A Robust Routing Plan to Optimize Throughput in Core Networks

Ali Tizghadam and Alberto Leon-Garcia

University of Toronto, Electrical and Computer Engineering Department
{ali.tizghadam, alberto.leongarcia}@utoronto.ca

Abstract. This paper presents an algorithm for finding a robust routing plan in core networks that takes into consideration network topology, available capacity, traffic demand, and quality of service (QoS) requirements. The algorithm addresses the difficult problem in routing and traffic engineering of optimal path selection. Our approach is inspired by the concept of "between-ness" from graph theory, from which we introduce quantitative metrics for link and path criticality. Paths are ranked according to path criticality and the algorithm tries to avoid placing flows on the most critical paths, maximizes throughput over the short term in the presence of QoS constraints, and attempts to increase the bandwidth of the critical paths for future use. The proposed approach shows promise relative to previous proposals in simulations on benchmark and experimental networks.

Keywords: Flow Assignment; Graph Theory; Quality of Service (QoS); Routing; Traffic Engineering.

1 Introduction

An abundance of work has already been done in the research community and industry to address the routing and flow assignment problem and the traffic engineering issues in core network systems, especially MPLS based networks [1], [2], [3] but still far from an ideal routing scheme. The main goal of the research reported in this paper is to examine the problem from a new standpoint, motivated by the definition of "between-ness" from graph theory [4], [5]. We introduce the notion of link and path criticality, and use them to identify the most critical paths, to build the routing plan on less critical paths in the short-term and optimize the throughput, as well as to plan the increase of bandwidth in the paths with high criticality index. We will show the extension of our algorithm to incorporate the quality of service (QoS) based routing. Our simulations on benchmark networks and realistic topologies discussed in the research literature show that the approach is promising.

The paper is organized as follows. Section 2 reviews the state of the art in routing plan and flow assignment problems. We describe the issues with existing methods and the associated research challenges. In section 3, we give the formal description of the problem, then in section 4, we propose our path-criticality based routing scheme.

Section 5 provides an extension of our approach to cover situations that have additive QoS constraints on the links of the network. Section 6 provides a "proof of concept" for our proposal. We assess the proposed method on benchmark networks as well as experimental scenarios and present encouraging results. Finally we conclude with a discussion of open issues and future work.

2 Previous Works

The most popular algorithm used in the research community for routing is the *shortest path routing algorithm* (SP). While the shortest path algorithm enjoys the benefit of simplicity, it can suffer from major problems to the network due to the lack of any load balancing mechanism. Widest shortest path (WSP) [7], improves the performance of SP, but the possibility of having bottlenecks remains. Furthermore both SP and WSP do not impose any form of admission control to control the flow in the network.

Ref. [1] introduced the minimum interference routing algorithm (MIRA). In contrast to the prior methods, MIRA considers the effect of source-destination pairs on the routing plan. A number (max-flow) is assigned to every source-destination pair (S, D), indicating the maximum amount of traffic that can be sent from S to D through the network. MIRA operates based on the notion that running traffic on some of the links may decrease the max-flow of (S, D). This process is called "interference". In brief MIRA tries to build the paths in such a way as to minimize interference. Introducing the notion of "interference" is significant, but some problems are still present with the algorithm introduced in [1]. MIRA concentrates on the effect of interference on just one source-destination pair, but there are situations where some links can cause bottlenecks on a cluster of node pairs. Ref. [3] investigates three benchmark networks: parking-lot (Fig. 1), concentrator and distributor, and shows that MIRA is unable to respond to the network flow requests appropriately and causes blocking for a large number of incoming flows in these networks. Finally MIRA is only designed to provide bandwidth guaranteed paths. It does not account for any other QoS constraint in the network.

Profile-based routing (PBR) is another proposal for routing of bandwidth guaranteed flows in MPLS networks [3]. PBR assumes that the source-destination pair and the traffic-profile between them are known. According to PBR, a traffic profile is the aggregate bandwidth demand for a specific traffic class between a source-destination pair. PBR has two phases. In the offline phase a multicommodity flow assignment problem is solved with the goal of routing as much commodity as possible. PBR also has some problems. Like MIRA, PBR deals only with bandwidth and does not consider QoS. Furthermore in [8] the authors introduce a network called "Rainbow Topology" and show that the performance of PBR in this network is much worse than MIRA and WSP. The main reason behind this is that PBR relies on the results of the offline phase which are not always correct.

