Traffic Engineering Using OSPF Weights and Splitting Ratios

J. Murphy, R. Harris and R.Nelson*
CATT Centre, RMIT Universty,

**Department of Electrical Engineering, Monash University*

Abstract: A method is proposed to perform traffic engineering for Autonomous Systems by setting OSPF weights and distributing new metrics called node splitting ratios. The quality of the traffic engineering is comparable to that achievable using MPLS. The weights and splitting ratios can be calculated using linear programming techniques. This potentially allows the solution of very large problems of a scale often found in an ISP backbone. Furthermore changes to network topology or addition of new flows can be easily done without performing the whole optimisation procedure. The method we have presented here offers the ability to perform Traffic flow optimisation without maintaining any per flow state in routers and without introducing additional protocols or packet overhead, unlike MPLS which is commonly used for Traffic Engineering.

Key words: Traffic Engineering, Load Balancing, Network Optimization

1. INTRODUCTION

Many present day routers use the OSPF or IS-IS protocol [1] for directing Internet traffic. These protocols route traffic on the shortest path between an origin and destination pair as defined by a suitable distance metric. Since this in no way takes account of the set of traffic demands, many situations

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35673-0_28
C. McDonald (ed.), *Converged Networking*

occur in which the links on shortest path routes become congested while links on other paths remain relatively free [2]. For best effort traffic, this is not usually a problem. For traffic requiring guarantees on delay and jitter, however, congestion cannot be tolerated. One needs to somehow "spread" the traffic so that as many links as possible are moderately loaded – one needs to perform some form of constraint based routing (CBR). There are two main ways of doing this: by using Multi-Protocol Label Switching (MPLS) [2][3] or by extending existing protocols. New protocols work well but take time, energy, money, and standardization effort to implement. Extensions to existing protocols, on the other hand, are far easier to implement but in many cases only offer a "band aid" solution. Fortz and Thorup [4], however, have proposed a scheme to determine the set of OSPF weights that will almost optimally "load balance" a network. They used a computationally intensive local search technique to determine the set of weights. Despite this they managed to produce results for moderately sized networks (50-90 nodes) In contrast, in our approach we use an LP formulation to determine the set of OSPF weights and a set of splitting ratios. This has two huge advantages: linear programming is a mature field that has spawned a plethora of efficient techniques to solve large problems. This includes the development of "state of the art" proprietary software packages (eg. CPLEX) that allow one to use many additional sophisticated techniques such as the use of an advanced basis, network algorithms, and decomposition to dramatically increase the size of solvable problems. Secondly, and this may be equally important, using sensitivity analysis, if any changes are made to any of the input parameters one can quickly determine the new feasible optimal solution. This means if a link failure occurs or a small fraction of the set of demands changes it is not necessary to recalculate the whole solution Consequently update times for changes in traffic conditions will be dramatically reduced.

The disadvantage of our approach is that far more state information is needed (ie the splitting ratios) than the set of "optimal" weights generated by Fortz and Thorup's work. We show however that this additional computational overhead is still manageable.

Both of our approaches assumes the existence of a demand matrix of flows between origin-destination pairs. This information may be gleaned from measurement of the flows directly using such tools as NeTraMet [5] or Cisco Netflow [6] or could be based on knowledge of the subscriptions to a virtual leased line service [4]. In either case, no account will be taken of the traffic's burstiness. As the simplex algorithm is far too slow for on line or real time calculations anyway, this is not seen as major disadvantage.

