AIMD-Based Online MPLS Traffic Engineering for TCP Flows via Distributed Multi-Path Routing

Onur Alparslan, Nail Akar, and Ezhan Karasan

Abstract

In this paper, we propose a distributed online traffic engineering architecture for MPLS networks. In this architecture, a primary and secondary MPLS LSP are established from an ingress LSR to every other egress LSR. We propose to split the aggregate TCP traffic between the primary and secondary paths using a distributed mechanism based on ECN marking and AIMD-based rate control. Inspired by the random early discard mechanisms for active queue management, we propose a random early reroute scheme to adaptively control the delay difference between the primary and secondary LSPs. Considering the adverse effect of packet reordering on TCP performance for packet-based load balancing schemes, we propose that the TCP splitting mechanism operates on a per-flow basis. Using flow-based models developed for Internet traffic and simulations, we show that flow-based distributed multi-path traffic engineering outperforms on a consistent basis the case of a single path in terms of per-flow goodputs. Due to the elimination of out-of-order packet arrivals, flow-based splitting also enhances TCP performance with respect to packet-based splitting especially for long TCP flows that are hit hard by packet reordering. We also compare and contrast two queuing architectures for differential treatment of data packets routed over primary and secondary LSPs in the MPLS data plane, namely first-in-first-out and strict priority queuing. We show through simulations that strict priority queuing is more effective and relatively more robust with respect to the changes in the traffic demand matrix than first-in-first-out queuing in the context of distributed multi-path routing.

Keywords: Multipath routing; MPLS; traffic engineering; TCP; AIMD rate control

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

O. Alparslan, N. Akar, and E. Karasan are with the Electrical and Electronics Engineering Department, Bilkent University, Ankara, Turkey.

This work is supported in part by the Science and Research Council of Turkey (Tübitak) under projects EEEAG-101E048 and EEEAG-101E025.

1. Introduction

Internet Traffic Engineering (TE) is defined as the set of mechanisms required for enhancing the resource utilization of an operational network. In particular, traffic engineering controls how traffic flows through a network so as to optimize resource utilization and network performance [1]. TE mechanisms can be applied to hop-by-hop, explicit, or multi-path routed networks.

Traditional hop-by-hop routed IP networks using OSPF or IS-IS routing protocols resort to shortest path routing with simple link weights such as hop-count or delay. 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. A number of research studies have recently been carried out on traffic engineering in hop-by-hop routed networks. For a given traffic demand matrix, these studies seek an optimal set of link weights to improve the routing performance and these link metrics are computed using a centralized optimization algorithm [2-4]. Once computed, the link metrics can be configured into the core IP routers either manually or automatically. We note that an accurate estimate of the traffic demand matrix should be available for this approach for acceptable performance, which is generally hard to obtain [5]. This approach is effective particularly when the traffic matrix does not change significantly in short time scales [6]. A robust extension of this approach that can cope with failures is also available in [7].

In the overlay approach, service providers establish logical connections between the edge nodes of a backbone, and then overlay these logical connections onto the physical topology. The hop-by-hop routing paradigm using shortest paths is overridden in this approach since logical connections can take any feasible path through the network. The emergence of Multi-Protocol Label Switching (MPLS) technology provides the necessary protocols and mechanisms in IP backbones for explicit routing to facilitate traffic engineering [1,8]. In MPLS backbones, a constraint-based routing scheme can be used so that the traffic may be controlled to flow optimally through certain routes [9,10]. In the general overlay approach, typically an initial logical connection layout is obtained for traffic engineering using a long-term traffic matrix and constraint-based routing. The bandwidth allocated to each virtual connection in this layout is directly related to the actual long-term traffic between the end points of the virtual connection. However, the actual traffic may occasionally deviate from the long-term value and this deviation generally cannot be predicted in advance. When there is a significant traffic increase in the actual traffic for a certain logical connection, additional bandwidth allocation will be needed, and the network will be signaled for an increase in the allocated bandwidth. If extra bandwidth is available, this request will be accepted and an additional allocation will be made. Otherwise, a new constraint-based route will be sought and if found, the original logical connection will be torn down and a new logical connection will be established. When there is a significant traffic reduction, the network will be signaled for the deallocation. A crucial performance impacting issue in such TE mechanisms is the determination of the amount of traffic change required to initiate such a signaling process.

Another TE approach that we focus on in the current paper is "multi-path routing," the goal of which is to improve the utilization of resources of an underlying physical network by providing multiple paths between source-destination (s-d) pairs. In the multi-path overlay approach, multiple logical connections with disjoint paths are established between the two end points of a network. These paths can be determined by using the long-term traffic demand. The goal of multi-path overlay traffic engineering is to increase the resource utilization of the network by intelligently splitting the traffic between an s-d pair among multiple alternative logical connections. The work in [11] proposes a dynamic multi-path routing algorithm in connection-oriented networks where the shortest path is used under light traffic conditions and multiple paths are used as the shortest path becomes congested. The adaptive multi-path approach, proposed in [12-15], considers a general cognitive packet network carrying smart, dumb, and acknowledgment packets. In this approach, smart packets explore and learn optimal routes using reinforcement learning in an adaptive manner and dumb packets that carry actual payload, follow these learned routes. Multi-path routing is studied in the context of wireless networks as well; a distributed multi-path routing scheme that selects a network path with sufficient resources in a dynamic multi-hop mobile setting is described in [16].

Recently, there have been a number of multi-path traffic engineering proposals specifically for MPLS networks that are amenable to distributed online implementation. In [6], probe packets are transmitted periodically to the egress Label Switch Router (LSR), which then returns them back to the ingress LSR. Based on the information in the returning probe packets, the ingress LSR computes the one-way congestion statistics that can be delay or loss, and uses a gradient projection algorithm for load balancing. In the model [6], all paths between an s-d pair are equally treated which may be problematic in scenarios for which some paths may have significantly longer hop lengths than their corresponding min-hop paths. On the other hand, Additive Increase/Multiplicative Decrease (AIMD) feedback algorithms are used generally for flow and congestion control in computer and communication networks [17,18]. The multi-path AIMD-based approach of [19] uses binary feedback information regarding the congestion state of the

LSPs and a traffic splitting heuristic using AIMD is proposed which ensures that source LSRs never send traffic to secondary paths of longer length before they make full use of their primary paths.

A challenging issue in multi-path routing is the potential de-sequencing (or reordering) of packets through the network since packets will occasionally take different routes with different delays. A queuing analysis of packet de-sequencing (or reordering) and required re-sequencing at the receiver site was carried out in [20] to show the impact of de-sequencing on total delays. The majority of the traffic in the current Internet is TCP-based and the impact of de-sequencing of TCP packets on application layer performance is crucial. However, TCP receiver behavior is quite complex and varies from one implementation to another and is not amenable to a stochastic analysis as described in [20]. However, experimental studies demonstrate that packet de-sequencing within a TCP flow can significantly deteriorate TCP performance [21,22]. When the traffic is split in a static manner (i.e., splitting ratios are fixed over time), hashing-based splitting schemes are shown to be effective in terms of both scalability and de-sequencing performance [21]. In dynamic traffic splitting, the splitting ratios change adaptively over time with the changing congestion status of the corresponding alternative paths. Flow-based multi-path routing is a dynamic splitting scheme that operates on a per-flow basis with the aim of avoiding packet de-sequencing within a flow; see for example [23,24] for related work. We refer the reader to [25,26] for flow-based multi-path routing with emphasis on routing of elastic flows like TCP flows. Flow-based routing in the QoS routing context in MPLS networks is described in [27,28], but these studies require a flow aware core network and therefore have scalability problems with increasing number of instantaneous flows [27].