In more recent works, the concentration is on oblivious routing to make the routing scheme independent of traffic but all of these approaches are far from an optimal solution due to over-provisioning [6] or because of considering a special case of core networks such as mesh and two-hop routes [15].

3 Problem Statement

Our goal is to achieve a robust routing plan. Robust routing plan in this work means a routing strategy that can cope with variations in traffic matrices as much as possible, as well as link/node failures and changes in community of interest (source-destination pairs). We start with the traditional formulation of the problem which is an LP formulation to optimize the selected metric or metrics. This formal presentation of the problem is adopted from [6]. Assume that the network is modeled as a directed graph $G(V,E), |V| = n, |E| = m$. In [13] the "Hose Model" for the network is proposed and used to manage VPN service. According to this model, one does not need to know the exact traffic matrix, but the maximum ingress/egress capacity of a node. This condition means that one can have any traffic matrix as long as the sum of its columns does not exceed γ_i^{in} (maximum ingress capacity for node i) and the sum of its rows is not more than γ_i^{out} (maximum egress capacity for node i) for any node in the network.

$$\sum_{j,j\neq i}\gamma_{ij} \leq \gamma_i^{in},\quad \sum_{j,j\neq i}\gamma_{ji} \leq \gamma_i^{out} \tag{a}$$

We denote the set of all traffic matrices which are satisfying (a) with Γ. Now we are looking for maximum multiplier θ such that all traffic matrices in $\theta \times \Gamma$ can be routed. Assuming that the maximum utilization of all the links in the network is u, then maximizing the throughput is equal to minimizing u. To find link-based routing (path-based can be easily obtained then [12]), we run one unit of bandwidth into the network between a source-destination pair (i, j) to obtain fraction of traffic that traverses link l. We show this fraction by $x_{ij}(l)$. Now we can write the linear programming (LP) problem to give us these fractions:

Minimize u

Subject to:

$$\sum_{l\in T^{in}(k)} x_{ij}(l) - \sum_{l\in T^{out}(k)} x_{ij}(l) = \begin{cases} 1 & k = i \\ -1 & k = j \\ 0 & otherwise \end{cases} \qquad i,j,k \in V \tag{1}$$

$$\sum_{i,j}\gamma_{ij}x_{ij}(l) \leq u \times c_l \qquad \forall l \in E,\ [\gamma_{ij}] \in \Gamma \tag{2}$$

$$x_{ij}(l) \geq 0 \qquad \forall l \in E,\ \forall i,j \in V \tag{3}$$

where $T^{in}(k)$, $T^{out}(k)$ are sets of incoming and outgoing links for node k respectively.

The set of constraints in (2) lead to an infinite number of constraints due to the hose model (traffic matrices could change as long as they meet requirements of (a)). Although it is possible to change this problem to an equivalent one with limited number of constraints using the dual of this LP problem [6], we do not choose this way as we want to provide a dynamic routing scheme that is robust to changing the traffic matrices as well as link failure and changing community of interest (source-destination pair).

4 Path Criticality Routing (PCR)

In this paper our goal is to find a robust routing plan for the core network that allows the network service provider to manage the assignment of flows to the paths primarily at the edge of the core network and obtain close to maximum throughput. To achieve the goal first we need to identify the important factors affecting the routing plan and flow assignment. One can summarize these factors as:

1. Network topology and connectivity.
2. Community of interest
3. Capacity of the links.
4. Traffic Matrix.

In order to have a robust routing plan we need to recognize the effect of link and node on network connectivity. Connectivity is a well studied subject in graph theory [4], [5], [10] allowing us to define some useful metrics to measure the sensitivity of the network to node or link failures. Capacity of a network is another key issue in flow assignment problem. Clearly the paths with more capacity are desired since the low capacity paths are prone to congestion. Hence an intelligent routing plan should avoid routing the flows onto the low capacity paths and should request for capacity increases for those paths if possible. Finally traffic demand directly affects the routing plan. The traffic demand profile may change from time to time (e.g. week-day traffic profile). Traffic changes might be predictable and periodical or chaotic. We need to find a routing scheme which is robust to the predicted traffic patterns and unpredicted ones to the extent possible.

