COMBINED USE OF PRIORITIZED AIMD AND FLOW-BASED TRAFFIC SPLITTING FOR ROBUST TCP LOAD BALANCING

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Onur Alparslan

January 2005

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Ezhan Karaşan(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof Dr. Nail Akar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Erdal Arıkan

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

COMBINED USE OF PRIORITIZED AIMD AND FLOW-BASED TRAFFIC SPLITTING FOR ROBUST TCP LOAD BALANCING

Onur Alparslan
M.S. in Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Ezhan Karaşan
January 2005

In this thesis, we propose a multi-path TCP load balancing traffic engineering methodology in IP networks. In this architecture, TCP traffic is split at the flow level between the primary and secondary paths in order to prevent the adverse effect of packet reordering on TCP performance occuring in packet-based load balancing schemes. Traffic splitting is done by using a random early rerouting algorithm that controls the queuing delay difference between the two alternative paths. We apply strict priority queuing in order to prevent the knock-on effect that arises when primary and secondary path queues have equal priority. Probe packets are used for getting congestion information from the output queues of links along the paths and AIMD (Additive Increase/Multiplicative Decrease) based rate control using this congestion information is applied to the traffic routed over these paths. We compare two queuing architectures, namely first-in-first-out (FIFO) and strict priority. We show through simulations that strict priority queuing has higher performance, it is relatively more robust than FIFO queuing and it eliminates the knock-on effect. We show that avoiding packet reordering by flow level splitting significantly improves the performance of long flows. The capabilities of ns-2 simulator is improved bu using optimizations in order to apply the simulator to relatively large networks. We show that incorporating a-priori knowledge of the traffic demand matrix into the proposed architecture can further improve its performance in terms of load balancing and byte rejection ratio.

Keywords: Traffic engineering, load balancing, multi-path routing, TCP, AIMD rate control.

ÖZET

ÖNCELİKLENMİŞ AIMD VE AKIM TABANLI TRAFİK BÖLÜMÜ KULLANARAK DAYANIKLI TCP YÜK DENGELEMESİ

Onur Alparslan
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Yard. Doç. Dr. Ezhan Karaşan
Ocak 2005

Bu tezde, IP ağları için çokyollu TCP yük dengelemesi tabanlı bir trafik mühendisliği yöntemi önerilmektedir. Bu mimaride, TCP trafiği birincil ve ikincil yollara akım seviyesinde bölünmektedir. Bunun nedeni paket tabanlı yük dengeleme sistemlerinin paket sırası değişikliğine neden olarak TCP performansını düşürmesidir. Trafik bölmesi bir rasgele erken tekrar yönlendirme algoritması tarafından yapılmaktadır. Bu algoritma, iki alternatif yolun kuyruk gecikmesi farkını kontrol etmektedir. Birincil ve ikincil yollardaki kuyrukların eşit öncelikli olması durumunda oluşan zincir etkisini önlemek için kesin öncelikli kuyruklama kullanılmaktadır. Sonda paketleri kullanılarak yollardaki çıkış kuyruklarındaki tıkanıklık bilgisi elde edilmektedir. Bu bilgi kullanılarak AIMD-tabanlı hız kontrolü uygulanmaktadır. Bu çalışmada iki kuyruklama sistemi karşılaştırılmaktadır. Bunlar ilk-giren-ilk-çıkar (FIFO) ve kesin öncelikli kuyruklamalardır. Simülasyonlarla kesin öncelikli kuyruklamanın daha yüksek performansa sahip olduğu, göreceli olarak daha dayanıklı olduğu ve zincir etkisini önlediği gösterilmektedir. Akım tabanlı bölme sayesinde paket sırası değişikliğinin engellenmesi uzun akımların performansını önemli oranda arttırmaktadır. Ayrıca ns-2 simülatörünün çokgen ağ topolojisi simülasyon kapasitesi bu simülasyonları gerçekleştirebilmek için ciddi oranda arttırılmıştır. Trafik istek matriksi hakkında önsel bilginin önerdiğimiz yapıya dahil edilmesi durumunda yük dağılımı ve bayt reddetme oranı bakımından performansın daha da arttırılabileceği gösterilmektedir.

Anahtar kelimeler: Trafik Mühendisliği, Yük Dengelemesi, Çokyollu Yönlendirme, TCP, AIMD Hız Kontrolü

ACKNOWLEDGEMENTS

I gratefully thank my supervisor Asst. Prof. Dr. Ezhan Karaşan and co-supervisor Asst. Prof. Dr. Nail Akar for their supervision and guidance throughout the development of this thesis.

Contents

List of Figures

List of Tables

To My Family. . .

