TCP FLOW AWARE ADAPTIVE PATH SWITCHING IN DIFFSERV ENABLED MPLS NETWORKS

<table>
<thead>
<tr>
<th>Journal:</th>
<th>European Transactions on Telecommunications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID:</td>
<td>draft</td>
</tr>
<tr>
<td>Wiley - Manuscript type:</td>
<td>Research Article</td>
</tr>
<tr>
<td>Date Submitted by the Author:</td>
<td></td>
</tr>
<tr>
<td>Complete List of Authors:</td>
<td>Alparslan, Onur; Osaka University, Department of Information Networking Akar, Nail; Bilkent University, Electrical and Electronics Engineering Dept. Karasan, Ezhan; Bilkent University, Electrical and Electronics Engineering Dept.</td>
</tr>
<tr>
<td>Keywords:</td>
<td>Multipath routing, path switching, traffic engineering, MPLS, differentiated services</td>
</tr>
</tbody>
</table>
TCP FLOW AWARE ADAPTIVE PATH SWITCHING
IN DIFFSERV ENABLED MPLS NETWORKS*

Onur Alparslan¹, Nail Akar² and Ezhan Karasan³

ABSTRACT

We propose an adaptive flow-level multipath routing-based traffic engineering solution for an IP backbone network carrying TCP/IP traffic. Incoming TCP flows are switched between two explicitly routed paths, namely the primary and secondary paths, for resilience and potential goodput improvement at the TCP layer. In the proposed architecture, primary paths receive a preferential treatment over secondary paths with respect to packet forwarding using differentiated services mechanisms. The reason for this choice is not for service differentiation but for coping with the detrimental knock-on effect stemming from the use of longer secondary paths that is well known for conventional network load balancing algorithms. Moreover, both paths are congestion-controlled using ECN marking at the core and AIMD rate adjustment at the ingress nodes. The delay difference between primary and secondary paths is estimated using two per-egress rate-controlling buffers maintained at the ingress nodes for each path, and this delay difference is used for determining the path over which a new TCP flow will be routed. We perform extensive simulations using ns-2 in order to demonstrate the viability of the proposed distributed adaptive multipath routing method in terms of per-flow TCP goodput. The proposed solution consistently outperforms the single-path routing policy and provides substantial per-flow goodput gains under poor primary path conditions. Moreover, highest goodput improvements under the proposed scheme are achieved by flows that receive the lowest goodputs with single-path routing, while the performances of the flows with high goodputs do not deteriorate with the proposed path switching technique.

Keywords: Multipath routing; path switching; traffic engineering; MPLS; differentiated services.

1. INTRODUCTION

Traffic Engineering (TE) is defined as the set of mechanisms that control how traffic flows through a network so as to optimize resource utilization and network performance [1]. Traffic engineering mechanisms can be applied to hop-by-hop, explicit, or multi-path routing networks. Traditional hop-by-hop routed IP networks using intra-domain routing protocols, such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS), resort to shortest path routing with simple link weights such as hop-count, delay, or the inverse of the link bandwidth. Although the simplicity of this approach allows IP routing to scale to very large networks, it does not make the best use of network resources. If the traffic demand matrix is known a-priori, the link weights can be determined by solving an optimization problem for traffic engineering purposes [2]. Although the problem of obtaining optimal weights is known to be NP-hard, a number of heuristics can be used to find near-optimal solutions [2],[3]. Despite the relative ease of implementation of link weight-based methods, it is not only hard to reach optimality by using shortest path routing but also hard to estimate traffic demands that may comprise unpredictable spikes [4]. On the other hand, explicitly routed networks allow traffic to follow any desired route as opposed to one that is computed by hop-by-hop destination-based routing protocols. As a technology, Multi-Protocol Label Switching (MPLS) provides the necessary protocols and mechanisms for IP backbones to enable explicit routing to facilitate traffic engineering. There are a variety of on-line and off-line traffic engineering proposals for explicitly routed MPLS networks; see [5] for an extensive treatment of MPLS TE methods and applications. Off-line methods solve mathematical programs for optimal routes using long term average traffic demands as inputs. However, the drawback of using offline methods is that they cannot react to real-time traffic changes or traffic spikes.

* This work is supported in part by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under projects EEEAG-104E047, EEEAG-105E065 and EEEAG-106E046.

¹Advanced Network Architecture Laboratory, Department of Information Networking Technology, Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan.
²Electrical and Electronics Engineering Department, Bilkent University, 06800 Bilkent, Ankara, Turkey.
³Corresponding Author, Electrical and Electronics Engineering Department, Bilkent University, 06800 Bilkent, Ankara, Turkey. Phone: (90) 312 2901308, Fax (90) 312 2664192, e-mail: ezhan@ee.bilkent.edu.tr.