We now introduce two metrics to estimate the effect of the aforementioned characteristics: link criticality index (LCI) and path criticality index (PCI) which are built based on the theory of graphs [4], [5]. We will subsequently propose our routing algorithm based on PCI.

A. Link Criticality Index

Freeman [4] introduced a useful measure in graph theory called "between-ness centrality." Suppose that we are measuring the centrality of node k. The between-ness centrality is defined as the share of times a node i needs a node k in order to reach a node j via the shortest path. We can modify the definition of between-ness centrality to introduce a useful measure for criticality of links in a network. Suppose

p_{sd} is the number of paths between source-destination pair (s,d) and p_{sld} is the number of paths between (s,d) containing the specific link l. Inspired by the definition of between-ness, one can quantify the effect of network topology by dividing p_{sld}/p_{sd} over all source-destination pairs. This gives an indication of how critical the link l is in the network topology. This topological aspect of the criticality of link l is then:

$$LCI_{top} = \sum_{s,d} p_{sld}/p_{sd} \tag{4}$$

The effect of link capacity and average demand for the source-destination (s,d) (provided by the traffic matrix) is accounted for in the residual bandwidth of the link. The residual bandwidth of link l is the capacity available after considering the flows already traversing the link, and is denoted by $c_r(l)$. Obviously the link criticality has an inverse relation with available bandwidth, and so we can account for residual bandwidth by multiplying equation (4) by $1/c_r(l)$.

The ability of a link to handle a given offered volume of data flow also has to be reflected in the link criticality. For example, suppose a new request offers flow γ_l to the link. Let the indicator function $I(x)$ and the modified link criticality be:

$$I(x) = 1 \ \ if \ \ x\rangle 0 \ \ otherwise \ \ 0 \tag{5}$$

$$LCI_{trc}(l) = 1/c_r(l) \times 1/I(c_r(l) - \gamma_l) \tag{6}$$

In addition: $$\frac{1}{c_r(l)} = \frac{1}{c_l - c_{used}} = \frac{1}{c_l \times (1-u_l)} \tag{7}$$

In this formula c_l is the link capacity, c_{used} is the used bandwidth by link l, c_l is the link capacity, c_{used} is the used bandwidth by link l, and u_l is the utilization of link l.

Hence: $$LCI(l) = \sum_{s,d} \frac{p_{sld}}{p_{sd}} \times \frac{1}{c_l \times (1-u_l)} \times \frac{1}{I(1-u_l)} \tag{8}$$

The indicator function can be written as above because $c_l \geq 0$. In this section we assume that there is no QoS constraint other than bandwidth. We will consider the case of multi-constraint routing in section 5. One can clearly see the effect of topology and connectivity p_{sld}/p_{sd} as well as capacity and traffic matrix $c_r(l)$ in the definition of LCI. In addition, this formulation is inline with our goal which was to minimize the maximum link utilization or maximizing throughput of the network. Indeed according to (8) when the utility of the link l increases and u_l gets near to 1, the criticality of the link increases dramatically, forcing the routing plan to find an alternative path.

B. Path Criticality Index (PCI)

Now we are ready to move to the next step and define the path criticality index (PCI). For every path between a source-destination pair (s,d) consisting of the links $(l_1, l_2, ..., l_n)$, the path criticality index is defined as the average of the link criticality index of l_1 to l_n. In other words:

$$PCI(s,d) = \frac{s(P)}{|P|} = \frac{\sum_{i=1}^{n} LCI(l_n)}{n} \quad \text{Where } |P| = n \text{ is the order of the path} \qquad (9)$$

In general PCI is a function LCIs of the path. Finding the best form of this function is one of our ongoing research topics, but the average function (9) which is introduced here works well for all of the benchmark networks and experimental topologies that we have examined.

C. Path Criticality Routing Algorithm (PCR)

The basic idea of our routing algorithm is to accommodate new requests for connections along paths that have a low PCI. This requires that we find the link criticality indices. To do this, we need to obtain all possible paths for each source-destination pair. This is not feasible since the number of paths grows rapidly with the number of network nodes and links. Although the shortest path is not necessarily the path with the lowest PCI, one can expect that the path or paths with lowest PCI are among the *k-shortest paths* of the network. Hence we use the k-shortest path method proposed by Eppstein [11] with a modification to avoid loops.

