# Combined Use of Prioritized AIMD and Flow-Based Traffic Splitting for Robust TCP Load Balancing ⋆

Onur Alparslan, Nail Akar⋆⋆, and Ezhan Karasan

Electrical and Electronics Engineering Department,
Bilkent University, Ankara 06800, Turkey

**Abstract.** In this paper, we propose an AIMD-based TCP load balancing architecture in a backbone network where TCP flows are split between two explicitly routed paths, namely the primary and the secondary paths. We propose that primary paths have strict priority over the secondary paths with respect to packet forwarding and both paths are rate-controlled using ECN marking in the core and AIMD rate adjustment at the ingress nodes. We call this technique "prioritized AIMD". The buffers maintained at the ingress nodes for the two alternative paths help us predict the delay difference between the two paths which forms the basis for deciding on which path to forward a new-coming flow. We provide a simulation study for a large mesh network to demonstrate the efficiency of the proposed approach in terms of the average blocking rate and the average per-flow goodput.

**Key words:** Traffic engineering; load balancing; multi-path routing; TCP.

## 1 Introduction

IP traffic engineering controls how traffic flows through an IP network in order to optimize the resource utilization and network performance [3]. Multi-path routing is one of the traffic engineering approaches that has recently caught attention in the networking research community [5]. In multi-path routing, multiple explicitly routed paths with possibly disjoint links and nodes are established between the two end points of a network in order to optimize the resource utilization by intelligent traffic splitting. These explicitly routed paths are readily implementable using standard-based layer 2 technologies like ATM or MPLS or using source routed IP tunnels.

The work in [4] proposes a dynamic multi-path routing algorithm in connection-oriented networks where the shortest path is used under light traffic conditions

⋆ This work is supported in part by the Science and Research Council of Turkey (Tübitak) under project EEEAG-101E048

⋆⋆ Corresponding Author: Nail Akar, Electrical and Electronics Engineering Department, Bilkent University, Bilkent, Ankara 06800, Turkey, *e-mail:* akar@ee.bilkent.edu.tr, *voice:* 90 312 2902337, *fax:* 90 312 2664192

and as the shortest path becomes congested, multiple paths are used when and if available in order to balance the load. Recently, there have been a number of multi-path traffic engineering proposals specifically for MPLS networks that are amenable to distributed online implementation. In [9], probe packets are periodically transmitted to the egress Label Switch Router (LSR) which returns them back to the ingress LSR, so that the ingress LSR can collect the one-way congestion measures. The ingress LSR then uses these measures with a gradient projection algorithm for balancing the load among the LSPs (Label Switched Paths). Additive Increase/Multiplicative Decrease (AIMD) feedback algorithms are used generally for flow and congestion control in computer and communication networks [8],[11]. The multi-path AIMD-based approach of [24] uses binary feedback information for detecting the congestion state of the LSPs for allocating bandwidth to competing flows in an MPLS network. A traffic splitting heuristic using AIMD is proposed in [24] which ensures that source LSRs do not send traffic to secondary paths of longer length before making full use of their primary paths.

Some multi-path routing proposals cause possible de-sequencing (or reordering) of packets of a TCP flow. This is due to sending successive packets of a TCP flow over different paths with different one-way delays. The majority of the traffic in the current Internet is based on TCP and this packet de-sequencing adversely affects the application-layer performance of TCP flows [7],[13]. In order to avoid packet de-sequencing in multi-path routing, a flow-based splitting scheme that operates on a per-flow basis can be used [23],[14]. In [19] and [20], flow-based multi-path routing of elastic and streaming flows are discussed. Flow-based routing in the QoS routing context in MPLS networks is described in [15] and [17]; but these two studies require a flow aware core network and the cost of flow awareness may cause scalability problems with increasing number of instantaneous flows [15].

Recently, a new scalable flow-based multi-path traffic engineering approach for best-effort IP/MPLS networks is first proposed in [1] which employs max-min fair bandwidth sharing using an explicit rate control mechanism. This work demonstrates the performance enhancements attained by the flow-based splitting approach using comparisons with packet-based (i.e., non-flow based) multi-path routing and single-path routing when streaming traffic (i.e., UDP) is used. Significant reductions in packet loss rates are obtained relative to single-path routing in all the scenarios tested. This architecture is then studied for load balancing of elastic traffic (i.e., TCP) with AIMD-based rate control (as opposed to explicit rate) but with simple topologies (i.e., 3 node network) [2]. It is shown in [2] that flow-based multi-path routing method consistently outperforms the case of single-path. In the current paper, we provide an extensive simulation study of the approach proposed in [2] for TCP load balancing in larger and realistically sized mesh networks.