http://mc.manuscriptcentral.com/ett
Multi-path routing is a general umbrella term referring to routing mechanisms that take advantage of path diversity between source-destination pairs. Dispersity routing, for example, subdivides a certain message and disperses it through a multiplicity of paths using redundancy so as to cope with paths with long delays or high loss rates [6]. Path switching, on the other hand, refers to the ability of an end system to dynamically switch among multiple paths (one path at a time) for a given message or flow, to a destination depending on the current status of these paths made available by path probing mechanisms [7]. In MATE [8] proposed for MPLS networks, the ingress Label Switch Router (LSR) transmits probe packets periodically to the egress LSR which then returns the probe packets back to the ingress LSR, carrying information on the Label Switched Path (LSP) characteristics. Based on the gathered information, the MATE algorithm tries to equalize the congestion measure, e.g., delays, for each LSP, by appropriately splitting traffic across multiple LSPs. In a similar technique [9], the authors focus on elastic traffic and make adaptive path switching decisions at the ingress device based on Round Trip Time (RTT) measurements. Some of the existing path switching proposals rely on path diversity provided by multihoming or overlay scenarios. One such scenario from the existing literature is a VoIP service provider multihomed to several ISPs which dynamically determines the path to forward voice packets [10]. The work of [11] uses similar principles but for an overlay networking scenario in conjunction with redundant dispersity routing to improve application-level performance under time-varying congestion on Internet paths and failures. Although most path switching proposals rely on path probing for which no network involvement is required, a number of methods exist that are based on network assistance. For example, TeXCP proposed in [12] splits the multi-path routing problem into two components; namely the load balancer and the rate controller. The load balancer component takes the state of the network as the input and makes path switching decisions to minimize utilization. On the other hand, each path is congestion controlled by network involvement with the control loop operating at a faster time scale compared to the load balancer.

Another line of recent research focuses on finding stability conditions for rate control in conjunction with dynamic multi-path routing. For example, in [13], sufficient conditions for the local stability of end-to-end algorithms for joint routing and rate control are given for a network with arbitrary interconnection of sources and resources, and heterogeneous propagation delays. The reference [14] provides theoretical justification of certain decentralized algorithms for joint optimization of congestion control and multipath routing. However, experimental studies of moderately sized TCP/IP networks are not presented in these two papers.

In the current paper, as opposed to a theoretical justification, we take the approach of experimenting with a heuristical dynamic multipath routing algorithm in moderately-sized TCP/IP networks and for a wide range of scenarios. In particular, we address the problem of path switching in an explicitly routed IP network provider scenario (e.g., MPLS network) in which the ingress nodes of the network decide on which explicitly routed path to forward the incoming IP packets. We concentrate only on elastic TCP traffic in this paper due to its dominant use in the Internet although the extension of the ideas to UDP is straightforward. The case of two paths, one being the min-hop primary path and the second one being the alternative secondary path, is studied here, but the architecture is amenable for extension to two or more secondary paths. Experimental studies reveal that packet de-sequencing within a TCP flow can significantly deteriorate end-system TCP performance [15] and therefore we focus on flow-level path switching mechanisms as in [9] and [16] that identify the TCP flows at the ingress of the network and dynamically determine the most convenient path on a per flow-basis with the aim of avoiding packet de-sequencing within a flow. Although flow-level granularity for traffic splitting suffices for shorter flows (flows of “mice” type), it may not give as satisfactory results for longer flows (flows of “elephants” type). For this reason, the concept of flowlet-level traffic splitting as opposed to packet- or flow-level splitting is introduced in [17], where a flowlet is defined as a burst of packets appropriately chosen to avoid reordering. Our work is similar in spirit to [12] which uses feedback based rate control for each path and packet-level non-elastic UDP traffic splitting whereas we study elastic TCP flow-level splitting in the current paper. Improved splitting granularity, such as flowlet-level switching, is left outside the scope of the paper.

It is well-known that using alternative longer paths by some traffic 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 literature and is studied in depth for alternately routed circuit switched networks [19]. Precautions should be taken to mitigate the knock-on effect for example the well-known “trunk reservation”
concept in circuit switched networks [19]. One of the key ingredients of our proposed architecture for IP-MPLS packet networks is the use of Modified Deficit Round Robin (MDRR) per-class scheduling that favors packets routed along primary paths (PP) over those routed along secondary paths (SP) to cope with the knock-on effect. In this way, we are able to use min-hop paths as long as they are not congested and start moving some of the traffic to alternative paths when congestion arises along PPs without jeopardizing the already existing flows on min-hop paths. In this paper, we also compare MDRR scheduling with the widely deployed First In First Out (FIFO) queuing in terms of their capabilities to deal with the knock-on effect in the context of TCP flow-level traffic engineering. Prioritized routing or trunk reservation concepts have also appeared recently in [16] and [20] in the context of Internet and optical burst switching networks, respectively.

From a positioning standpoint, we note two differences of our model from the multihoming or overlay path switching scenarios that are already studied in the literature:

• In the current network provider scenario, the path switching entity belongs to the network provider and therefore the network can actively participate in making path switching decisions,

• While choosing the paths to route flows, the network addresses the question of “what is good for the network” rather than “what is good for the application” since all the path switching entities in the network would collectively operate for optimizing the resource utilization and network performance as a whole. In this sense, we use path switching for traffic engineering purposes within a network provider domain.

This paper finds its roots in our preliminary studies [21] and [22], but differs from those in the following:

• We propose MDRR scheduling at the core nodes rather than strict priority queueing to mitigate the knock-on effect which leads to significant performance improvement.