Our algorithm begins with a predefined value of k (default is 1), but the value may be increased during the course of running the routing algorithm if the desired number of paths to route the traffic cannot not be found. We use thresholds tr_1 (the default value is infinity) and tr_2 (the default value is zero) for PCI. The first threshold defines the lower confidence boundary for the path criticality index. All the paths with path criticality index less than tr_1 are considered eligible to route traffic. On the other hand all the paths with the criticality index larger than tr_2 are considered too risky and may be identified to the (offline) core network management system for increased capacity assignment. The paths with criticality index in between the thresholds will share traffic based on their criticality index as long as they remain within the boundaries. We note that when a path accepts traffic, the residual capacity of its links will decrease for the duration of the traffic flow. This means that the criticality index of this path must be increased. In other words a constant monitoring (not real-time necessarily but in reasonable time slots) of the PCI for all the paths is necessary.

PCR:

Input: A network or more formally a graph $G(V,E)$, a set of capacities (residual capacities if we are not in the initial stage), a set of source-destination pairs (s,d) and traffic matrix Γ for these source-destination pairs.

Output: A set of routes (LSPs in case of MPLS) between all source-destination pairs meeting the demand requirements according to the traffic matrix.

Algorithm: (By default we are in idle state)

```
1. Go to the initial state, Select $k$, $tr_1$ and $tr_2$ (PCR uses default values of $tr1$, $tr2$, and $k$ not concerned about the thresholds).
2. Compute the k-shortest paths for all source-destination pairs to meet the demand requirements and measure their $PCI$.
3. If a path with $PCI \leq tr_1$ exists, choose that path to route the demand (in case there are more than one path meeting the threshold requirements then choose the one with lowest PCI).
        i.Adjust residual capacities
        ii.Adjust path criticality indexes accordingly
   If there were still more than one path, choose one in round robin fashion.
4. If there are paths with $PCI \geq tr_2$ send a message to the core management system requesting additional bandwidth for these paths.
5. If there is no path satisfying the condition of step 3 then increase $k$ by one and go to step 2.
6. In case that no path with criticality index less than $tr_1$ can be found (this happens when $k$ keeps increasing without any satisfactory path results) then use the path with minimum PCI so that $tr_1 \leq PCI \leq tr_2$. If still there were more than one path, choose one in round robin fashion.
```

The most time-consuming part of the algorithm is the k-shortest path calculation. According to [11] the complexity of the proposed k-shortest path algorithm is $O(m + n\log n + k)$ where, m is the number of links and n is the number of the nodes. The algorithm is polynomial time and as a result PCR is also a polynomial time algorithm if we set a maximum value for k such as $k_{\max}$. The complexity of the other parts of the algorithm (without k-shortest path) is $O(m^2)$. The problem in this case is that we might not get the desired path from the algorithm by $k_{\max}$ iteration. On the other hand if we do not place any upper bound for k then we can find a desired path (if one exists) but not necessarily in polynomial time. This comes from the fact that the problem we are trying to solve by nature is an NP-Hard one [12].

In MIRA the time complexity without considering the max-flow algorithm is also $O(m^2)$. But if we compare the time complexity of the Tarjan max-flow method [12] that is being used in MIRA ($O(n \times m \times \log(n))$ where n and m are node and link dimensions) with the Eppstein k-shortest path method [11] used in PCR algorithm

($O(m+n\times\log n+k)$) we notice that the complexity of the k-shortest path algorithm is less than max-flow one.

5 Multiple QoS Constraints Case

We briefly discuss the situation in which the QoS constraints are included in the path routing (LSP routing in case of MPLS) problem. We can assume that the QoS constraints are additive without loss of generality [14].

To describe the multi-constrained routing we consider a graph $G(V,E)$ and assume each link $l=(u,v)$ is characterized by link weight vector $\vec{W}(l)=[w_1(l),w_2(l),...w_r(l)]^T$ where the component $w_i \succ 0$ is an additive QoS measure such as delay. There is also a vector of r constraints $\vec{R}=[R_1,R_2,...,R_r]^T$ determining the upper bounds on QoS measures (multi-constrained problem or MCP [14]). For any path P we have:

$$w_i(P)\equiv\sum_j w_i(l_j) \quad \forall \quad 1\le i\le r \text{ where } l_j\text{s are constituent links of } P.$$

A path is feasible if it satisfies all the constraints. To quantify these constraints we need to use a path length or "norm" in its mathematical sense. In other words any function $L(.)$ satisfying:

- $L(\vec{p})>0$ for all non zero vectors and $L(\vec{p})=0$ *iff* $\vec{p}=0$.
- For all $\vec{p}$ and $\vec{q}$ $L(\vec{p}+\vec{q})\le L(\vec{p})+L(\vec{q})$.