2. PROBLEM FORMULATION

The routing problem can be formulated as a multi-commodity flow problem [7]. Consider a network consisting of N nodes and M directed arcs. We denote the flow of commodity k on arc (i, j) as x^k_{ij} and the cost of commodity k using arc (i, j) as c^k_{ij} .Our aim is to minimize the cost function z given by:

$$z = \sum_{k=1}^{K} \mathbf{c}^k \mathbf{x}^k \tag{0}$$

where $\mathbf{c}^k$ is the row vector of link costs, c^k_{ij} and $\mathbf{x}^k$ is the column vector of link flows, x^k_{ij} .
subject to

$$\mathbf{N}\mathbf{x}^k = \mathbf{b}^k \tag{1}$$

for all k= 1,K, and

$$\sum_{k=1}^{K} x_{ij}^k \leq u_{ij} \tag{2}$$

N is the node-arc incidence matrix, u_{ij} is the capacity of each arc (i, j) in the network and b is known as the right hand side vector. The set of constraints in (1) are known as the mass balance constraints. They are an expression of the conservation of flow as the sum of the elements b(i) in $\mathbf{b}^k$ must equal zero. In the above case, there are only 2 non-zero elements in $\mathbf{b}^k$, b(s) corresponding to the source of a flow and b(t), the destination. The constraints in (2) are known as the "bundle" constraints as they tie together all the flows by restricting the amount of flow on a particular link (i,j). If there were no bundle constraints, this problem would revert to K single commodity flow problems. The commodity in our case is the flow from a source node to a destination node. The number of constraints in this formulation is NK. We can, however, make a transformation to the path flow formulation by making the substitution

$$x^k_{ij} = \sum_{P \in P^k} \delta_{ij}(P) f(P) \tag{3}$$

For each commodity k, $\mathbf{P^k}$ denotes the collection of all directed paths from the origin node to the destination node, f(P) is the flow on one of these paths, and $\delta_{ij}(P)$ equals one if arc (i,j) is contained in the path P and is zero otherwise. This transformation results in eqns. (0) –(2) becoming

$$\text{Minimize} \sum_{1\le k\le K} \sum_{P\in P^k} c^k(P)f(P) \tag{4}$$

subject to

$$\sum_{1\le k\le K} \sum_{P\in P^k} \delta_{ij}(P)f(P) \le u_{ij} \tag{5}$$

for all links (i,j) and

$$\sum_{P\in P^k} f(P) = d^k \tag{6}$$

for all k= 1, K and $f(P) \ge 0$ for all k = 1.....K In a sparse network, which most telecommunications networks are, the path flow formulation reduces the number of constraints by a factor of approximately N. In contrast, it increases the number of path flow variables enormously but few of these paths carry flow in the optimal solution.

3. THE COMPLEMENTARY SLACKNESS CONDITIONS AND SHORTEST PATH ROUTING

The complementary slackness conditions have interesting consequences for the routing problem therefore we outline them here. The path flow formulation contains a dual variable ω_{ij} for each link and another dual variable σ^k for each commodity k = 1...K . We define the reduced cost as

$$c_P^{\sigma,\omega} = c^k(P) + \sum_{(i,j)\in P} \omega_{ij} - \sigma^k$$

The path flow complementary slack conditions are stated in Ahuja et al [7]:

The commodity path flows f(P) are optimal in the path flow formulation eqn. (4) of the multi-commodity flow problem, if and only if for some arc prices

ω_{ij} and commodity prices σ^k the reduced costs and arc flows satisfy the following complementary slackness conditions:

$$\omega_{ij}[\sum_{1\le k\le K}\sum_{P\in P^k}\delta_{ij}(P)f(P) - u_{ij}] = 0 \text{ for all links (i,j)} \quad (7)$$

$$C_P^{\sigma,\omega} \ge 0 \text{ for all k = 1, K and all } P \in \mathbf{P^k} \quad (8)$$

$$C_P^{\sigma,\omega} f(P) = 0 \text{ for all k = 1,K and all } P \in \mathbf{P^k} \quad (9)$$

Eqn (7) implies that the dual price of a link is zero if the optimal solution f(P) does not use all the capacity of the link. Since in an optimal solution there must be some flow on the shortest path this also implies at least one flow using that link will be split. It is further stated in [7] that eqns.(8) and (9) imply that:

σ^k is the shortest path distance from origin node to destination node (Commodity k) with respect to the modified costs $c_{ij} + \omega_{ij}$ and in the optimal solution every path from source node to destination node that carries a positive flow must be a shortest path with respect to the modified costs.