• Using ns-2, we simulate a 12 node network at the TCP level for validation purposes and obtain entirely new results that quantify the gain in TCP-level goodput using the proposed approach with respect to the shortest path only routing which is most common in the current Internet. We note that TCP flow-level simulation experiments in the context of multipath routing are rare due to the difficulty of maintaining states for a very large number of TCP flows, which we were able to achieve in this study.

• An important shortcoming of the solutions proposed in [21] and [22] is that although the average per-flow goodput increases with respect to the shortest path routing, there are a number of flows routed over SPs whose performance degrades due to starvation of service. The proposed MDRR scheduling technique addresses this starvation problem. We show that substantial performance improvements are obtained in terms of per-flow TCP goodputs using MDRR scheduling, especially for those flows that perform poorly under shortest path routing. Meanwhile, performances of flows that are already performing well with the shortest path routing are not adversely affected.

The remainder of the paper is organized as follows. In Section 2, we present our adaptive flow-level multipath routing-based traffic engineering architecture. Experimental results using ns-2 simulations are presented in Section 3. Conclusions and future work are given in the final section.

2. PATH SWITCHING ARCHITECTURE

In the proposed path switching architecture, we envision an IP backbone network which consists of edge and core nodes (i.e., routers) which has mechanisms for establishing explicitly routed paths such as MPLS. In this network, edge (ingress or egress) nodes are gateways that originate/terminate explicitly routed paths and core nodes carry only transit traffic. Edge nodes are in charge of per-egress and per-class based queuing, flow identification, rate control, and path switching as will be described in the sequel. Core nodes support per-class queuing and Explicit Congestion Notification (ECN) marking. In this architecture, flow awareness requirement is restricted to only edge nodes making the overall architecture scalable.

The proposed architecture is based on the following building blocks: (i) path establishment, (ii) queuing in network nodes, (iii) feedback mechanism and rate control, and (iv) traffic splitting, that are described next.
2.1 Path Establishment

We assume in this study that edge nodes are single-homed, i.e., they have a link to a single core node. We set up one Primary Path (PP) and one Secondary Path (SP) from each ingress node to every other egress node. We impose that the two paths are link-disjoint within the scope of the core network. The paths are computed using a two-step approach. First, the PP is 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 the PP and compute the min-hop path in the remaining network graph. A tie in this step is broken similarly. If the connectivity is lost after the first step, we do not establish an SP. We prefer to use this simple path computation scheme which does not perform any load balancing since we do not assume a-priori knowledge of the traffic demand matrix. We also note that the min-hop path may not always be the min-delay path but we are most concerned about the utilization of network links from the network operator standpoint and we therefore use the min-hop paths as the primary paths to minimize network resource use.

2.2 Queueing in Network Links

As far as queueing is concerned, all the links in the network employ a differentiated services (diffserv) mechanism, namely per-class queueing, with three drop-tail queues, namely gold, silver, and bronze queues. For scheduling, we propose to use the MDRR algorithm of [23] which is also described in [24]. In our implementation, the gold queue has strict priority over the silver and bronze queues and the gold queue is used for Resource Management (RM) and TCP ACK packets (role of RM packets will be explained in the sequel). We propose that ACK packets are identified by the ingress node and the encapsulation header for ACK packets are marked accordingly. In this paper, we assume that TCP ACK packets are not piggybacked on TCP data packets on the reverse path. If so, one could use the gold queue only for RM packets, the case of which is not studied in the current paper. Silver and bronze queues are used for TCP data packets only.

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 cascading effect is called the knock-on effect in the literature and is studied in depth for alternate routing algorithms for circuit switched networks. Preventive measures are taken to mitigate the knock-on effect for example the well-known trunk reservation concept in circuit switched networks. We address this problem in IP networks by per-class queueing mechanisms. In this paper, we compare and contrast two such methods. The first method is FIFO queueing in which all TCP data packets join the silver queue irrespective of the type of path (primary or secondary) they use. However, this queuing policy triggers the knock-on effect due to lack of preferential treatment to packets using fewer resources (i.e., traversing fewer hops). In order to mitigate the knock-on effect, longer secondary paths should be resorted to only if primary paths can no longer accommodate additional traffic and traffic flowing over longer secondary paths should not influence adversely the traffic flowing on the PPs. We therefore propose per-class queueing for all the links in the network in which TCP data packets routed over PPs use the silver queue and those using SPs join the bronze queue.

In our MDRR implementation, a Deficit Round Robin (DRR) scheduler [23] is used to arbitrate among the silver and bronze queues. In DRR, the silver queue is associated with a quantum and a deficit counter denoted by $\phi_s$ and $\delta_s$, respectively. Similarly, the bronze queue is associated with its quantum and its deficit counter, denoted by $\phi_b$ and $\delta_b$, respectively. When one of these two queues is served, DRR serves the packets at the head of queue for which the deficit counter is greater than the head-of-the-line packet's size. Each time DRR serves a queue, the deficit counter of the corresponding queue is first increased by the corresponding quantum, i.e., $\delta = \delta + \phi$. If a packet is transmitted, the deficit counter is reduced by the packet size. In this way, multiple packets can be dequeued if the total length of the packets do not exceed the deficit counter. When the visited queue is empty, the deficit counter is set to zero.