It is well-known that using alternative longer paths by some sources force other sources whose min-hop paths share links with these alternative paths to also use alternative paths [18]. This fact is called the knock-on effect in the lit-

erature and is studied in depth for alternately routed circuit switched networks [12]. Precautions should be taken to mitigate the knock-on effect for example the well-known "trunk reservation" concept in circuit switched networks [12]. One of the key ingredients of our proposed architecture is the use of strict priority queuing that favors packets of primary paths (PP) over those of secondary paths (SP) to cope with the knock-on effect. In this paper, we also compare and contrast strict priority queuing with the widely deployed FIFO queuing in their capabilities to deal with the knock-on effect in the TCP load-balancing context.

The remainder of the paper is organized as follows. In Section 2, we present our traffic engineering architecture. We provide our simulation results in Section 3. The final section is devoted to conclusions and future work.

## 2 Architecture

This section is mainly based on [2] but the proposed architecture is outlined here for the sake of completeness. In this study, we envision an IP backbone network which consists of edge and core nodes and which has mechanisms for establishing explicitly routed paths. In this network, edge nodes originate traffic and core nodes carry only transit traffic. Edge nodes are responsible for per-destination and per-class based queuing, flow classification, traffic splitting and rate control. Core nodes support per-class queuing and ECN (Explicit Congestion Notification) marking. In this architecture, flow awareness requirement is restricted to edge nodes making the overall architecture scalable.

Our architecture is based on three building blocks: (i) path establishment and queuing models used in network nodes, (ii) feedback mechanism and rate control, (iii) traffic splitting. Now we will describe these blocks. As far as queuing is concerned, the core nodes employ per-class queuing with three drop-tail queues, namely the gold, silver, and bronze queues. We propose that the gold queue has strict-priority over the silver queue which has strict-priority over the bronze queue. The gold queue is used for Resource Management (RM) and TCP ACK packets. Silver and bronze queues are used for TCP data packets according to the selection of paths as explained below. We assume in this study that edge nodes are single-homed, i.e., they have a link to a single core node. We setup one PP and one SP from an ingress node to every other edge node for which there is direct TCP/IP traffic. We impose that the two paths are link-disjoint. The PP is first established as the min-hop path. If there are multiple min-hop paths, the one with the minimum propagation delay is chosen as the PP. In order to find the route for the SP, we prune the links used by PP and compute the min-hop path in remaining network graph. A tie in this step is broken similarly. If the connectivity is lost after the first step, we do not establish a SP. If the traffic demand matrix was available, this information could be used for determining the optimum path sets; however we do not assume a-priori knowledge on traffic demands in this study.

In this paper, we study two queuing models based on the work in [1]. The first one is FIFO (first-in-first-out) queuing in which all the TCP data packets

**Table 1.** The AIMD algorithm

| |
|---|
| *if RM packet marked as CE* |
| $\quad ATR := ATR - RDF \times ATR$ |
| *else* |
| $\quad ATR := ATR + RIF \times PTR$ |
| $ATR := min(ATR, PTR)$ |
| $ATR := max(ATR, MTR)$ |

join the silver queue irrespective of the type of path they ride on. However, this queuing policy triggers the knock-on effect due to the lack of preferential treatment to packets using fewer resources (i.e., traversing fewer hops). Using longer secondary paths by some sources may force other sources whose primary paths share links with these secondary paths to also use secondary paths. In order to mitigate this cascading effect, longer secondary paths should be resorted to only if primary paths can no longer accommodate additional traffic. Based on the work described in [1] and [2], we propose strict priority queuing in which TCP data packets routed over PPs use the silver service and those routed over SPs receive the bronze service.