3.1 A Simple LP Solution

Our aim in performing CBR is to "convince" the OSPF protocol to route the flows in correspondence with the primal solution of the multi-commodity flow problem. Consider a set of traffic demands, which, if routed on the shortest paths between origin destination pairs, would result in the capacity constraints, eqn (5), being violated. By solving the multi-commodity flow problem we can find additional paths for these flows, which, if a feasible solution exists, do not violate the capacity constraints. These paths, by virtue of the complementary slackness conditions must be shortest modified cost paths. In Fig. 1, if we let $C_{01} = C_{02} = C_{12} = 1$ (ie the given costs of the links). The dual solution to the linear program will be $\omega_{01} = \omega_{12} = 0$, and $\omega_{02} = 1$. This means there will be two equal "modified cost" paths from node 0 to node 2 both of length two units. From the primal solution, path 0-2, carries five units of flow and path 0-1-2 carries two units of flow. Again this is

consistent with the complementary slackness conditions in which a link with a non-zero dual price must be capacitated.

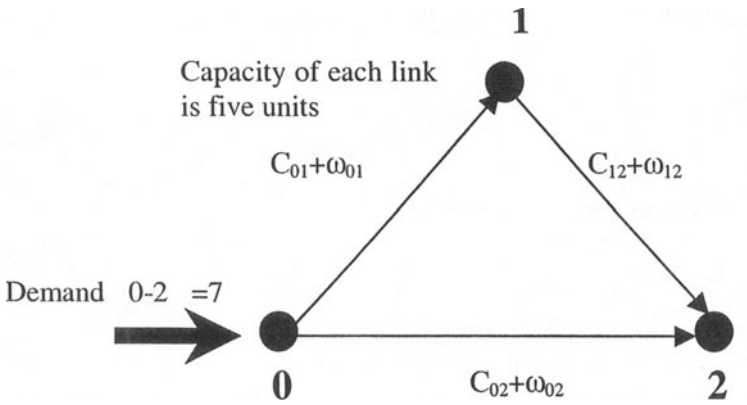

Fig. 1 An example of an LP solution in which some dual prices are non-zero.

The novel idea behind this method is that we can take advantage of the feature of the OSPF protocol to load balance flows across multiple paths without forming routing loops. This is precisely what the work of Fortz and Thorup [4] achieves. In contrast, in this method the required OSPF weights are directly outputted by the *dual* solution of the multi-commodity flow problem (as outlined above. It then remains to add the dual link costs (prices) to the original link costs to form the "modified link costs". By assigning the modified link costs to the OSPF metrics, several equal cost paths will become available to each flow if any links would have originally been congested. One must then calculate the proportion of each flow that is assigned to each equal modified cost path. We call this quantity the "splitting ratio". This information is obtained from the primal solution. Its method of calculation is described in the next section.

4. FINDING THE NODE SPLITTING RATIOS

Each router has a forwarding table in which each entry contains a destination and the link(s) a packet should take to reach that destination. If there is more than one path available from a particular node to a particular destination one must assign a "splitting ratio" in the forwarding table that specifies what proportion of the incoming flow should be sent on each outgoing link. Using

the primal solution we find that for node "n" the splitting ratio for traffic destined for node "t" on link (n,j) is given by

$$R_{nj}^{nt} = \frac{\sum_{D_t} \sum_{P \in P^k} \delta_{nj}(P) f(P)}{\sum_{A_n} \sum_{D_t} \sum_{P \in P^k} \delta_{nj}(P) f(P)} \tag{10}$$

for (n,j) $\in A_n$ and where A_n is the set of all outgoing links at node n and D_t is the set of flows going to t. We note here that $\mathbf{P^k}$ is the set of paths that the kth flow can take to destination t.