Recently, a new flow-based multi-path traffic engineering approach for best-effort networks is proposed in [29]; this reordering-free architecture is tested for UDP traffic, and its performance improvements relative to the case of non-flow based multi-path routing and that of single path are reported. The approach imposes flow awareness only at the edge devices and therefore does not have the scalability problems of other flow-based QoS routing schemes. In the current paper, we extend the work in [29] towards a reordering-free flow-based traffic engineering architecture for TCP traffic in MPLS backbones. In this architecture, two link disjoint MPLS LSPs, one being the primary Label Switched Path (P-LSP) and the latter being the secondary LSP (S-LSP), are established between LSR pairs between which there is direct TCP traffic. The proposed architecture is amenable to extension to arbitrary number of paths but this generalization is intentionally left outside the scope of the current paper to improve its readability. After establishing these two LSPs, we develop an algorithm that splits the traffic between the two LSPs in order to improve the overall throughput. Motivated by the ABR (Available Bit Rate) service category used for flow control in ATM networks, we proposed an explicit rate feedback mechanism for traffic engineering purposes in MPLS networks in [29]. Since explicit rate feedback is not an MPLS standard, we propose in this paper, a binary feedback mechanism that can be implemented with the help of standards-based ECN (Explicit Congestion Notification)-capable LSRs with little additional complexity. In the proposed mechanism, edge LSRs maintain two drop-tail queues, one for the P-LSP and one for the S-LSP. These queues are drained using AIMD triggered by the binary feedback mechanism.

In the proposed flow-based multi-path traffic engineering architecture, a traffic splitting algorithm is proposed in which individual traffic flows are identified and probabilistically assigned to one of the two LSPs based on the average difference between the delays in the corresponding queues. The algorithm we propose for traffic splitting is called RER (Random Early Reroute), which is inspired by the RED (Random Early Discard) algorithm used for active queue management in the Internet. The flow-based mechanism ensures that packet reordering would not take place at the receiving end of the corresponding flow.

One other important issue in dynamic load-balancing is the design of the MPLS data plane in terms of its queuing architecture. 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 [30]. If the alternative paths use more resources (or hops) on the average, some improperly designed load balancing algorithms may perform poorly even relative to a scheme that uses a single path for every s-d pair. This fact is called the knock-on effect in literature, and precautions should be taken to minimize this effect [30]. Inspired by the well-known principles of dynamic routing in circuit switched networks, a trunk reservation based approach is presented to deal with this effect in [26]. We take a different approach in this paper for the same problem using per-class queuing. In [29], we proposed a queuing architecture in the MPLS data plane that favors packets of P-LSPs over those of S-LSPs in Strict Priority (SP) sense to cope with the knock-on effect. In this paper, we compare and contrast the SP and the widely deployed First-In-First-Out (FIFO) queuing architectures in the TCP load-balancing context and show through simulations that SP queuing is more effective and relatively more robust with respect to the changes in the traffic demand matrix than FIFO queuing.

The remainder of the paper is organized as follows. In Section 2, we present our traffic engineering architecture. Our simulation results are presented in Section 3 and conclusions and future work are provided in the final section.

2. Architecture

In this study, we envision an MPLS network consisting of edge and core LSRs depicted in Fig. 1. TCP/IP traffic is sourced from and destined to the edge LSRs while the core LSRs carry only transit traffic. Flow classification, traffic splitting, per-destination queuing and rate control, and the conventional edge MPLS functionalities belong to the edge LSRs in the proposed architecture. The core LSRs are not flow aware, and their sole responsibility is label switching, per-class queuing, and ECN marking. Consequently, the proposed architecture has the potential to scale to large networks since flow awareness and flow processing capabilities are required only at the edge LSRs, but not on the core LSRs where scalability requirements are elaborate.

Our proposed distributed on-line traffic engineering architecture is comprised of the following three main components: (i) LSP establishment and queuing model used in core LSRs, (ii) feedback mechanism and rate control, (iii) traffic splitting algorithm. We now study each of these components and their interactions.

We propose in this study that the core LSRs employ output queuing and they support differentiated services (diffserv) with the gold, silver, and bronze services (i.e., Olympic services). These services can be implemented with per-class queuing with three drop-tail queues, namely gold, silver, and bronze queues, at each outgoing physical interface. In terms of scheduling, we propose that the gold queue has strict priority over the silver queue, which has strict priority over the bronze queue. In our proposed architecture, the gold service is dedicated not only to Resource Management (RM) packets (their role will be described in detail later) used for gathering binary congestion status from the network but for TCP ACK (i.e., acknowledgment) packets as well. The motivation behind this choice is to provide prompt feedback to TCP end users, and also to traffic splitters deployed at edge LSRs. How the remaining silver and bronze services will be used for TCP data packets are based on the way LSPs are established, which is described below.

We assume in this study that edge LSRs are single-homed, i.e., they have a link to a single core LSR. Two LSPs, which are link disjoint in the core MPLS network, are then setup from an ingress LSR to every other edge LSR for which there is direct TCP/IP traffic. If edge LSRs are multi-homed then the "link disjointness" requirement could be imposed on the entire MPLS cloud including the edge LSRs; we do not study multi-homing in the current paper. The P-LSP uses the minimum hop path found using Dijkstra's algorithm. When there is a tie in the algorithm, this tie is broken randomly. The route for the S-LSP is found by pruning the links in the core MPLS network used by the P-LSP and randomly choosing one of the minimum hop paths in the remaining network graph. For an example, we refer the reader to Fig. 1 in which the P-LSP and S-LSP traverse two and three hops, respectively, in the core network. If the connectivity is lost after pruning the links from the graph, the secondary LSP is not established. We note that if an accurate estimate of the traffic demand matrix is known a-priori, more sophisticated algorithms might be used to select the routes LSPs take. However, we do not assume a-priori knowledge on traffic demands in this study.

There are two queuing models based on the work in [29] that we study in this paper. In FIFO queuing, data packets of P-LSPs and S-LSPs join the same silver queue and we do not make use of the bronze queue at all. In other words, in FIFO queuing, there is no preferential treatment to packets using fewer resources (i.e., traversing fewer hops). However, this policy might carry the risk of promoting the knock-on effect, i.e., 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. We note that secondary paths use more resources (i.e., longer hop lengths) and they should be resorted to only when the primary paths can no longer accommodate additional TCP traffic. Based on the work described in [29], we propose Strict Priority (SP) queuing in which TCP data packets belonging to P-LSPs and s-LSPs use the silver and bronze services, respectively. We note that the above-mentioned queuing models are implementable using the standards-based E-LSP (EXP-inferred-PSC LSP) method by which the three-bit experimental (EXP) field in the MPLS header is used by edge LSRs to code the PSC (Per Hop Behavior Scheduling Class) [31]. We propose that the EXP bits are devoted to marking the packet as a

(A1) Forward RM packet for a P-LSP,
(A2) Backward RM packet for a P-LSP,
(B1) Forward RM packet for an S-LSP,
(B2) Backward RM packet for an S-LSP,
(C) TCP data packet for a P-LSP,
(D) TCP data packet for an S-LSP,
(E) TCP ACK packets.