The second building block of the proposed architecture is the feedback mechanism and rate control. In our proposed architecture, ingress nodes periodically send RM packets to egress nodes, one over the PP (P-RM) and the other over the SP (S-RM). These RM packets are sent in every $T_{RM}$ seconds with the direction bit set to indicate the direction of flow. If strict priority queuing is used and when an P-RM (S-RM) packet arrives at the core node on its forward path, the node compares the percentage queue occupancy of its silver (bronze) queue on the outgoing interface with a predetermined configuration parameter $\mu$ and it sets the CE (Congestion Experienced) bit (if not already set) of the P-RM (S-RM) packet accordingly. If FIFO queuing is used then it is the silver queue occupancy that needs to be checked for both P-RM and S-RM packets. When an RM packet arrives at the egress node, it is sent back to the ingress LSR after resetting the direction bit of the RM packet. RM packets traveling over the reverse path are not marked whatsoever by the core nodes. When the RM packet arrives back at the ingress node, the CE bit indicates the congestion status of the path it was sent over. According to the information, the ingress node updates the ATR (Allowed Transmission Rate) of the corresponding rate-controlled path by using the AIMD algorithm given in Table 1 [8]. In this algorithm, $MTR$ and $PTR$ denote the Minimum Transmission Rate and Peak Transmission Rate and $RDF$ and $RIF$ denote the Rate Decrease Factor and Rate Increase Factor, respectively. We assume that the switching technology in the core has the necessary fields in the encapsulation header so as to implement the above-mentioned mechanisms.

The final ingredient to the proposed approach is the way we split traffic over the PP and the SP. The edge nodes first identify new flows. Then the delay estimates for the PP and SP queues (denoted by $D_{PP}$ and $D_{SP}$, respectively) in the edge nodes are calculated by dividing the occupancy of the corresponding

queue with the current drain rate. Upon the arrival of the first packet of the $n$th flow (e.g., a TCP SYN segment) a running estimate of the delay difference (denoted by $d_n$) is calculated as

$$d_n = \beta(D_{PP} - D_{SP}) + (1 - \beta)d_{n-1}, \quad (1)$$

where $\beta$ is a free parameter to be set by the network operator. If $d(n) \leq min_{th}$ ($d(n) \geq max_{th}$) then we forward the flow over the PP (SP). When $min_{th} \leq d_n \leq max_{th}$, then the new flow is forwarded over the SP with probability $p_0(d_n - min_{th})/(max_{th} - min_{th})$ where $min_{th}, max_{th}, p_0$ are free algorithm parameters to be set. All the remaining packets of the same flow will then follow the same path. This traffic splitting mechanism is called Random Early Reroute (RER) which is inspired by the RED (Random Early Detect) algorithm used for active queue management in the Internet [10]; note the similarity in the algorithm parameters. RED is used for controlling the average queue occupancy whereas the average smoothed delay difference of silver and bronze queues is controlled by RER. RER parameters are generally chosen so that the PP is favored (i.e., $min_{th} \geq 0$) and proportional control (as opposed to on-off control) is used ($max_{th}$ strictly larger than $min_{th}$).

## 3 Numerical Results

In this paper, we present the simulation results of our AIMD-based multi-path TE algorithm for TCP traffic over a mesh network called the hypothetic US topology that has 12 POPs (Point of Presence). This network topology and the traffic demand matrix is given in `www.fictitious.org/omp` and is also described in [1]. In our simulations, we scaled down the capacities of all links by a factor of 45/155 (OC-3 links with DS-3) to reduce the simulation runtimes. We assume that each of the POPs have multiple edge nodes connected via very high speed links to one core node.

In the simulations, we used a traffic model where flow arrivals occur according to a Poisson process and flow sizes have a bounded Pareto distribution denoted by $BP(k, p, \alpha)$ [22]. The following parameters are used for the bounded Pareto distribution in this study: $k = 4000$ Bytes, $p = 50 \times 10^6$ Bytes, and $\alpha = 1.20$, corresponding to a mean flow size of $m = 20{,}362$ Bytes.

Single-path routing uses the minimum-hop path with the AIMD-ECN capability turned on. We use the term Shortest Delay (SD) for the routing policy when RER parameters are $min_{th} = max_{th} = 0$ msec and $p_0 = 1$; SD forwards each flow simply to the path with the minimum estimated queuing delay at the ingress edge node, and therefore it does not favor the PP. SD is used in conjunction with the FIFO queuing discipline where there is no preferential treatment between the PP and the SP at the core nodes.