Similar to weighted fair queuing, one can assign a weight to one of the two DRR queues reflecting the relative bandwidth allocated to the queue, and each quantum is chosen to be proportional with the weight of the corresponding queue. By setting the weight of the silver queue large enough with respect to that of the bronze queue, one can ensure preferential treatment to IP packets using the silver queue, i.e., PP packets, thus mitigating the knock-on effect. Moreover, generally the quantum should not be smaller than the maximum transmission unit (MTU) of the interface so that at least a packet from a non-empty queue can be dequeued in one visit. On the other hand, in our MDRR implementation, the gold queue packets have priority over the
silver and bronze queues and we use the alternate priority mode of the MDRR scheduler described in [23]. In this mode, the gold queue (or the low latency queue) is serviced in between the other queues and at each visit, all packets residing at the gold queue are dequeued. The serving order of our MDRR implementation is then Gold, Silver, Gold, Bronze, Gold, Silver, ... .

In our particular implementation of the MDRR scheduler, each time the bronze queue is served when non-empty, its quantum \( \phi_b \) is set to the length of the head of line packet and the quantum \( \phi_s \) is set to the product of \( \phi_b \) and the ratio of the configured weight of the silver queue to that of the bronze queue. Otherwise, the quantum sizes are not modified. This policy ensures that at least a packet from the bronze queue will be dequeued at each visit. For example, in this paper the silver queue will be assumed to have a weight of 90% and the bronze queue a weight of 10%. Therefore, the size of the head of line packet in the bronze queue gives us the quantum for the bronze queue and when multiplied by 9 (90/10) gives the quantum size for the silver queue for the underlying MDRR scheduler.

2.3 Feedback Mechanism and Rate Control

Another building block of the proposed architecture is the feedback mechanism and rate control. For intelligent path switching, the current status of the two paths needs to be available at the path switching entity. This information can be provided using path probing mechanisms without network assistance by measuring delay, loss, and other attributes of the paths. On the contrary, we assume network assistance in this paper and 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 towards the network every \( T_{rm} \) seconds with the direction bit set to indicate the direction of flow. If DRR is used and when a 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 Congestion Experienced (CE) 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 node after resetting the direction bit of the RM packet. RM packets travelling over the reverse path are not marked 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 Allowed Transmission Rate (ATR) of the corresponding rate-controlled path by using the Additive Increase Multiplicative Decrease (AIMD) algorithm given in Table 1. In this algorithm, MTR and PTR denote the Minimum and Peak Transmission Rates, and RDF and RIF denote the Rate Decrease and Rate Increase Factors, respectively.

The proposed architecture is depicted in Figure 1 for an example 3-node core network in which solid lines correspond to PPs whereas the dotted lines denote SPs originating at ingress node 0. For the sake of simplicity, the internal architecture of only ingress node 0 is given which maintains two queues for each egress for rate control purposes at the per-egress queueing stage, one for the PP and the other for the SP. Both queues are drained at a rate dictated by the corresponding ATR obtained through the algorithm of Table 1. IP packets leaving the per-egress queueing stage join their respective queues in the per-class queueing stage for the edge-core link. Also note that the per-class queueing stage is not limited to edge-core links but to all links in this network including core-core links.

Finally, we assume that the switching technology in the core network has the necessary fields in the encapsulation header for implementing the above-mentioned mechanisms and in particular a field for indicating whether the packet is: i) forward RM packet for a PP, ii) backward RM packet for a PP, iii) forward RM packet for an SP, iv) backward RM packet for an SP, v) TCP data packet for a PP, vi) TCP data packet for an SP, or vii) TCP ACK packet.

2.4 Path Switching

The final ingredient to the proposed approach is the way path switching is performed for an incoming TCP flow. The edge nodes first identify new flows. The delay estimates for the PP and SP queues (denoted by \( D_{pp} \) and \( D_{sp} \), respectively) at the edge nodes are then calculated by dividing the occupancy of the corresponding
For Peer Review

Figure 1. The proposed architecture for path switching.

Table 1. AIMD algorithm implemented for each path at the ingress node.

per-egress queue by the current drain rate (ATR). Upon the arrival of the first packet of the \( n \)-th flow (i.e., a TCP SYN segment), a running estimate of the delay difference (denoted by \( \Delta_n \)) is calculated as

\[
\Delta_n = \beta(D_{pp} - D_{sp}) + (1 - \beta)\Delta_{n-1},
\]

where \( \beta \) is a smoothing parameter. If \( \Delta_n \leq \min_{th} \) (\( \Delta_n \geq \max_{th} \)), then the flow is switched towards the PP (SP). When \( \max_{th} > \Delta_n > \min_{th} \), the new flow is switched towards the SP with probability \( p_0 \) and towards the PP with probability \( 1 - p_0 \), as \( p_0 \) is given by

\[
p_0 = \frac{\Delta_n - \min_{th}}{\max_{th} - \min_{th}}
\]
queues is controlled by RER. RER parameters are generally chosen so that the PP is favored, i.e., \( \min_{th} > 0 \) and we employ proportional control as opposed to on-off type control, i.e., \( \max_{th} > \min_{th} \). The reason for proposing RER is that longer secondary paths not only lead to knock-on effects but also reduce TCP level goodput stemming from increased RTTs. Moreover, once a path decision is made for the first packet of a flow, all the remaining packets of the same TCP flow will follow the same path. Therefore, the secondary path should be far better than the primary path already in terms of edge queueing delays (by suitable selection of RER parameters) to justify a decision to switch to the secondary path particularly for long flows. On the other hand, the reason to use proportional control is to avoid an oscillatory response and break synchronization among contending flows.