The second basic component of the overall architecture is the feedback mechanism and rate control to be used for traffic engineering. MPLS technology does not currently have a standards-based feedback mechanism, but we proposed in [29] that a protocol very similar to that of the ABR service category in ATM networks to be used in MPLS networks as well. Explicit rate feedback was shown to be useful for traffic engineering purposes in [29] due to its promptness and well-proven transient properties. However, we relax this requirement in the current paper for ease of potential implementation. In the current architecture, we require that the core LSRs are ECN-capable [32], and upon congestion, they mark the packet as Congestion Experienced (CE), where the marking can be done with an additional EXP bit; see [33] for an initial proposal on ECN support in MPLS which has not advanced. In the architecture discussed in this paper, the ingress LSR of each LSP periodically sends RM packets along with data packets on the same LSP towards the egress LSR. If the forward RM packet belongs to a P-LSP (i.e., marked as (A1)), the LSRs check the percentage queue occupancy of the silver queue of that interface and sets the CE bit (if not already set) if this value exceeds the configuration parameter μ. Similarly, in the case of an S-LSP forward RM packet (i.e., marked as (B1)), the transit LSR compares the percentage queue occupancy of the bronze (silver) queue with μ if SP (FIFO) queuing is employed and sets the CE bit accordingly. This RM packet is then returned back by the egress LSR to the ingress node indicating congestion status of the forward LSP it belongs to. For every LSP, RM packets are sent to the network once in every T_{RM} seconds. When the ingress LSR receives the congestion information about the LSP, it will invoke an AIMD algorithm to compute the ATR (Allowed Transmission Rate) of the corresponding LSP. The AIMD algorithm is given in Table 1; see also [17]. In this algorithm, *RDF* and *RIF* denote the Rate Decrease Factor and Rate Increase Factor, and *MTR* and *PTR* correspond to Minimum Transmission Rate and Peak Transmission Rate, respectively.

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

Table 1. AIMD Algorithm

We now describe the splitting process for the TCP traffic at the edge LSRs. The edge LSRs identify TCP traffic flows and maintain a list for keeping track of each active flow. We note that the traffic carried between the source and destination LSRs is an aggregation of multiple traffic flows generated by multiple users/applications. Two drop-tail queues per egress LSR, namely the P-LSP and S-LSP queues, are maintained at the edge LSRs. Both queues are drained using the ATR information calculated by the AIMD algorithm given in Table 1. At this stage, we decide on which service queue each TCP flow would join. When a packet arrives, which is not associated with an existing flow, a decision on how to forward the packets of this new flow needs to be made. For this purpose, we employ $D_{P\text{-}LSP}$ and $D_{S\text{-}LSP}$, the delay estimates for the P-LSP and S-LSP queues in the edge LSR, respectively. These delay estimates are calculated by means of dividing the corresponding queue occupancy by the drain rate, ATR, of that queue. The notation d_n denotes the exponential weighted moving averaged difference between the delay estimates of these two queues at the epoch of the nth packet arrival which is updated as follows:

$$d_n = \beta(D_{P\text{-}LSP} - D_{S\text{-}LSP}) + (1 - \beta)d_{n-1}$$

where β is the averaging parameter to be set by the network operator. We refer the reader to earlier work described in [13-15] for the use of exponentially averaged delay or other QoS estimates for making routing decisions.

In our traffic engineering architecture, every new active flow is identified with the arrival of the first packet of the new flow, say nth packet arrival. This new flow is assigned to the secondary LSP with probability $p(d_n)$ which is given in Fig. 2 as a function of the estimated delay difference at the arrival of the nth packet. Equivalently, the new flow is said to be assigned to the primary LSP with probability $(1 - p(d_n))$. This LSP assignment curve is similar to the Random Early Discard (RED) curve used for active queue management [34]. We call this policy for multi-path traffic engineering as the Random Early Reroute (RER) policy; see also [29]. RED has the goal of controlling the average queue occupancy whereas in multi-path TE, the average (smoothed) delay difference between the two queues is controlled by the RER. The RER uses a proportional control rather than a simple threshold policy in order to control the potential fluctuations in the controlled system. The RER favors the min-hop path and resorts probabilistically to the secondary path when the P-LSP queue builds up.

If the delay estimates of the P-LSP or the S-LSP queues exceed a pre-determined threshold, the packets destined to these queues are dropped. Once an LSP is selected upon the arrival of the first packet of a new flow, all successive packets of the same flow will be forwarded over the same LSP. Finally, the edge LSRs are assumed to have the same per-class queuing functionality at their outgoing physical interfaces as core LSRs.

The proposed architecture is summarized in an example network with three edge LSRs (0-2) and three core LSRs depicted in Fig. 3. In this figure, the internals of only the edge LSR 0 are given. The P-LSP(n) queue, $n=1,2,$ refers to the queue maintained for TCP data packets destined for the egress LSR n and using the primary path. These packets then join the silver queue of the per-class queuing stage for later transmission towards the core LSR. The S-LSP(n) queue, $n=1,2,$ is similarly defined for packets to be routed over the S-LSP. If SP is used, TCP data packets using the secondary paths will join the bronze queue in the second stage. In Fig. 3, the SP queuing case is depicted. If FIFO queuing were employed, TCP data packets routed over the S-LSP would also join the silver queue as those packets routed over the P-LSP. All queues in the per-destination queuing stage are drained by the ATR of the corresponding queue, which is calculated by the AIMD-based algorithm. RM packets and TCP ACK packets bypass the first stage, and they directly join the gold queue of the second stage.

3. Simulation Results

The performance of the AIMD-based multi-path TE algorithm for TCP traffic is evaluated for the three-node network shown in Fig. 3. We assume that the core LSRs are connected by links each with a capacity of 50 Mbit/s, and each link has a propagation delay of 10 ms. We also assume that edge LSRs are connected to the core LSRs with 1 Gbit/s links and therefore the core-to-core links are the potential bottleneck links.

In our simulations, flow arrivals occur according to a Poisson process, and flow sizes have a bounded Pareto distribution [35]. The bounded Pareto distribution is used as opposed to the normal Pareto (similar to [36]) because the latter distribution has infinite variance, and thus excessively long simulations are required for convergence. Moreover, the bounded Pareto distribution 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 α is the exponent of the power law. As α is decreased, the tail gets larger, and the ratio of long flows increases. The probability density function for the $BP(k, p, \alpha)$ is given by

$$f(x)=\frac{\alpha \mathrm{k}^{\alpha}}{1-(\mathrm{k/p})^{\alpha}}x^{-\alpha-1}, k \le x \le p, 0 \le \alpha \le 2 .$$

The average flow size, m, for the $BP(k, p, \alpha)$ distribution is given by [35]

$$m=\frac{\alpha}{(1-\alpha)(p^{\alpha}-k^{\alpha})}(pk^{\alpha}-kp^{\alpha}) .$$

The following parameters are used for the bounded Pareto distribution in this study: k = 4KBytes, p = 50MBytes, and α = 1.06 or 1.20. The mean flow sizes can then be calculated as m = 30,544 Bytes for α = 1.06, and m = 20,362 Bytes for α = 1.20.

Consider the network in Fig. 3. We fix the average outgoing traffic from each edge LSR to 70 Mbps in our simulations. The offered traffic from edge LSR i to edge LSR j is denoted by $T_{i,j}$. For simplicity, we assume that that $T_{i,((i+1)\ mod\ 3)} = \gamma T_{i,((i-1)\ mod\ 3)}$ for $0 \le i \le 2$. The traffic spread parameter, γ, $0 \le \gamma \le 1$, is introduced in order to characterize the effect of the traffic distribution on multi-path TE. We note that the case of γ = 1 (γ = 0) corresponds to fully symmetrical (asymmetrical) traffic. In the case of γ = 1, we have 35 Mbps average outgoing traffic in each direction, whereas all the outgoing traffic takes the counter-clockwise direction in the γ = 0 scenario.

