] M.R. Meyer, et al., “MPLS Traffic Engineering Soft preemption”, Internet Draft, IETF, 2004

[3] Fortz, B., and M. Thorup, “Internet traffic engineering by optimizing OSPF weights”, paper presented at INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol.2, Iss., 2000

[4] M.Howarth, et al. “D2.1: Initial specification of protocols and algorithms”, IST MESCAL, 2004

[5] Open Shortest Path First, www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ospf.htm

[6] Awduche, D.O., J. Malcom, J. Agogbua, M. O'Dell, and J. McManus (eds.), “Requirements for traffic engineering over MPLS”, Request For Comments, IETF Network Working Group, 1999