3. SIMULATION STUDY

We obtain the simulation results using the network-assisted path switching architecture by simulating (with ns-2) IP/MPLS networks with enhanced ECN capability and modified MPLS header that accommodates a field to indicate packet type (a)-(g) which is necessary for proper operation.

3.1 Topology and Traffic Matrix

We study two topologies in this paper, a small 3-node triangle core network depicted in Figure 1 and a medium sized 12-node network available in the public domain [25], which is depicted in Figure 2. For the 3-node network, core nodes are connected by links each with a capacity of 50 Mbps, and each link has a propagation delay of 10 ms. There is an edge node connected via a 1 Gbps link to each core node therefore the core links are the potential bottleneck links in the 3-node network. The average traffic sourced out of each edge node is fixed at 70 Mbps. The traffic demand \( T_{i,i+1} \) from node \( i \) to node \( i+1 (mod\,3), i = 1,2,3, \) is uniformly distributed in the interval (0,70) Mbps and that of \( T_{i,i-1} \) is set to 70 Mbps \(-T_{i,i+1} \). The distribution of traffic sourced out of each node is independent of others. 50 random traffic demand matrices are generated as inputs to the simulation study based on the above statistical characterization. The goodput results are obtained from the collection of 50 independent runs each with its own randomly drawn traffic matrix. On the other hand, the 12-node network we study is known as the hypothetic US topology which is available in [25] together with the link capacities and the traffic demand matrix. This topology has 19 links and has been used for several optimal multipath studies. In the original network, all network links have a capacity of 155 Mbps except for the two bidirectional links de ↔ ch and ch ↔ cl which have a capacity of 310 Mbps in both directions. In order to keep the memory requirements of the simulator to a reasonable level, we reduced all the link capacities by a factor of 1/3 as well as all the traffic demands so as to work with a scaled down version of the original network but operating at the same load.

![Hypothetic US Topology](http://mc.manuscriptcentral.com/ett)

Figure 2. Hypothetic US Topology as given in [25].
3.2 Traffic Model

Given a traffic demand in Mbps between a source-destination pair, we need to have a TCP traffic model that generates traffic according to the traffic demand. Motivated by recent studies on Internet traffic modeling, in our simulations we assume that flow arrivals occur according to a Poisson process with rate $\lambda$, and flow sizes obey a Bounded Pareto (BP) distribution. The BP distribution is preferred in this study rather than the ordinary Pareto distribution since the latter distribution has infinite variance, and thus excessively long simulations would be required for convergence. Moreover, the BP distribution still exhibits the large variance and heavy tail properties of the flow size distribution of Internet traffic and allows us to set a bound on the largest flow size. The bounded Pareto distribution is denoted by $BP(k, p, \alpha)$, where $k$ and $p$ denote the minimum and maximum flow sizes, respectively, and the shape parameter $\alpha$ is the exponent of the power law.

As $\alpha$ gets smaller the tail gets heavier, and the likelihood of having large flows increases. The probability density function for the $BP(k, p, \alpha)$ is given by

$$f(x) = \frac{\alpha k^\alpha}{1-(k/p)^\alpha} x^{-\alpha-1}, k \leq x \leq p, 0 \leq \alpha \leq 2$$  \hspace{1cm} (3)

In this study, we use $k = 4$ KBytes, $p = 50$ MBytes, and $\alpha = 1.20$, corresponding to an average flow size of 20,362 Bytes.

3.3 Routing Policies

In order to validate the effectiveness of the path switching idea introduced in this paper, we compare and contrast several routing policies using simulations. As a reference policy, we use rate-controlled single path routing in which the traffic follows the minimum-hop route with the ECN and AIMD capabilities turned on but with no path switching. In short, we call this method “single path routing”. It would be desirable to have a path switching mechanism whose performance surpasses that of single-path routing in all possible scenarios. Viability of such a mechanism is the scope of the current simulation study.

When path switching is used, two further policies are used to handle the implications of the knock-on effect: i) RER used for path switching at the edge node, ii) Queueing and scheduling used in the network links.

In this paper, we study two sets of RER parameters:

- $min_{th} = max_{th} = 0$ ms and $p_0 = 1$, called the Shortest Delay (SD) policy since the path with the shorter smoothed average buffering delay is chosen when SD policy is employed. The SD policy switches each new TCP flow simply to the path with the shorter estimated buffering delay at the ingress edge node, and therefore the SD policy does not favor the PP in path switching.