In order to evaluate the performance of the flow-based multi-path TE algorithm, we use single-path routing and packet-based TE algorithms for reference. In packet-based TE, packets are routed over the P-LSP or the S-LSP using the RER mechanism given in Fig. 2, but irrespective of the flow they belong to. The packet-based multi-path TE is known to cause out-of-order packet delivery at the destination, and this may adversely affect the TCP performance [23,24]. We study this packet reordering effect on TCP-level goodput in our numerical experiments. Single-path routing uses the minimum-hop path with the AIMD-ECN capability turned on. We use the term "shortest-path routing" to refer to this scheme.

Two sets of RER parameters (see Fig. 2) are used in this study:

- min_{th} = max_{th} = 0 ms and p_0 = 1, called the Shortest Delay (SD) policy since the path with the shorter average delay is chosen when this policy is employed,

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

The SD policy forwards each flow (for flow-based TE) or packet (for packet-based TE) simply to the path with the shorter estimated queuing delay at the ingress edge LSR, and thus it does not favor the P-LSP. SD is used in conjunction with the FIFO queuing discipline where there is no preferential treatment between the P-LSP and the S-LSP at core LSRs. For UDP traffic, we had used RER parameters min_{th} = 30 ms, max_{th} = 150 ms and p_0 = 1 in [29]. However, for the TCP traffic, the queue buildup is less severe due to the rate adaptivity of users so we had to reduce the thresholds accordingly. We experimented extensively with different RER parameters but we observed that in the neighborhood of the chosen RER parameter set, the performance of the RER is quite robust. The results of these exploratory numerical studies are not included in this paper in order to make the presentation more concise. The delay averaging parameter is selected as $\beta = 0.3$. Moreover, if the delay estimate of either the P-LSP or the S-LSP queue exceeds 360 ms, the packets destined to these queues are dropped.

The 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 LSRs, 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 19,840 Bytes.

We use the following parameters for the AIMD algorithm (see Table 1): T_{RM} = 0.1 s, RDF = 0.0625, RIF = 0.125, PTR = 50 Mbps, MTR = 0, and μ = 50%. The study of the impact of AIMD parameters on multipath routing performance is left for future research.

This proposed TCP TE architecture is implemented over ns-2 (Network Simulator) version 2.27 [37] and the TCP-Reno is used in our simulations. The simulation runtime is selected as 300 s. In all the simulation results, events concerning flow arrivals only in the period [95 s, 295 s] are reported. The following five algorithms are compared and contrasted in terms of their performance:

• Flow-based multi-path with RER and Strict Priority
• Flow-based multi-path with Shortest Delay and FIFO
• Packet-based multi-path with RER and Strict Priority

• Packet-based multi-path with Shortest Delay and FIFO

• Shortest-path (i.e., Single Path using the min-hop path)

The goodput of a TCP flow i (in Bytes/s), G_i, is defined as the service rate received by flow i during its lifetime or equivalently it is the ratio Δ_i/T_i, where Δ_i is the number of Bytes successfully acknowledged by the TCP receiver within the simulation duration. 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 runtime, Δ_i will be equal to the flow size in Bytes. The average goodputs for TCP flows as a function of the flow size are given in Fig. 4 for the flow size parameter $\alpha = 1.06$. The average goodput for each flow size range is computed by taking the arithmetic mean of all the individual goodputs of the flows having sizes within the given range. We have the following observations:

- The average TCP goodputs generally increase with the sample flow size since larger flows have the advantage of achieving larger TCP congestion windows as opposed to shorter flows whose congestion windows are limited due to the prevailing slow-start mechanism for such flows,
- The proposed flow-based multi-path TE algorithm using the RER policy and SP queuing always attained the highest average goodput for all tested values of the traffic spread parameter γ and flow size ranges.
- The Shortest Path policy performed very poorly for asymmetrical traffic ($\gamma = 0$) as expected. It is (still) slightly outperformed by the proposed flow-based TE with RER and Strict Priority for the fully symmetrical traffic scenario ($\gamma = 1$). The performance of the Shortest Path algorithm deteriorates compared to flow-based and packet-based TE algorithms as the traffic becomes more asymmetrical and P-LSPs become more congested, i.e., as γ decreases.
- Both packet-based TE algorithms, i.e., Strict Priority/RER and FIFO/Shortest Delay, perform poorly relative to their flow-based counterparts. This is a result of the packet reordering occurring in packet-based algorithms. Large flows that are active for a longer period are more adversely affected with this, whereas packet reordering affects short flows less, since they have relatively small TCP window sizes during their lifetimes due to the slow-start mechanism. The packet-based TE algorithm with Shortest Delay and FIFO is outperformed by the packet-based algorithm with RER and Strict Priority, since the former alternates packets between the P-LSP and S-LSP as d_n fluctuates around zero, causing relatively larger number of out-of-order packet arrivals.

- Each of the three components of the proposed architecture, namely RER, SP queuing, and flow-based splitting, plays a crucial role in the observed TCP performance for the simulated system. Using RER and SP queuing is most effective for larger γ, but for smaller values of γ, it is the flow-based nature of the policy that is most effective (since more packets are routed over the secondary paths). Combining all three components, as in our proposed algorithm, results in a very robust architecture that is effective for all values of γ. Here, we note that γ is representative of the dynamical changes in the traffic demand matrix.
- In Fig. 5, we obtain the same curves as in Fig. 4, but with the flow size parameter $\alpha = 1.20$, slightly reducing the mean flow size. For this particular case, we did not observe much of a difference with respect to the $\alpha = 1.06$ case in the relative performances of the five algorithms we studied.

The average goodputs for the five routing algorithms as a function of the traffic distribution parameter γ are shown in Fig. 6. In this figure, the average goodput is calculated as the arithmetic mean of all flow goodputs. We observe that the flow-based TE algorithm with RER and Strict Priority achieves the highest average goodput among all the TE algorithms considered in this study. The average goodputs for the flow and packet-based TE algorithms with RER and Strict Priority decrease as γ decreases since P-LSP becomes more congested. On the other hand, performances of the flow and packet-based TE algorithms with Shortest Delay and FIFO do not change significantly as γ changes between 0.0 and 1.0. This is a consequence of the equal treatment of the P-LSP and the S-LSP with these algorithms. The TE algorithms with RER and Strict Priority and the shortest path routing algorithm have similar performances for large γ since routing with TE algorithms boils down to the shortest path routing when γ is large as P-LSP becomes lightly loaded.

In order to have a more accurate representation of the goodputs achieved by individual packets, we compute the normalized goodput, which is defined as

$$G_{norm\text{-}avg} = \frac{\sum_i n_i G_i}{\sum_i n_i},$$

where G_i is the goodput attained by flow i, and n_i is the number of packets for flow i. The normalized goodputs for the five routing algorithms as a function of the traffic spread parameter γ

are shown in Fig. 7. Since $G_{norm\text{-}avg}$ emphasizes the effect of larger flows to a greater extent, the difference between the average goodputs achieved by flow and packet-based TE algorithms can be seen more clearly.

The relative change of the normalized goodputs with the four TE algorithms with respect to the shortest path routing are given in Table 2. This relative change, Δ^{TE}, is computed for a generic TE method as

$$\Delta^{TE} = \frac{G_{norm-avg}^{TE} - G_{norm-avg}^{ShortestPath}}{G_{norm-avg}^{ShortestPath}}$$

where $G_{norm-avg}^{ShortestPath}$ is the normalized goodput with the shortest path routing, and $G_{norm-avg}^{TE}$ denotes the normalized goodput with one of the four TE algorithms used for the calculation of the corresponding Δ^{TE}. The flow-based TE algorithm with RER and Strict Priority achieves the highest normalized goodputs compared to the other TE algorithms, in all the scenarios we studied. Although the flow-based TE algorithm with Shortest Delay and FIFO has higher normalized goodput relative to the shortest path algorithm for small values of γ, its performance degrades to worse than the shortest path routing for large values of γ, i.e., with more symmetrical traffic distribution and lightly loaded P-LSP. We also note that the Shortest Path algorithm performs very poorly for small γ, yielding very small normalized goodputs around $\gamma = 0$. This is because the shaping queue at the transmitter for the Shortest Path case becomes overloaded.