The delay averaging parameter is set to $\beta = 0.3$. TCP data packets are assumed to be 1040 Bytes long (after encapsulation). We assume that the RM packets are 50 Bytes long. All the buffers at the edge and core nodes, including per-destination (primary and secondary) and per-class queues (gold, silver and

bronze), have a size of 104,000 Bytes each. The TCP receive buffer is of length 20,000 Bytes.

We fix the following parameters for the AIMD algorithm. $PTR$ is chosen as the speed of the slowest link on its path. $MTR$ is chosen as 1 bit/sec in order to eliminate cases causing division by zero in the simulations. If the expected delay of a buffer exceeds 0.36 sec, the packets destined to this queue are dropped. We use $T_{RM} = 0.02$ sec. and $\mu = 20\%$.

This TCP TE architecture is implemented over ns-2 (Network Simulator) version 2.27 [16] and TCP-Reno is used in our simulations. We implemented a number of new modules over ns-2 that are required by the new architecture. These include a new per-destination based queuing system where multiple queues sharing the same link are drained according to their ATRs. For routing of packets, a new source routing module accepting multiple possible paths for flows is implemented. The flow requests are created offline by creating the arrival time, size and s-d pair of all flows according to the traffic demands of s-d pairs and bounded Pareto distribution of flow sizes. Each run of the simulator accepts this scenario file as the input. Therefore, flow arrival requests are the same in all simulations. In ns-2, the trace-driven source-level simulation [21] of mesh topologies for relatively long time periods is problematic for example the need for large memory requirements and excessively long simulation runtimes. In order to be able to efficiently simulate mesh topologies, we introduce a number of optimizations to the ns-2 simulator. A new simulation architecture is developed that detaches and stores the objects associated with flows whose data transmissions are completed, and reuses them upon new flow arrivals by resetting and attaching to the new s-d pair. In this architecture, a new flow is created only if there is no flow left in the list of unused flows. For stable systems that reach the steady-state, the maximum number of flows in the simulation, hence the maximum memory required by flows, becomes essentially fixed independent of the duration of the simulation. Moreover, the size of the routing table for the source routing module is minimized and made independent of the number of flows by using a hashing based on source-destination addresses and path numbers. We also make some modifications on the calendar scheduler [6] which is the default scheduler for ns-2 and is known to have performance problems in case the time distribution of events is highly non-uniform [6]. The simulation runtime is selected as 300 sec. We report only the statistics related to those flows that have started in the interval [90 sec, 250 sec].

The following three algorithms are compared in terms of their performances:

- Flow-based multi-path with RER and Strict Priority queueing
- Flow-based multi-path with Shortest Delay (SD) and FIFO queueing
- Shortest-path (i.e., Single Path using the min-hop path)

The goodput of the TCP flow $i$ (in KBytes/sec), $G_i$, is defined as the service rate received by flow $i$ during its lifetime or equivalently it is the ratio $G_i = \Delta_i/T_i$, where $\Delta_i$ is the number of Bytes successfully delivered to the application layer by the TCP receiver for flow $i$ within the simulation duration, i.e., before

300 sec. The parameter $T_i$ is the sojourn time of the flow $i$ within the simulation runtime. We note that if flow $i$ terminates within the simulation time, then $\Delta_i$ will be equal to the flow size in Bytes. We compute the normalized goodput of the total traffic, which is defined as

$$G_{norm-avg} = \frac{\sum_i \Delta_i G_i}{\sum_i S_i}$$

where $G_i$ is the average throughput attained by flow $i$, and $S_i$ is the size of flow $i$. We also compute the normalized goodput with respect to carried traffic, which is defined as

$$G_{norm-avg-ca} = \frac{\sum_i \Delta_i G_i}{\sum_i \Delta_i}$$

where $\Delta_i$ is the number of Bytes successfully delivered by flow $i$ within the simulation duration.

In order to show the ratio of data that cannot be delivered within the simulation duration, we calculate Byte Rejection Ratio denoted by $BRR$:

$$BRR = \frac{\sum_{s,d} N(s,d) - \sum_{s,d} \Gamma(s,d)}{\sum_{s,d} N(s,d)} * 100$$

where $N(s,d)$ is the sum of the sizes of flows demanded from node $s$ to node $d$, and $\Gamma(s,d)$ is the sum of the number of Bytes successfully delivered by the TCP flows from node $s$ to node $d$.