- $min_{th} = 1$ ms, $max_{th} = 15$ ms and $p_0 = 1$, called the RER policy; this choice of RER parameters fits well with the philosophy of RER discussed in Section 2 that favors the min-hop path over the longer SP.

For RER, we fix the delay averaging parameter as $\beta = 0.3$. For the queueing and scheduling policies in the network links, we consider the following three alternatives:

- **FIFO queueing** in which all TCP data packets (PP and SP) join a single queue, say the silver queue, and RM and TCP ACK packets use the gold queue.

- **Strict Priority Queueing (SPQ)** in which we have the gold, silver, and bronze queues used by RM/TCP ACK packets, PP TCP data packets, and SP TCP data packets, respectively, drained according to an SPQ policy with the gold (bronze) queue having the highest (lowest) priority.

- **MDRR scheduling** as explained in detail in Section 2.2 with 90% weight for the silver queue and 10% weight for the bronze queue. The reason we use MDRR as opposed to SPQ is that with SPQ some bronze queues in the network may get very close to zero service rate, starving the SPs using those queues. However, for probing mechanisms to work, we propose that a small weight (e.g., 10%) is to be assigned to the bronze queue to remedy this situation.

In this paper, we consider the following three combined policies comprising different queueing/scheduling and RER mechanisms, namely FIFO/SD, SPQ/RER and MDRR/RER.
### 3.4 IP/MPLS and Simulation Parameters

Data packets are assumed to be 1040 Bytes long including the MPLS header. We assume that the RM packets are 50 Bytes long. All the buffers at the edge and core nodes including per-egress and per-class queues have a size of 104,000 Bytes each. The TCP receive buffer is of length 20,000 Bytes. This proposed path switching architecture is implemented over ns-2 (Network Simulator) version 2.27 [14] and the TCP-Reno is used in the simulations. The simulation runtime is selected as 300 s. In all the simulation results, events concerning flow arrivals occurring only in the period [95 s, 250 s] are reported in order to avoid transient effects.

### 3.5 AIMD Parameters

The frequency of probe packets is chosen such that \( T_{RM} = 0.1 \) s for the 3-node network, and \( T_{RM} = 0.02 \) s for the 12-node network. We use RDF = 0.0625 in all simulations. For the 3-node network, RIF = 0.125 for PPs and SPs in FIFO/SD and PPs in single-path routing and MDRR/RER. The MDRR weight ratio of the silver and bronze queues in MDRR/RER is 9:1, so in case of congestion due to PP flows, the maximum drain rate of bronze queues is limited to only 10% of the link bandwidth. Gradual traffic increase by the AIMD algorithm of SPs must be in smaller steps in order to prevent buffer overflows in bronze queues in case their drain rate is limited to 10% of the link speed. Therefore, the SPs in MDRR/RER are proposed to have RIF = 0.0138. For the 12-node network, RIF = 0.0625 for PPs and SPs in FIFO/SD and PPs in single-path routing and MDRR/RER. SPs in MDRR/RER have RIF = 0.00694 due to the DRR weight ratio as explained above.

One of the problems of ECN based congestion detection algorithms is ACR beat down problem [28]. When a path between a source-destination (s-d) pair goes through several congested switches, CI bit of this s-d pair’s RM packets is marked more often than other s-d pairs having paths going through fewer number of congested switches, causing unfairness among s-d pairs. It is possible to improve fairness for PPs in single-path routing and for PPs and SPs in DRR/RER by properly setting MTR. MTR can be calculated separately for PPs and SPs in DRR/RER. Each ingress node calculates PP MTR (SP MTR) value for each link along PP (SP) as \( MTR = CBW \times \frac{W}{N} \), where BW is the bandwidth of the link, \( W \) is the weight of the bronze (silver) queue, \( N \) is the number of PPs (SPs) using the link and \( C < 1 \) is a constant. For MDRR/RER, \( W = 0.9 \) for the silver queue and \( W = 0.1 \) for the bronze queue. For single-path routing, \( W = 1 \) for the silver queue. Each s-d pair compares the PP MTR (SP MTR) values of links along the PP (SP) and chooses the minimum value as its MTR for PP (SP). For PPs and SPs when FIFO/SD is used, MTR = 1 bps is chosen in order to eliminate division by zero. PTR is chosen as the speed of the slowest link on the corresponding path. The constant \( C = 0.9 \) is chosen in the simulations. The percentage queue occupancy threshold \( \mu \) for setting the CE bit is set to \( \mu = 50\% \) for the 3-node network and \( \mu = 20\% \) for the 12-node network.