4.1 A Simple Example

Imagine we have two flows (commodities) A-2, and B-2 and each of these flows has two shortest paths to the destination node 2. We label the paths from node A to node 2 as P_1^1 and P_2^1 and the paths from node B to node 2 as P_1^2 and P_2^2. Using eqn. (10) we can calculate the splitting ratio for node 0 for traffic destined for node 2.

$$R_{01}^{02} = \frac{f(P_1^1) + f(P_1^2)}{f(P_1^1) + f(P_2^1) + f(P_1^2) + f(P_2^2)}$$

and

$$R_{02}^{02} = 1 - R_{01}^{02}$$

We note here that *every* flow from different origins to destination "t" will be split according to the ratios shown above. This is quite distinct from the primal solution and has implications for the splitting scheme employed at each node.

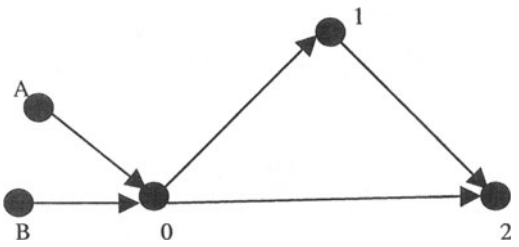

Fig 2. Schematic Diagram of Network with Two flows to a Single Destination

5. SCALABILITY ISSUES

5.1 Computational Complexity

The first scalability issue is the scalability of the LP program and the subsequent calculation of the splitting ratios. What size problem could we hope to solve using our standard CPLEX 7.1 tool? The multi-commodity flow problem is known to be polynomial [4]. Using the formulation in eqns. (0) –(3) a twenty-five node fully meshed network was tested with a set of approximately 600 demands. In this case there were approximately 15,000 constraints and 400,000 variables. The sizes of the demands were scaled up and randomised each time. At the point of infeasibility (the worst case) the run time was 25 minutes. Since most telecommunications networks are sparsely connected it is anticipated that the number of variables would be an order of magnitude smaller than this and, as a consequence, much larger problems could be tackled in a similar amount of time. To tackle even larger problems one needs to use the path flow formulation, eqns. (4) –(6) and restrict the number of allowed paths per flow. For the example above, this would reduce the problem (assuming five allowed paths per flow) to 3000 variables and 600 constraints.

Of course, each of set of paths have to be generated by a k-shortest path algorithm. This however is known to have a run time of $O(n^3)$ which is certainly faster than any algorithm used in CPLEX. Using the path flow formulation, assuming all origin-destination pairs as the set of demands then a problem with 125 nodes would generate 15,000 constraints and 75,000 variables. This is a large (non-hierarchical) network of a scale as large as many backbone ISPs [3].

It remains to estimate the computational complexity of the calculation of the splitting ratios, as defined by eqn (10). Clearly the set D_t is O(n) and

$\mathbf{P}^k$ is O(k). The paths f(P) will be stored in a link list whose length is $O(\sqrt{n})$ and there are m outgoing links from each node. Since we expect k to be a small number we postulate that k is $O(\sqrt{n})$ too. The number of outgoing links for each node is given by m thus the computational complexity of eqn.(10) is $O(mn^2)$. As we mentioned above, for sparse networks m is usually a small number hence we can safely assume that that the calculation of a set of splitting ratios from one origin to one destination is $O(n^2)$. Clearly the total computational complexity for calculating the entire routing table for each node is then $O(n^4)$. Again this is not expected to be less computationally complex than any algorithm used in CPLEX.

5.2 Quantity of State Information

The state information that has to be distributed is the set of "modified cost" link metrics and the set of "splitting ratios". The set of modified costs is the same for every node and of O(n) for sparsely connected networks. If we assume each piece of information (a floating point number) needs one byte, then a packet of length O(n) could be flooded throughout the network.