In Figure 1a and 1b, the effects of $RIF$ and $RDF$ on $G_{norm-avg}$ are shown. Similarly, in Figure 1c and 1d the effect of these AIMD parameters on $BRR$ is depicted. In these simulations, RER parameters are chosen as $min_{th} = 1msec$, $max_{th} = 15msec$ and the strict-priority policy is used. It is seen that multi-path strict-priority with RER gives better performance in both measures than single-path policy. The choice of $RDF = 0.0625$ and $RIF = 0.0625$ gives relatively good and robust performance and therefore we use these parameters in the rest of the paper. The effects of RER parameters on $G_{norm-avg}$ and $BRR$ are shown in Figures 2a and 2b, respectively. We observe that, except for the choices of RER parameters close to $min_{th} = max_{th} = 0$, which basically corresponds to the SD policy, the performance of the RER is quite robust. When SD policy is used, the performance is impacted severely. As we increase $min_{th}$ and $max_{th}$, we observe that the performance of RER converges to that of the single-path routing. Based on these observations, we use the RER parameters $min_{th} = 1$ msec. and $max_{th} = 15$ msec. We set $p_0 = 1$.

In order to show the effect of the total amount of the traffic demand, the traffic is scaled by multiplying the flow sizes with a traffic scaling parameter $\gamma$ where $0.5 \leq \gamma < 1$, while keeping the flow arrival times same. As seen in Figure 3a, at high traffic rates the multi-path TE with strict-priority and RER achieves the highest $G_{norm-avg}$. In fact, there are node pairs, that have the maximum traffic demand in the network, for which the increase in goodput is more than 10 times with the multi-path TE with strict-priority and RER compared to

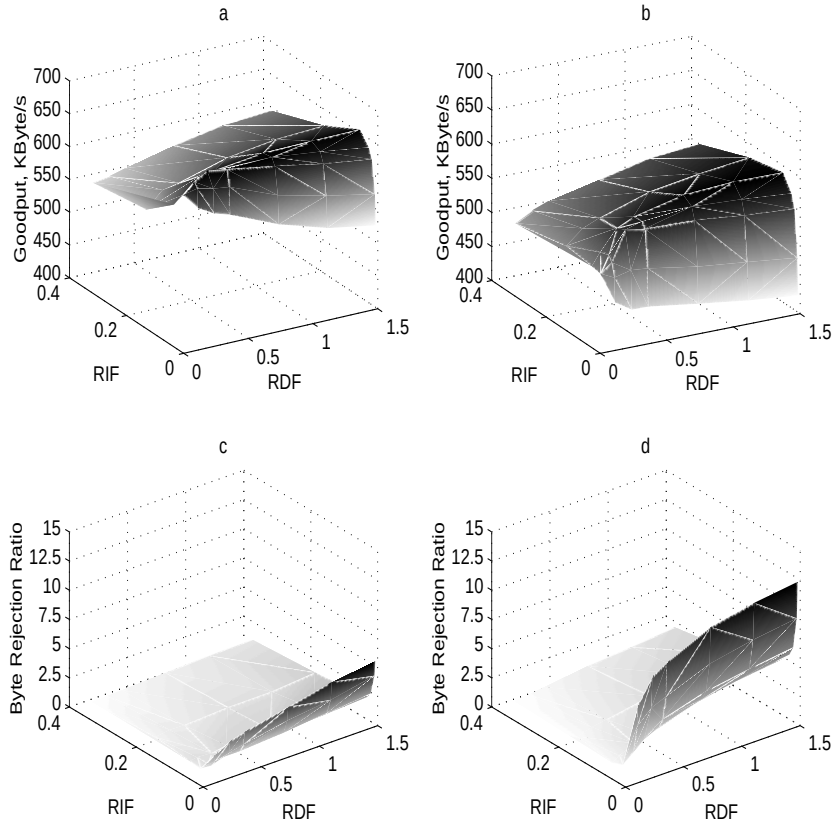


**Fig. 1.** As a function of $RIF$ and $RDF$: (a) $G_{norm-avg}$ for the multi-path TE with strict-priority and RER, (b) $G_{norm-avg}$ for the single-path routing, (c) $BRR$ for the multi-path TE with strict-priority and RER, (d) $BRR$ for the single-path routing