We are using Holder's q-vector norm [14]:

$$L_q(P)=(\sum_{i=1}^{r}[\frac{w_i(P)}{R_i}]^q)^{\frac{1}{q}} \quad \text{for } q\to\infty \text{ we have: } L_\infty=\max_{1\le i\le r}(\frac{w_i(P)}{R_i}) \tag{10}$$

Now we can use this norm function to quantify the QoS measure on paths and our formula for PCI of a path will be multiplied by length (10):

$$PCI_q(P)=\sum_i LCI(l_i)\Big/n\times L_\infty(P) \tag{11}$$

6 Proof of Concept

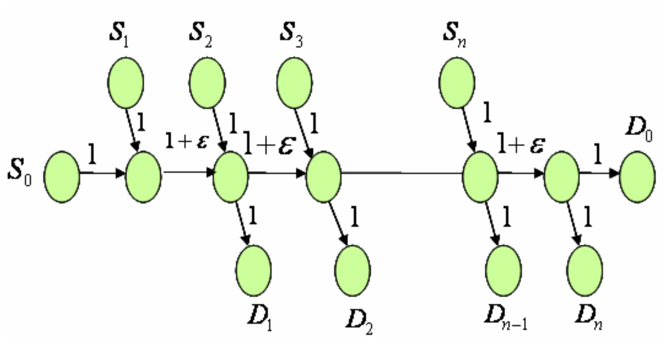
The PCR algorithm has been implemented with C++ and tested for many network configurations. Among these we have chosen one benchmark topology (parking lot in fig. 1) as well as a realistic networks (fig.2., a part of US network map [1] as well as Alilene [16]) to show the effectiveness of PCR algorithm to address problem of finding the best routing plan for networks that have been found difficult to be handled by previous proposals.

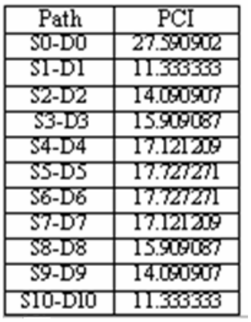
A. Parking-Lot Topology

The parking-lot network topology, shown in Fig. 1, is an interesting example. If one unit of bandwidth is requested to be sent from S_0 to D_0, all the previous routing approaches such as SP, WSP and MIRA will choose the straight path and run the flow resulting in the blocking of demands of one unit coming from any other source such as S_i to the destination D_i [3]. A wiser decision is to block the first request from S_0 to D_0 so the network will be able to route the other n source-destination pair requests.

To investigate the behavior of our *PCR* algorithm we suppose n = 10 and different combinations of source-destination pairs are possible. Our experiment results show that the criticality of the path $S0 \to D0$ is much higher than the other combinations. In Fig. 1 the results of our experiment with n=10 is reflected and clearly shows thate $S0 \to D0$ is the most critical path.

In general case of the parking-lot topology with n nodes the same approach can be followed and again the proposed routing plan will choose the straight path $S0 \to D0$ as the most critical one.





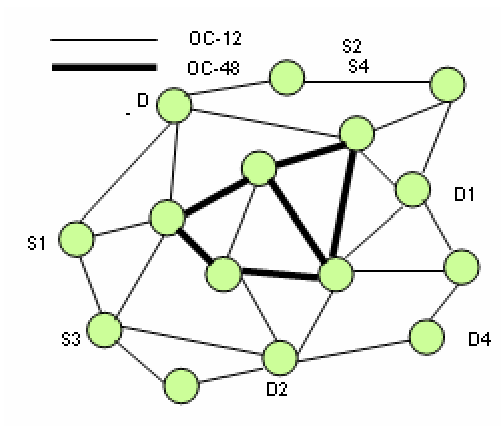
Path	PCI
S0-D0	27.590902
S1-D1	11.333333
S2-D2	14.090907
S3-D3	15.909087
S4-D4	17.121209
S5-D5	17.727271
S6-D6	17.727271
S7-D7	17.121209
S8-D8	15.909087
S9-D9	14.090907
S10-D10	11.333333

Fig. 1. Parking Lot Network (n=10)

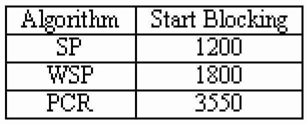
B. Simulation results for KL-Topology and Abilene

We ran a set of simulations on the network of Ref. [1] that we refer to as the KL-topology (Fig. 2(a). We assume the bandwidth of the thin links is 1200 units and that the thick links have 4800 units of bandwidth. In order to compare the results with [1] we implemented exactly the same simulations.