γ	Flow-based		Packet-based	
	SP/RER	**FIFO/SD**	**SP/RER**	**FIFO/SD**
0.00	42.55	34.99	19.83	13.21
0.06	5.71	2.18	2.48	0.47
0.13	3.03	0.18	1.05	-0.26
0.21	2.10	-0.05	0.79	-0.43
0.30	1.65	-0.20	0.63	-0.53
0.40	1.36	-0.32	0.51	-0.60
0.67	0.15	-0.67	-0.20	-0.81
1.00	0.02	-0.71	-0.23	-0.83

Table 2. Relative increase/decrease of normalized goodput Δ^{TE} for four TE algorithms with respect to shortest path routing

4. Conclusions

In this paper, we propose a flow-based distributed scheme for traffic engineering TCP flows in MPLS networks. 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 strict priority queuing mechanism not for QoS delivery but for improving routing performance via the mitigation of the knock-on effect that is known to exist for general load balancing schemes. Using Poisson flow arrivals and bounded Pareto distributed flow sizes, we show that the flow-based distributed multi-path traffic engineering based on random early reroute and strict priority queuing consistently outperforms the case of a single path in terms of the TCP goodput, whereas the gain in using this scheme increases when the traffic becomes less uniform. Due to the elimination of the out-of-order packet delivery, flow-based splitting also enhances the TCP performance with respect to packet-based splitting especially for long TCP flows, which are hit hard under packet reordering. For future work, we intend to simulate larger and more realistic topologies with a more diversified set of traffic demand matrices. We also plan to generalize our results to more than two alternative paths and to study more intelligent path selection schemes when a-priori information about the traffic demand matrix is available.

References

[1] 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.

[2] L. Berry, S. Kohler, D. Staehle, and P. Trangia, "Fast heuristics for optimal routing in IP networks," Universitat Wurzburg Institut fur Informatik Research Report Series, Tech. Rep. 262, July 2000.

[3] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings of INFOCOM*, Tel-Aviv, Israel, 2000, pp. 519–528.

[4] Y. Wang, Z. Wang, and L. Zhang, "Internet traffic engineering without full mesh overlaying," in *Proceedings of INFOCOM*, Anchorage, USA, 2001.

[5] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *Proceedings of the ACM SIGCOMM*, Pittsburgh, USA, August 2002.

[6] A. Elwalid, C. Jin, S. Low, and I. Widjaja, “MATE: MPLS Adaptive Traffic Engineering,” in *Proceedings of INFOCOM*, Alaska, USA, 2001, pp. 1300–1309.

[7] D. Yuan, “A bi-criteria approach for robust OSPF routing,” in *Proceedings of IEEE Workshop on IP Operations and Management*, Kansas City, Missouri, USA, 2003, pp. 91–98.

[8] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol label switching architecture,” RFC 2481, January 2001.

[9] M. Kodialam and T. V. Lakshman, “Minimum interference routing with applications to MPLS traffic engineering,” in *Proceedings of INFOCOM*, Tel-Aviv, Israel, March 2000.

[10] S. Plotkin, “Competitive routing of virtual circuits in ATM networks,” *IEEE Jour. Selected Areas in Comm.*, pp. 1128–1136, 1995.

[11] 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 *Proceedings of ACM SIGCOMM*, Maryland, USA, 1992, pp. 53–64.

[12] E. Gelenbe, R. Lent, and Z. Xu, “Towards networks with cognitive packets,” in *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, 2000, pp. 3–12.

[13] E. Gelenbe, R. Lent, and Z. Xu, “Measurement and performance of cognitive packet networks,” *Computer Networks*, vol. 37, pp. 691–701, 2001.

[14] E. Gelenbe, R. Lent, and Z. Xu, “Design and performance of cognitive packet networks,” *Performance Evaluation*, vol. 46, pp. 155–176, 2001.

[15] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, “Cognitive packet networks: QoS and performance,” in *Proceedings of the 10th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Fort Worth, TX, 2002, pp. 3–12.

[16] S. Chen and K. Nahrstedt, “Distributed quality of service routing in ad-hoc networks,” *IEEE Jour. Selected Areas in Comm.*, vol. 17, pp. 1488–1504, 1999.

[17] D. M. Chiu and R. Jain, “Analysis of the increase/decrease algorithms for congestion avoidance in computer networks,” *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, June 1989.

[18] V. Jacobson, “Congestion avoidance and control,” *ACM Computer Communication Review; Proceedings of the Sigcomm ’88 Symposium in Stanford, CA, August, 1988*, vol. 18, no. 4, pp. 314–329, 1988.

[19] J. Wang, S. Patek, H. Wang, and J. Liebeherr, "Traffic engineering with AIMD in MPLS networks," in *Proceedings of 7th IFIP/IEEE International Workshop on Protocols for High-Speed Networks*, Berlin, Germany, 2002, pp. 192–210.

[20] F. Baccelli, E. Gelenbe, and B. Plateau, "An end to end approach to the resequencing problem," *Journal of the ACM*, vol. 31, no. 3, pp. 474–485, 1984.

[21] Z. Cao, Z. Wang, and E. W. Zegura, "Performance of hashing-based schemes for internet load balancing," in *Proceedings of INFOCOM*, Tel Aviv, Israel, 2000, pp. 332–341.

[22] M. Laor and L. Gendel, "The effect of packet reordering in a backbone link on application throughput," *IEEE Network Magazine*, vol. 16, no. 5, pp. 28–36, 2002.

[23] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive routing of long-lived IP flows," in *Proceedings of ACM SIGCOMM*, 1999, pp. 215–226.

[24] Y. Lee and Y. Choi, "An adaptive flow-level load control scheme for multipath forwarding," in *Proceedings of Networking - ICN*, Colmar, France, 2001.

[25] S. Oueslati-Boulahia and E. Oubagha, "An approach to elastic flow routing," in *Proceedings of International Teletraffic Congress*, Edinburgh, UK, June 1999.

[26] S. Oueslati-Boulahia and J. W. Roberts, "Impact of trunk reservation on elastic flow routing," in *Proceedings of Networking 2000*, Paris, France, March 2000.

[27] Y.-D. Lin, N.-B. Hsu, and R.-H. Hwang, "QoS routing granularity in MPLS networks," *IEEE Comm. Mag.*, vol. 46, no. 2, pp. 58–65, 2002.

[28] S. Nelakuditi and Z.-L. Zhang, "A localized adaptive proportioning approach to QoS routing," *IEEE Comm. Mag.*, vol. 46, no. 2, pp. 66–71, 2002.

[29] N. Akar, I. Hokelek, M. Atik, and E. Karasan, "A reordering-free multipath traffic engineering architecture for Diffserv/MPLS networks," in *Proceedings of IEEE Workshop on IP Operations and Management*, Kansas City, Missouri, USA, 2003, pp. 107–113.

[30] S. Nelakuditi, Z. L. Zhang, and R. P. Tsang, "Adaptive proportional routing: A localized QoS routing approach," in *Proceedings of INFOCOM*, Tel Aviv, Israel, 2000.

[31] F. L. Faucheur, L.Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, and P. C. J. Heinanen, "MPLS support of differentiated services," RFC 3270, May 2002.

[32] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168, 2001.