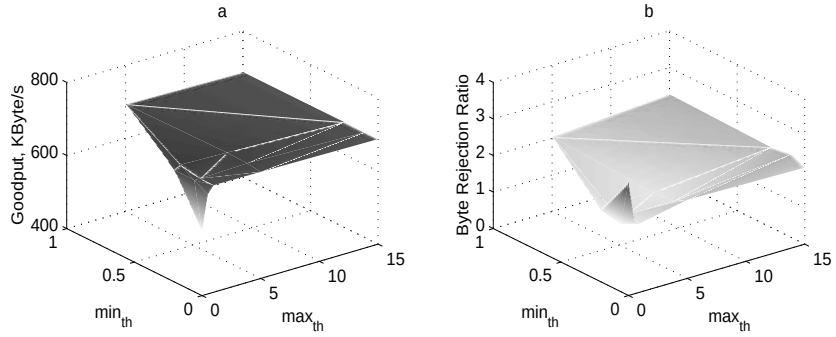


**Fig. 2.** As a function of $min_{th}$ and $max_{th}$: (a) $G_{norm-avg}$ for the multi-path TE with strict-priority and RER, (b) $BRR$ for the multi-path TE with strict-priority and RER

the single-path routing. For these node pairs the PP is heavily congested, and the SP substantially improves the performance. On the other hand, for many node pairs multi-path routing does not improve the goodput since the PP is not congested. The overall performance, represented by $G_{norm-avg}$ which is the average normalized goodput taken over 132 node pairs, still shows a significant improvement for the congested cases.

It is also observed from Figure 3a that at high traffic rates the multi-path TE with strict-priority and RER achieves the highest $G_{norm-avg-ca}$(denoted by CA in the figure). This shows that the multi-path TE with strict-priority and RER not only carries more traffic in Bytes, but also the carried flows are transported faster. In Figure 3b, we observe that the policy of multi-path routing with strict-priority and RER has a $BRR$ which is approximately half of the $BRR$ for the single-path routing. This indicates that the performance of congested paths are drastically improved when multi-path routing with strict-priority and RER is used. As the traffic demand decreases, we see that the gap between the multi-path routing with strict-priority and RER and the single-path routing disappears. This is due to the fact that at light traffic loads PP is not congested, and the multi-path routing effectively behaves as single-path routing. In Figure 3b, $BRR$ for the multi-path routing with SD and FIFO is less than the multi-path routing with strict-priority and RER, but the normalized goodputs for this scheme, as shown in Figure 3a, are much lower.

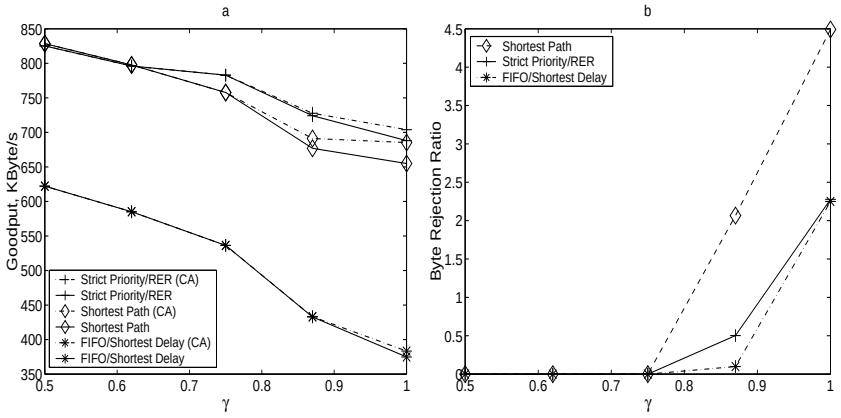


**Fig. 3.** As a function of traffic scaling parameter $\gamma$: (a) $G_{norm-avg}$ and $G_{norm-avg-ca}$ (denoted by CA), (b) Byte Rejection Ratio

## 4 Conclusions

In this paper, we report our findings on a recently proposed TCP load balancing architecture that uses prioritized AIMD and flow-based multi-path routing with RER (Random Early Reroute). Using a publicly used test network, we show that our proposed architecture consistently outperforms the case of a single path in terms of average normalized goodput and the byte rejection ratio. On the other hand, employing load balancing without RER and prioritization does not always produce better results than that of a single path which can be explained by the knock-on effect.