In the first experiment the requests for bandwidth (which is our main QoS measure in these simulations) arrive with Poisson distribution and stay for ever (no departures). In our tests the bandwidth requests for paths are taken to be uniformly distributed between 1 to 3 units. In Fig. 2(b) we show the number of rejected calls for the KL-topology and we compare the performance to that of shortest path, widest shortest path and PCR (with initial value $k = 1$ and possible subsequent increments based on PCR). We measured the number of blocked requests for from $S1$ to $D1$. As one can see after about 1200 trials the SP algorithm starts to experience blocking while the PCI-based algorithm can adapt itself and still accept bandwidth requests

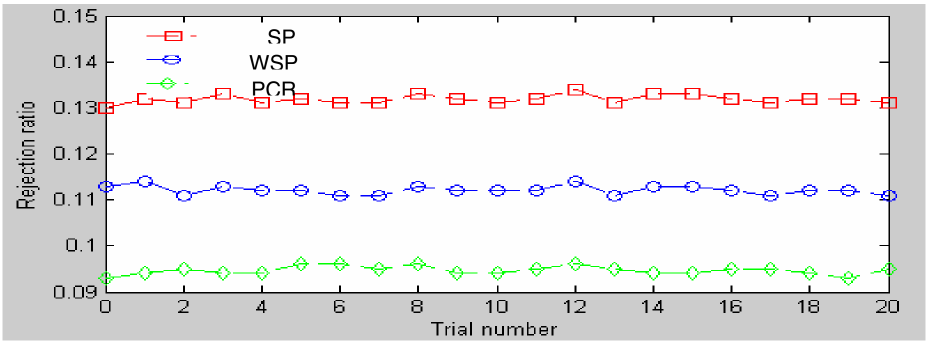


(a) KL Network



Algorithm	Start Blocking
SP	1200
WSP	1800
PCR	3550

(b) Static case, blockage



(c) Dynamic, Rejection ratio

Fig. 2.

without significant blockage. We observed that the PCR begins increasing the value k when the knee in the curve is reached.

In another experiment we examined the behavior of the algorithms in the presence of dynamic traffic. Fig. 2(c) shows the proportion of the path requests rejected in 20 experiments for the following scenario. Path requests arrive between each source-destination point according to a Poisson process with an average rate λ, and the holding times are exponentially distributed with mean $1/\mu$.

We assume $\lambda/\mu = 150$ in our experiments. In this scenario, we scale down the bandwidth of each link in the KL-network with the ratio of 10 to have bandwidths of 120 and 480 units for thin and thick links respectively. Next we generate about 1,000,000 requests and measure the rejection ratio for each one of the algorithms. The results are shown in Fig. 2(c) and again the results are very close to MIRA.

In last experiment we compare the response of PCR algorithm with the optimal solution which is obtained by solving the Linear Programming (LP) equations (1), (2), (3) using the method described in [6]. We conducted our algorithm on Abeline

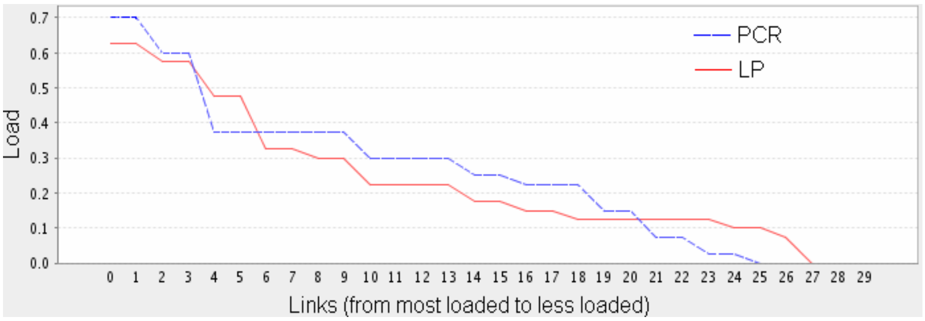


Fig. 3. Abilene network load distribution

network [16] (11 nodes, 28 links) using the traffic matrices obtained from [17] and observed the load distribution on different links of the Abilene network using LP and PCR methods. Figure 3 shows the result. The deviation from the optimal response using PCR was always less than 8%.