### 3.6 Performance Metrics

The goodput of a TCP flow \( i \cdot G_i \), is defined as the service rate received by flow \( i \) during its lifetime or equivalently it is the ratio \( \Delta_i / T_i \), where \( \Delta_i \) is the number of bits successfully delivered to the application layer by the TCP receiver and \( T_i \) is the sojourn time of the flow \( i \) within the simulation runtime. We note that if flow \( i \) terminates within the simulation runtime then \( \Delta_i \) is equal to the flow size \( S_i \). Note that some flows may not be entirely carried or may not even be admitted (i.e., SYN packets dropped) during the simulation duration due to overloading of certain links in the network. We therefore introduce a performance measure, called the net average goodput over all TCP flows, denoted by \( G_{net} = \sum_i \Delta_i / S_i \), where the service rate for uncarried packets is set to zero. \( G_{net} \) can be computed for an individual source-destination pair or for the network as a whole. For the same effect, we suggest another measure, called the Byte Incompletion Ratio (BIR), to quantify the portion of data that cannot be delivered within the simulation duration, in percentage. An incomplete packet (byte) means a generated packet (byte) within the simulation duration but has not been received yet by the intended receiver within the simulation duration. Mathematically,

\[
BIR = \frac{\sum_{s,d} (N(s,d) - \Gamma(s,d))}{\sum_{s,d} N(s,d)} \cdot 100, \tag{4}
\]
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 total traffic (in bytes) successfully delivered to the application layer from node $s$ to node $d$ within the simulation runtime. BIR is indicative of the ratio of flows that receive a small goodput (leading to larger delays) to the overall number of flows and also of the amount of in-transit traffic at the epoch of simulation completion.

### 3.7 Simulation Results

We first study the effect of the RER parameters $\min_{th}$ and $\max_{th}$ on TCP goodput and BIR. The network-wide average goodput per TCP flow and BIR for the 12-node mesh network are shown in Figure 3. Based on these results, in our simulation studies, we propose to use $\min_{th} = 1$ms and $\max_{th} = 15$ ms. It can be observed that both performance metrics have a smooth behavior around the selected values of the parameters. However, when these parameters are chosen close to zero (mimicking the SD policy), the goodputs drop considerably as an outcome of the knock-on effect and this particular regime needs to be avoided by the operator. More work is needed to tune the RER parameters but our preliminary results show that the performance of the system is quite robust with respect to our choice of the specific RER parameters.

![Figure 3. The network-wide goodput and BIR as a function of $\min_{th}$ and $\max_{th}$](image)

First we compare SPQ/RER and MDRR/RER. The goodputs received by the PPs and the SPs for MDRR/RER and SPQ/RER for the 3-node and 12-node topologies are depicted in Figure 4. Note that the obtained goodputs are over the 50 independent runs and for all pairs simulated for the 3-node network whereas we have a single run for the 12-node network. It is observed that in SPQ, some flows receive lower goodput when forwarded over the SP compared to PP. If SP is passing through a congested link, SPQ causes ATR of the SP to get close to zero. In this case, RER algorithm may give incorrect routing decisions and route more flows to SPs resulting in flow starvation. On the other hand, MDRR/RER assigns a non-zero weight to the bronze queue eliminating the possibility of starvation of SPs. As seen in the figures, with MDRR/RER, the goodput ratio of the PP and the SP becomes more stable especially for slow s-d pairs. The path goodput ratio becomes smaller for high goodput s-d pairs, however this is less important since the number of flows forwarded to SP is small for high goodput s-d pairs due to RER. For the rest of the paper, we use MDRR/RER in order to mitigate the flow starvation problem instead of SPQ/RER which was used in our earlier work [21].

The impact of the architectural constructs RER and MDRR proposed in this study is studied in terms of their ability to mitigate the knock-on effect. MDRR/RER policy, where the MDRR method is used in network links for queueing and scheduling and RER is used at the edges for intelligent path switching, is compared with single-path routing and FIFO/SD schemes. We present the cumulative histogram diagram in Figure 5 showing the fraction of source-destination pairs receiving an average goodput $G_{av}$ less than a certain value in Kbps with the x-axis using logarithmic scale. The cumulative histograms are given for both the 3-node (using 50 randomly generated traffic matrices) and 12-node networks (using the given traffic demand matrix, where the histogram is obtained using the goodput distribution over node pairs).
First consider the 3-node case. With respect to single-path routing, SD/FIFO treats favorably the low goodput node pairs but while doing this high goodput pairs are seriously harmed, resulting in reduction of overall average goodput. We observe that when the goodputs increase, the performance of the primary and secondary paths of FIFO/SD simultaneously deteriorate which can be explained by the knock-on effect stemming from FIFO queuing and SD type path switching. We show that it is feasible to favor low goodput pairs without having to jeopardize high goodput pairs by using the MDRR/RER method which takes the best of the two worlds, namely that of FIFO/SD and single-path routing. We note that the MDRR method consistently outperforms the single-path routing where the performance improvement is magnified for the cases where the shortest path is congested and TCP flows using congested shortest paths in single-path routing receive low goodputs. The fact that these results are obtained for a set of 50 randomly generated traffic matrices is indicative of the robustness and resilience of the proposed path switching architecture. Similar conclusions can be drawn for the 12-node case for which we ran a single instance of a traffic matrix that was obtained out of a publicly available test case and a handful of node pairs took advantage of path switching for this particular scenario.