In the current study, we did not assume a-priori knowledge on the traffic demand matrix and we therefore did not optimize the path sets with respect to a certain criterion. Consequently, we observe in the simulations that a number of s-d pairs traverse multiple bottleneck links on their PPs. Moreover, a number of them have SPs that traverse heavily congested links, which limits the usage of them. In spite of these limitations, our proposed architecture is shown to give better or at least equivalent results than that of the single-path routing policy. We are currently extending this framework by using an estimate of the traffic demand matrix in optimal path set determination.

## References


1. N. Akar, I. Hokelek, M. Atik, and E. Karaşan. A reordering-free multipath traffic engineering architecture for Diffserv/MPLS networks. In *Proceedings of IEEE Workshop on IP Operations and Management*, pages 107–113, Kansas City, Missouri, USA, 2003.
2. O. Alparslan, N. Akar, and E.Karasan. AIMD-Based Online MPLS Traffic Engineering for TCP Flows via Distributed Multi-Path Routing. www.binlab.bilkent.edu.tr/e/journal/125/annales_final.pdf. Accepted for publication in Annales Des Telecommunications.
3. D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. IETF Informational RFC-3272, May 2002.
4. S. Bahk and M. E. Zarki. Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *ACM SIGCOMM*, pages 53–64, 1992.
5. Debashis Basak, Hema Tahilramani Kaur, and Shivkumar Kalyanaraman. Traffic engineering techniques and algorithms for the Internet. www.ecse.rpi.edu/Homepages/shivkuma/research/papers/survey-paper-v3.pdf, 2002. Tech. report.
6. R. Brown. Calendar queues: a fast O(1) priority queue implementation for the simulation of event set problem. *Communications of the ACM*, 31(10):1220–1227, 1988.
7. Z. Cao, Z. Wang, and E. W. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM (1)*, pages 332–341, 2000.
8. D. M. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, June 1989.

9. A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *Proceedings of INFOCOM*, pages 1300–1309, 2001.
10. Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.
11. V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, 18, 4:314–329, 1988.
12. F. P. Kelly. Routing in circuit switched networks: Optimization, shadow prices and decentralization. *Advances in Applied Probability*, 20:112–144, 1988.
13. M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *IEEE Network Magazine*, 16(5):28–36, 2002.
14. Youngseok Lee and Yanghee Choi. An adaptive flow-level load control scheme for multipath forwarding. In *Networking - ICN 2001*, 2001.
15. Y-D. Lin, N-B. Hsu, and R-H. Hwang. QoS routing granularity in MPLS networks. *IEEE Comm. Mag.*, 46(2):58–65, 2002.
16. S. McCanne and S. Floyd. ns Network Simulator. Web page: http://www.isi.edu/nsnam/ns/, July 2002.
17. S. Nelakuditi and Z-L. Zhang. A localized adaptive proportioning approach to QoS routing. *IEEE Comm. Mag.*, 46(2):66–71, 2002.
18. S. Nelakuditi, Z. L. Zhang, and R. P. Tsang. Adaptive proportional routing: A localized QoS routing approach. In *Proceedings of INFOCOM*, Anchorage, USA, 2000.
19. S. Oueslati-Boulahia and E. Oubagha. An approach to elastic flow routing. In *International Teletraffic Congress*, Edinburgh, UK, June 1999.
20. S. Oueslati-Boulahia and J. W. Roberts. Impact of trunk reservation on elastic flow routing. In *Networking 2000*, March 2000.
21. V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. In *1997 Winter Simulation Conference*, 1997.
22. Idris A. Rai, Guillaume Urvoy-Keller, and Ernst W. Biersack. Analysis of LAS scheduling for job size distributions with high variance. In *Proceedings of ACM Sigmetrics*, pages 218–228, 2003.
23. A. Shaikh, J. Rexford, and Kang G. Shin. Load-sensitive routing of long-lived IP flows. In *ACM SIGCOMM*, pages 215–226, 1999.
24. J. Wang, S. Patek, H. Wang, and J. Liebeherr. Traffic engineering with AIMD in MPLS networks. In *7th IFIP/IEEE International Workshop on Protocols for High-Speed Networks*, pages 192–210, 2002.