7 Concluding Remarks

In this paper we have proposed a new approach for path setup and routing of flows in core networks. The most important problem with the existing approaches is that each one of them solves a part of the overall problem but fails with other parts. We have tried to consider different aspects of the network (i.e. topology, capacity, and demand) and quantified these aspects using measures inspired by the mathematics of graphs. The essence of our work is based on determining a path criticality index for each path showing how critical that path is to the changes in the topology and traffic demand of a network. Our algorithm identifies the least critical paths for allocation of new traffic flow requests. The results from applying the proposed algorithm to networks that are difficult to handle by existing approaches are very encouraging. These results confirm the validity of the notion of path criticality. The simulation results show that PCR matches the performance of MIRA in typical networks. We also showed that the complexity of PCR relative to MIRA shows improvement.

However there are many issues that remain to be investigated in the new approach. We need to investigate more on the effect of the threshold parameters. The PCI is a function of link criticality indexes and in our first algorithm we used "average function" to obtain PCI but more elaboration is necessary. As another research challenge we need to look into the back up paths and the efficient algorithms to find them again with the goal of having less critical paths and back up paths.

References

[1] Kar, K., Kodialam, M., Lakshman, T.V.: Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. IEEE Journal on Selected Areas in Communications 18(12), 2566–2579 (2000)

[2] Awduche, D.O., Berger, L., Gan, D., Li, T., Swallow, G., Srinivasan, V.: Extensions to RSVP for LSP tunnels. Internet Draft draf-ietf-mpls-rsvp-lsp-tunnel-04.txt (September 1999)

[3] Suri, S., Waldvogel, M., Warkhede, P.R.: Profile-Based Routing: A New Framework for MPLS Traffic Engineering. In: Quality of Future Internet Services. LNCS, vol. 2156, Springer Verlag, Heidelberg (2001)

[4] Freeman, L.C.: Centrality in Networks: I. Conceptual Clarification. Social Networks 1, 215–239 (1978/79)

[5] Borgatti, L.S.P.: Centrality and Network Flow. Social Networks 27(1), 55–71 (2005)

[6] Kodialam, M., Lakshman, T.V., Sengupta, S.: Efficient and Robust Routing of Highly Variable Traffic, HotNets-III (November 2004)

[7] Guerin, R., Orda, A., Williams, D.: QoS routing mechanisms and OSPF extensions. In: Proceedings of 2nd Global Internet Miniconference (November 1997)

[8] Yilmaz, S., Matta, I.: On the Scalability-Performance Tradeoffs in MPLS and IP Routing. In: Proceedings of SPIE ITCOM (May (2002)

[9] Ott, T., Bogovic, T., Carpenter, T., Krishnan, K.R., Shallcross, D.: Algorithms for Flow Allocation for Multi Protocol Label Switching. MPLS International Conference (October 2000)

[10] Dekker, H., Colbert, B.D.: Network Robustness and Graph Topology. In: Proceedings of 27th Australasian Computer Science Conference, Vol. 26, pp. 359-368 (January 2004)

[11] Eppstein, D.: Finding the k Shortest Paths. Society for Industrial and Applied Mathematics (SIAM) Journal of Computing 28(2), 652–673 (1998)

[12] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs (1999)

[13] Duffield, N.G., Goyal, P., Greenberg, A.G., Mishra, P.P., Ramakrishnan, K.K., Van der Merwe, J.E.: A flexible model for resource management in virtual private network, ACM SIGCOMM 1999 (August 1999)

[14] Van Mieghem, P., Kuipers, F.A.: Concepts of Exact QoS Routing Algorithms. IEEE/ACM Transactions on Networking 12(5) (October 2004)

[15] Zhang-Shen, R., McKeown, N.: designing a Predictable Internet Backbone Networks. HotNet-III (November 2004)

[16] http://abilene.internet2.edu/

[17] http://totem.info.ucl.ac.be/