We present the gain in per node pair goodputs via the use of MDRR/RER and FIFO/SD with respect to the reference single path routing in Figure 6. The $x$ and $y$ coordinates of each marked point in this figure correspond to the goodput obtained with single path routing for a certain node pair and the goodput gain in using MDRR/RER or FIFO/SD for the same pair, respectively. Consider first the 3-node case and MDRR/RER. All the marked points are located in a right triangle whose bottom side coincides with the unity (or no) gain line, which shows that MDRR/RER consistently outperforms single-path routing. There are a
number of marked points along this horizontal side implying that path switching does not introduce any gain for those pairs but does not worsen the results either. On the other hand, goodput gains up to two orders of magnitude are observed for some node pairs. Most importantly, the maximum gain is achieved for node pairs for which the single path routing produced the lowest per-pair goodput (along the vertical side of the right triangle). There are also a number of marked points along the hypotenuse of the right triangle, which shows that gains using MDRR/RER drop as goodputs achieved by single path routing increase. Other marked points lie inside the right triangle revealing gains with varying intensity. On the contrary, the FIFO/SD approach yields marked points inside a parallelogram rather than a triangle implying goodput losses with respect to single path routing for many node pairs, especially for those with high goodputs. This parallelogram type behavior of FIFO/SD is a consequence of the knock-on effect. Similar conclusions can be drawn for the 12-node network.

In order to demonstrate the effect of the total amount of the traffic demand, the traffic is scaled for the 12-node network by scaling down the given traffic demand matrix with a traffic scaling parameter $\gamma \leq 1$. We observe from Figure 7(a) that the network-wide $G_{net}$ of MDRR/RER and single path routing closely follow each other as $\gamma$ changes, but the performance of FIFO/SD is far lower. We do not observe a crucial difference in network-wide goodput between MDRR/RER and single path routing since MDRR/RER improves mainly the performance of some low goodput s-d pairs that are limited in number. When $\gamma = 1$, single path routing even gives a slightly better network-wide $G_{net}$ than DRR/RER. Revisiting Figure 6(b), we observe that the goodputs of low goodput s-d pairs are improved up to 15 times while the goodputs of some of high goodput node pairs may reduce by 0-15% due to the choice of MDRR weights that slightly favor longer paths. The fact that high goodput node pairs dominate in the network-wide average goodput calculations explains the slight outperformance of single path routing over MDRR/RER. In Figure 7(b), the performance improvement for low goodput flows with MDRR/RER becomes more apparent since only low goodput flows might have a BIR larger than zero. We observe that BIR of SD/FIFO and MDRR/RER are close to each other and they both reveal a significant improvement over BIR of single path routing. It is also clear that when the traffic demand is increased, more TCP flows get bottlenecked and BIR increases. As a summary, MDRR/RER takes the best of the two worlds by improving upon the BIR for low-speed TCP flows as in SD/FIFO but without having to hamper high-speed TCP flows as in single path routing.

The ratios of total data transmitted over PP and SP for each flow using MDRR/RER and FIFO/SD are shown in Figure 8 as a function of the goodput received by the flow. These figures show that MDRR/RER controls the ratio of amount of data sent to PP and SP according to the performance of paths. It uses SP only if PP is congested and SP is better. A substantial portion of the data is transmitted through PP if the goodput achieved through PP is high. On the other hand, SD/FIFO routes flows to the SP regardless of the PP goodput resulting in unnecessary forwarding of flows to the SP. Since SPs are typically longer and use more resources, aggressive use of SPs induces the knock-on effect.

4. CONCLUSIONS

In this paper, we propose a flow-based distributed path switching mechanism for traffic engineering TCP flows in networks with explicit routing capability such as MPLS. In this architecture, TCP traffic is split at the flow level between the primary and secondary paths using a random early rerouting principle that controls the queuing delay difference between the two alternative paths. We propose ECN marking and AIMD-based rate control so that the traffic splitting decisions can be made at the flow-aware edges of the MPLS network. Moreover, we propose a modified deficit round robin queueing and scheduling mechanism not for QoS delivery but for favoring shortest paths and thus improving routing performance. Using Poisson TCP flow arrivals and bounded Pareto distributed flow sizes, we show that it is possible to substantially improve the performance of low-speed TCP flows with network-assisted path switching without having to jeopardize the performances of already high-speed TCP flows. Consistent outperformance against single-path routing and substantial per-flow goodput gains under poor primary path conditions are the main outcomes of this study.

We tend to believe that prioritized routing, flow-aware networking, and switching decisions that favor min-hop paths are the minimal requirements for success for an adaptive multipath routing technique to be useful. Simulation of larger topologies with a more diversified set of traffic demand matrices, use of more than two paths, and flowlet switching especially for long flows are considered as future work. Another future work is to use probing based bandwidth estimation techniques (rather than network assistance that uses RMs and ECN capability in the core) to reduce implementation complexity.
Figure 6. The gain in per-node pair goodputs via the use of DRR/RER and FIFO/SD algorithms with respect to single path routing.

Figure 7. The network-wide goodput and BIR as a function of the scaling parameter $\gamma$.

Figure 8. The ratio of total data transmitted over the primary path PP and the secondary path SP.

REFERENCES


O. Alparslan, N. Akar and E. Karasan, “Combined use of prioritized AIMD and flow-based traffic splitting for robust TCP load balancing,” Fifth International Workshop on Quality of future Internet Services (QofIS’04), Barcelona, Spain, July 2004.


The Network Simulator – ns-2, developed by L. Berkeley Network Laboratory and University of California Berkeley, http://www.isi.edu/nsnam/ns