[33] B. D. K. Ramakrishnan and S. Floyd, "A proposal to incorporate ECN in MPLS," Internet Draft, draft-ietf-mpls-ecn-00.txt, 1999.

[34] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[35] I. A. Rai, G. Urvoy-Keller, and E. W. Biersack, “Analysis of LAS scheduling for job size distributions with high variance,” in *Proceedings of ACM Sigmetrics*, CA, USA, 2003, pp. 218–228.

[36] L. Guo and I. Matta, “The war between mice and elephants,” in *Proceedings of ICNP’2001: The 9th IEEE International Conference on Network Protocols*, Riverside, CA, 2001.

[37] S. McCanne and S. Floyd, “ns Network Simulator,” Web page: `http://www.isi.edu/nsnam/ns/`, July 2002.

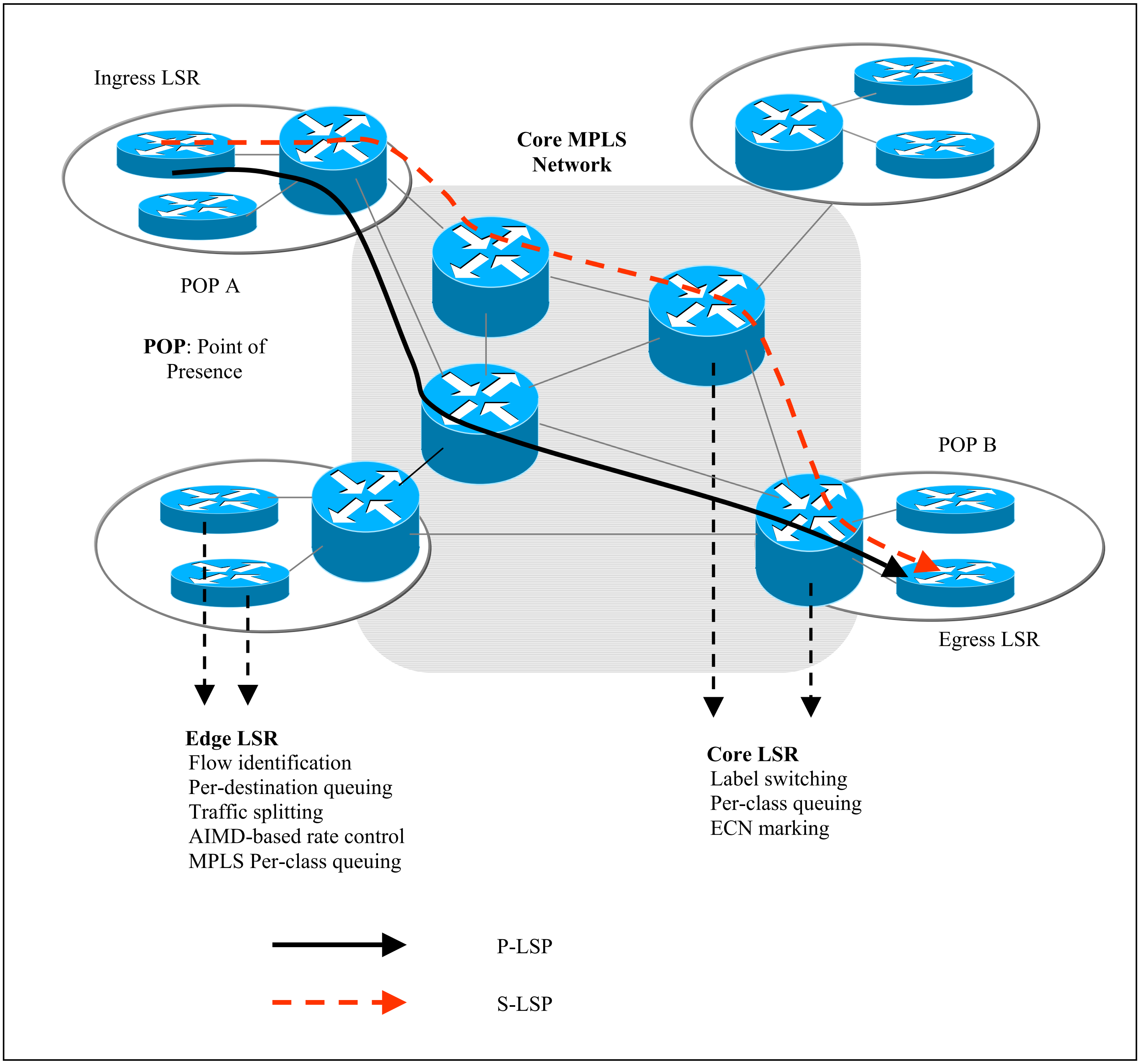

Fig. 1 MPLS network with edge and core LSRs and the minimum hop P-LSP and the S-LSP from the ingress LSR at POP A to the egress LSR at POB B