We mount a similar argument for the distribution of the splitting ratios. In the worst case that every node has traffic split across all its outgoing links the number of splitting ratios would be $O(mn^2)$. This is an extremely unlikely event, as it implies that every link is full. In practice, a solution would become infeasible long before this point. Also in cases of moderate loading (ie where only a few links are full) only a handful of splitting ratios are needed. Nevertheless, let us consider the worst case where every node is to be sent O(mn) pieces of information. Each piece of information is a floating point number, a fraction in fact, therefore we postulate that sufficient splitting granularity can be achieved (up to .5%) with a single byte of information. Thus a packet roughly of length O(mn) will need to be sent to each router.

For static routing, the time between updates is anticipated to be the order of hours, or even days, thus even for large networks (more than 100 nodes) the overhead involved in distributing the state information does not seem prohibitive.

Implementation

The choice of technique for both the collection of measurements and the distribution of splitting weights are dependent on the implementation of other parts of the system. Typically, both would use some form of network management protocol. SNMP (the Simple Network Management Protocol) is the most common such protocol and is particularly well suited to

measurement collection. SNMP can also be used for setting parameters in routers to distribute the splitting weights. Other protocols are also available for these functions such as telnet to router management interfaces, COPS (The Common Open Policy Service protocol) and modern XML based management [8].

Variable ratio traffic splitting (step 5) is not normally a part of most router implementations. Work has been done, however, to investigate the practicality of such techniques as part of the Optimised Multi-path initiative [9]. Briefly, effective use of multi-path cannot be done on a packet by packet basis in the general case due to the likelihood of packet re-ordering adversely impacting on transport layer performance. Therefore balancing traffic across multiple paths must be done per micro-flow. A common method for achieving this is to apply a hash function to the micro-flow identifier (typically source and destination IP address and transport layer port numbers) such as CRC16 and then split the flows dependent on whether the hash value exceeds a set threshold. By setting the threshold values appropriately, arbitrary splitting weights can be achieved for the different paths available. This is dependent on their being a sufficiently large number of micro-flows between each source/destination pairs for the splitting granularity to be fine which is normally the case in backbone networks.

6. CONCLUSION

The routing problem is formulated as a multi-commodity flow problem. This is solved using linear programming techniques. If the dual solution values are assigned to the OSPF metrics and the primal solution is used to calculate node splitting ratios, CBR can be performed, in principle, using the OSPF protocol. The OSPF metrics and the node splitting ratios to the routers can be distributed using existing network planning management protocols. The traffic can be split in any predetermined ratio across different paths using threshold based hashing. Unlike the implementation of MPLS all of this can be achieved using existing well-tested protocols

Formulating the OSPF weight setting problem as a linear program allows one to use the whole well developed machinery of linear programming. This means very large problems can be solved either from scratch or from an advanced basis. In addition parametric analysis can be performed to determine critical links and flows.

REFERENCES

[1] C. Huitema, *Routing in the Internet* (Prentice Hall 1995)
[2] X. Xiao, A. Hannan, B. Bailey and L.M. Ni, *Traffic Engineering with MPLS in the Internet,* IEEE Network **14** Mar/Apr (2000)
[3] X. Xiao and L.M. Ni, *Internet QOS: A Big Picture,* IEEE Network **13** Mar/April (1999)
[4] B. Fortz and M. Thorrup, *Internet Traffic Engineering by Optimizing OSPF Weights*, 19th Annual Conference of IEEE Computer and Communications Societies Proceedings **2,** p 519 INFOCOM 2000
[5] Http://www2.auckland.ac.nz/net/Accounting/ntm.Release.note.html
[6] Http://www.cisco.com/warp/public/732/Tech/netflow/
[7] R. K. Ahuja, T. L. Magnanti and J.B. Orlin, *Network Flows* (Prentice Hall 1993)
[8] (http://www.juniper.net/techcenter/techpapers/200017.html).
[9] C. Villamizar, *OSPF Optimized Multipath (OSPF-OMP),* www.brookfield.ans.net/ospf-omp/ospf-omp