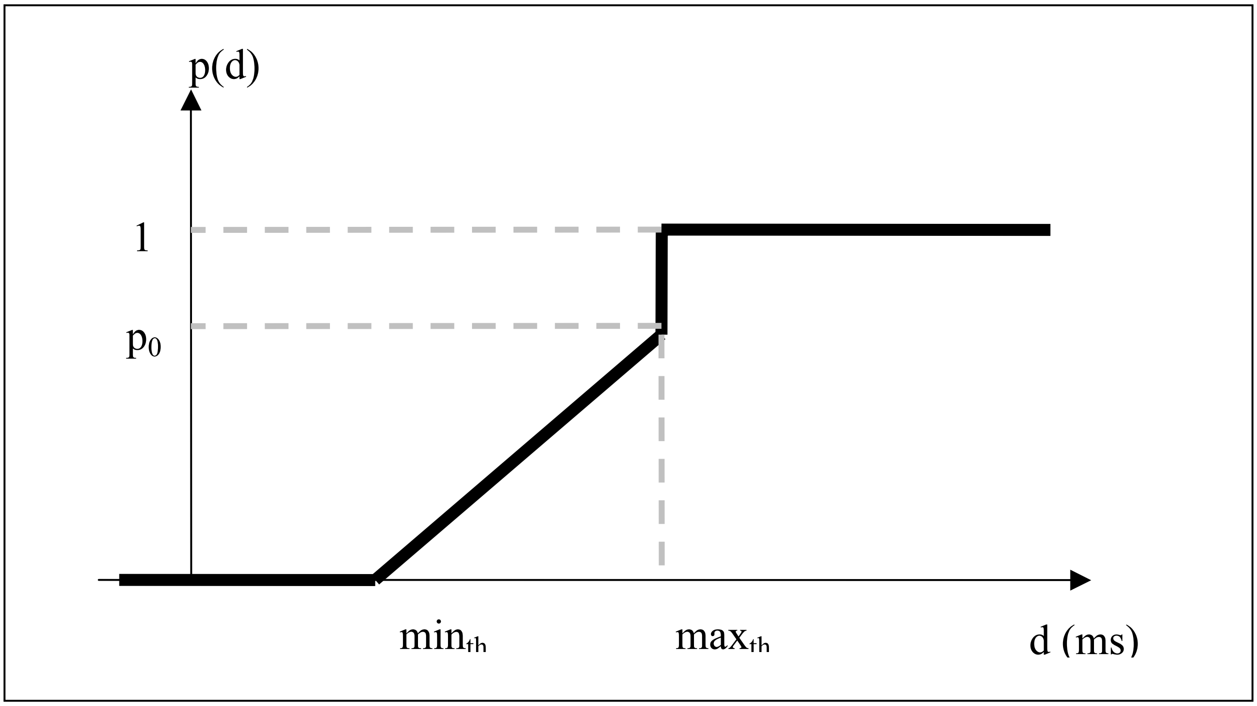

Fig. 2 The traffic splitting function *p(d)*

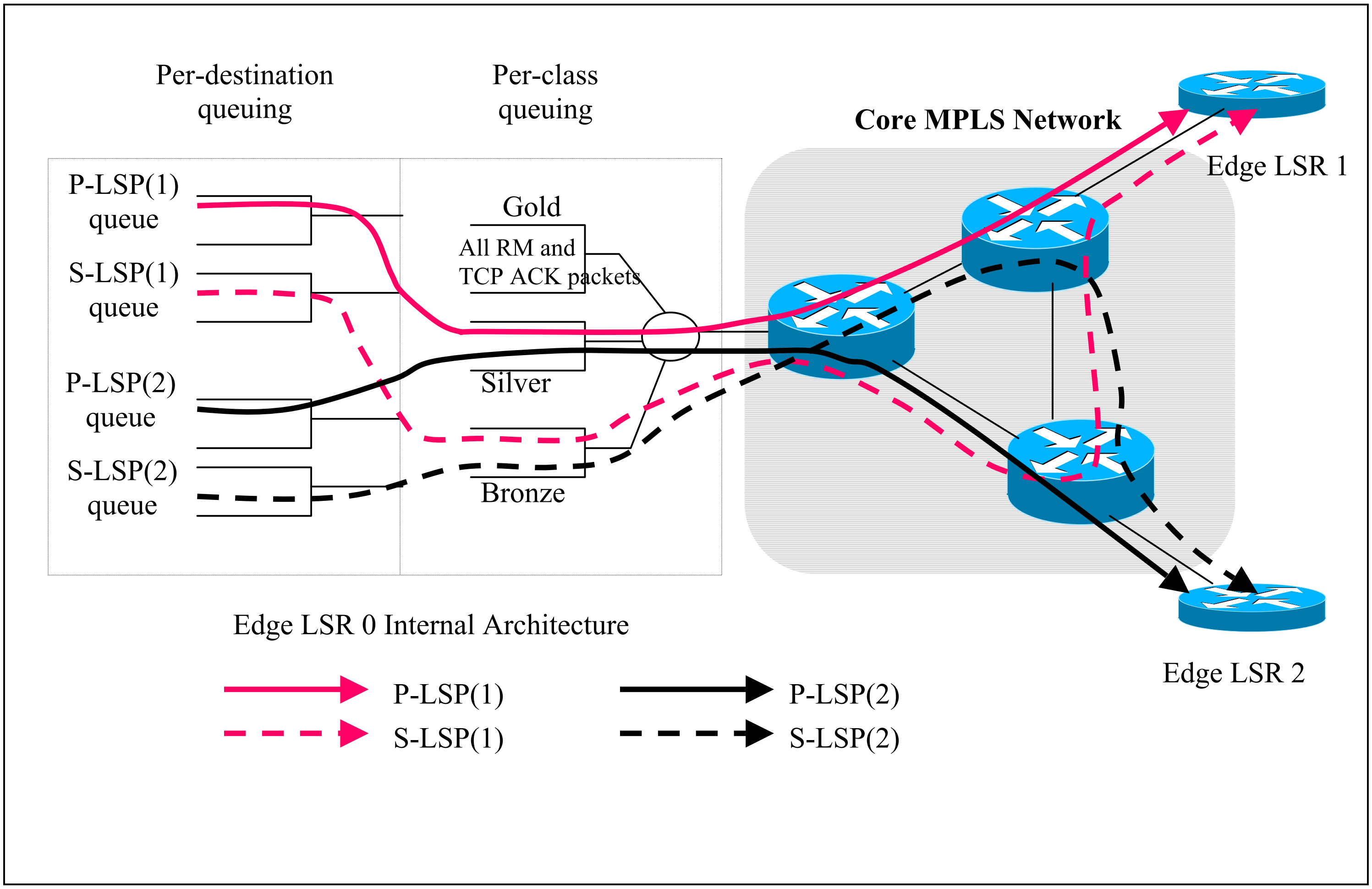

Fig. 3 MPLS network with 3 edge LSRs and 3 core LSRs with strict priority queuing at the physical interfaces. All outgoing routes for edge LSR 0 for TCP data packets are depicted.

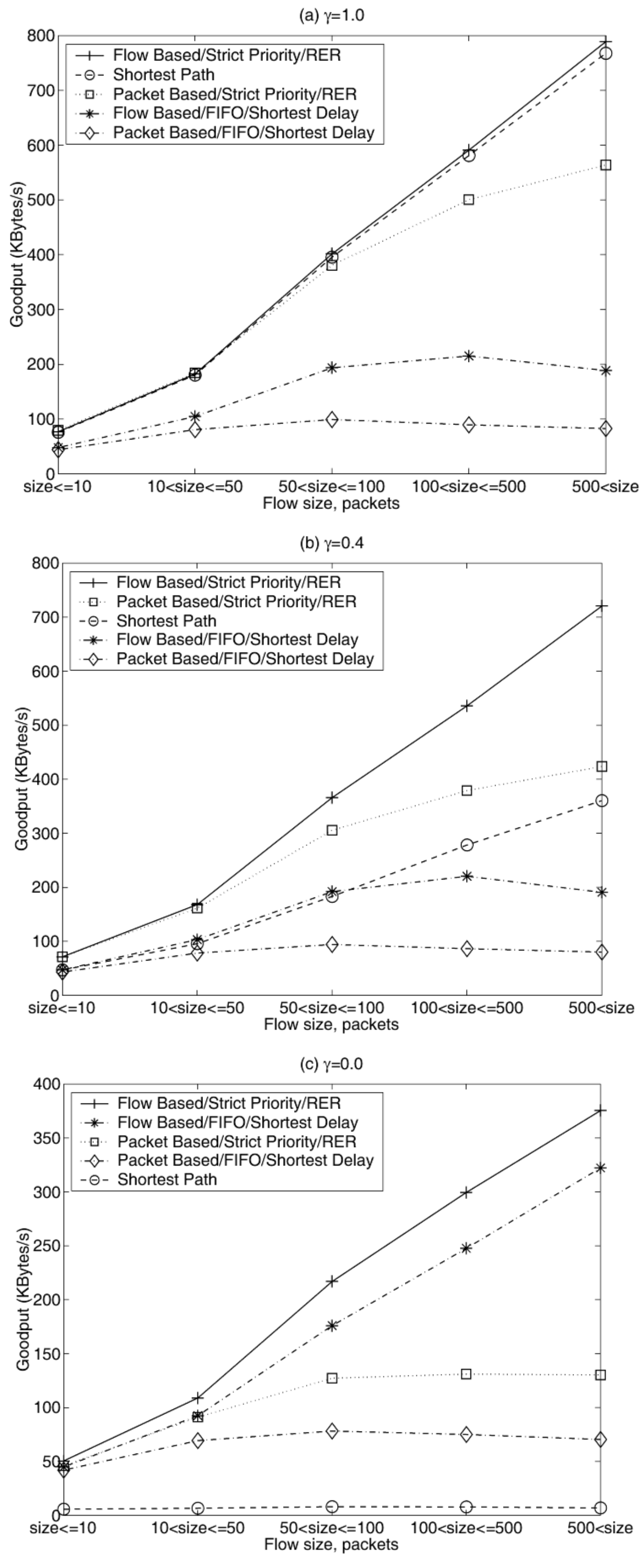

Fig. 4 Goodput as a function of sample flow size for different values of the flow size parameter $\alpha = 1.06$ and (a) $\gamma = 1.0$, (b) $\gamma = 0.4$, and (c) $\gamma = 0.0$.

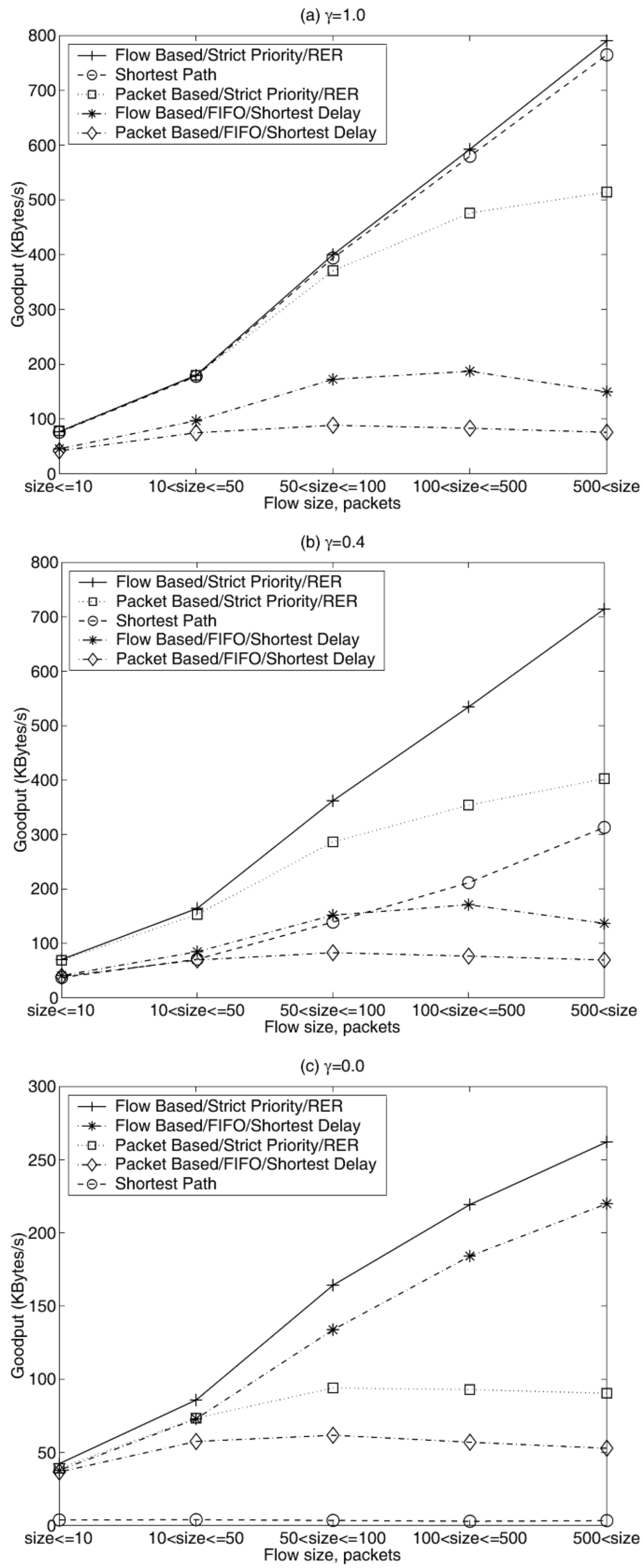

Fig. 5 Goodput as a function of sample flow size for different values of the flow size parameter $\alpha = 1.20$ and (a) $\gamma = 1.0$, (b) $\gamma = 0.4$, and (c) $\gamma = 0.0$.

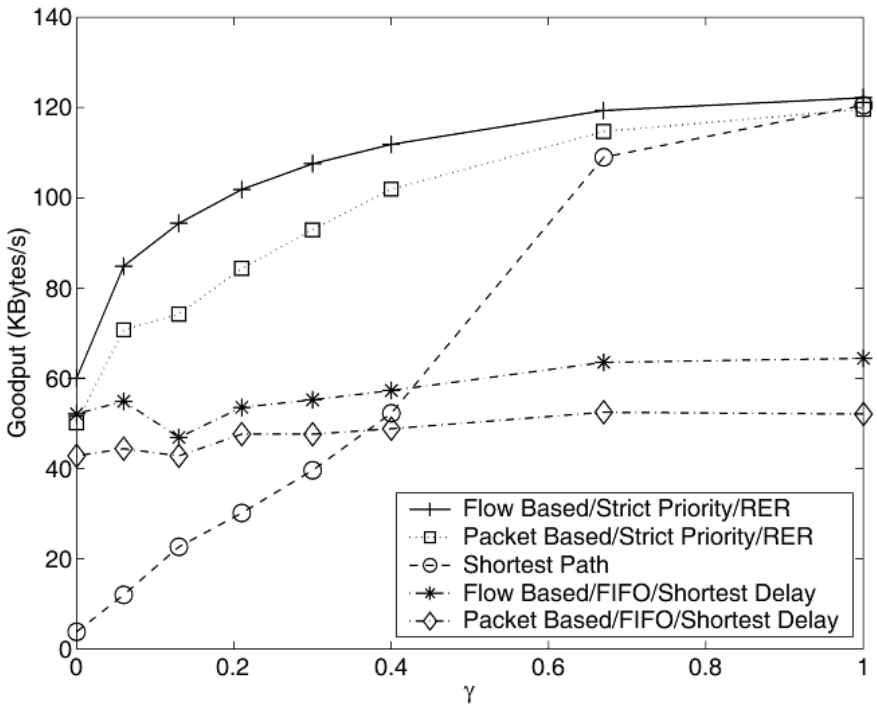

Fig. 6 Average per-flow goodput as a function of γ for $\alpha = 1.20$.

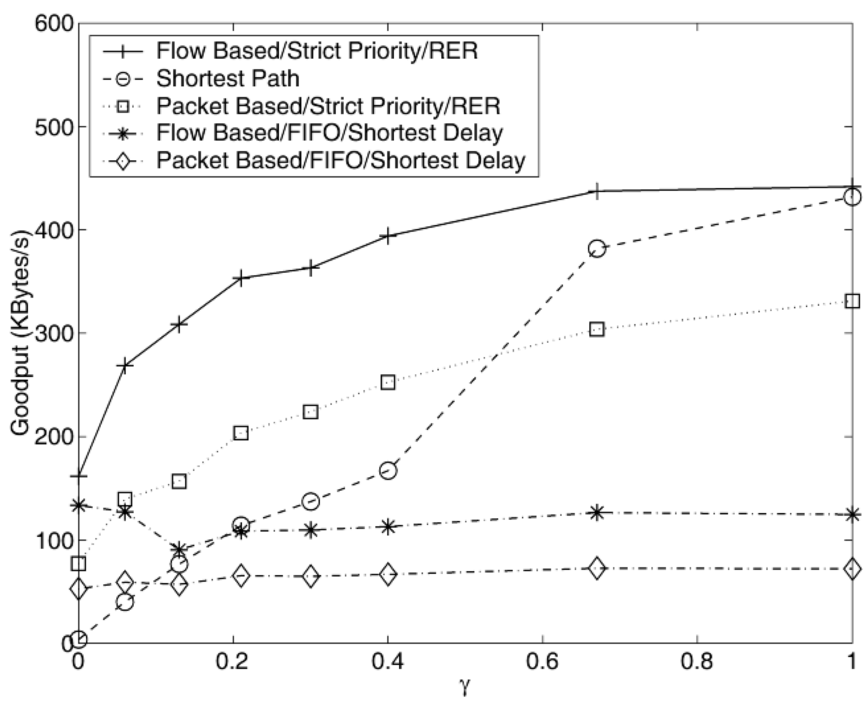

Fig. 7 Normalized goodput as a function of γ for $\alpha = 1.20$.