Glossary

ABR	Available Bit Rate, 10
AIMD	Additive Increase/Multiplicative Decrease, 6
AMP	Adaptive Multi-Path, 4
ATR	Allowed Transmission Rate, 18
BRR	Byte Rejection Ratio, 38
CE	Congestion Experienced, 18
CPN	Cognitive Packet Networks, 5
ECMP	Equal Cost Multi-path, 4
ECN	Explicit Congestion Notification, 10
FIFO	First-In-First-Out, 11
IGP	Interior Gateway Potocol, 5
IS-IS	Intermediate System - Intermediate System, 2
ISP	Internet Service Provider, 1
MPLS-OMP	MPLS Optimized Multipath, 5
MPTE	Multipath Traffic Engineering, 3
MTR	Minimum Transmission Rate, 19

OSPF	Open Shortest Path First, 2
OSPF-OMP	OSPF Optimized Multipath, 4
P-RM packet	Resource management packet of primary path, 18
PP	Primary path, 9
PTR	Peak Transmission Rate, 19
RDF	Rate Decrease Factor, 19
RDF	Rate Increase Factor, 19
RED	Random Early Detect, 10
RER	Random Early Reroute, 10
RM packet	Resource Management packet, 16
s-d	source-destination, 9
S-RM packet	Resource management packet of secondary path, 18
SD	Shortest Delay, 28
SMR	Split Multipath Routing, 6
SP	Secondary path, 9
TE	Traffic Engineering, 2

Chapter 1

Introduction

Today, Internet is a very important communications infrastructure. Many companies, governments, academic institutions, people etc. are using Internet for their economic, social, political, cultural, educational activities. The rapid increase in the amount of activities, also bring the rapid increase in the amount of traffic created and carried on the Internet. This rapid increase of traffic can decrease the performance of Internet unless precautions are taken. Therefore ISPs (Internet Service Providers) must cope with rapid traffic increase, higher quality service expectations of their customers and higher service requirements of new applications. There are two main approaches that ISPs make use of:

- Network Planning and Capacity Expansion
- Traffic Engineering

Network Planning and Capacity Extension is a very long-term process that aims to develop the network architecture, design, capacity, and the configuration of network elements to accommodate current expectations and also expanding current network capacity in order to accommodate future traffic expectations that are obtained from traffic forecasts.

Internet Traffic Engineering (TE) is defined as the set of mechanisms for performance evaluation and performance optimization of operational IP networks. In particular, traffic engineering controls how traffic flows through a network so as to optimize resource utilization and network performance. These evaluations and optimizations are carried on measures like delay, delay variation, packet loss, and throughput [1].

1.1 Traffic Engineering

TE mechanisms can be applied to hop-by-hop, explicit, or multi-path routed networks. Traditional hop-by-hop routed IP networks using IS-IS (Intermediate System - Intermediate System) or OSPF (Open Shortest Path First) routing protocols, which are link-state protocols based on the Dijkstra algorithm, make use of simple link weights such as hop-count, delay or bandwidth. Due to their simplicity and fast convergence, hop-by-hop routing algorithms allow IP routing to scale to large networks. However, they do not optimize resource utilization and network performance very well. There are a number of studies on optimizing the resource utilization and network performance in hop-by-hop routed networks by using traffic engineering. By using a given traffic demand matrix information, these studies try to calculate the optimal set of link weights that optimize the resource utilization and network performance [2, 3, 4]. However the success of these methods depends on the accuracy of traffic demand matrix, which can be difficult to achieve [5]. An extension that can handle failures is available in [6] for robust OSPF routing. However these methods can not always find the optimal solution, because there are some cases where there is no possible set of link weights achieving the optimal solution. Also they can cause severe oscillations due to coarse adjustments in link weights that bring instability.

In overlay networks, service providers establish logical connections between the edge nodes of a backbone, and then overlay these logical connections onto the physical topology. The established logical connections can take any feasible path through the network. Using a long-term traffic matrix and constraint-based routing, possible logical connection layouts are calculated and one of them is selected. In case of a big traffic increase in a logical connection, extra bandwidth is requested from the network. If it is possible, problem is solved by accepting this request and increasing the bandwidth of the logical connection. If it is not possible to give extra bandwidth, it is possible to perform path re-optimization by rearranging the logical connections, so that logical connections using the congested physical link can be re-routed to less congested paths. On the other hand, if there is a big traffic decrease in a logical connection, it is possible to deallocate unused bandwidth from this connection so that it can be used by other logical connections in case they need more bandwidth. One problem of the overlay approach is that for large networks, it may bring significant messaging overheads. Also current implementations of most of the routing protocols do not support a very large number of peers that limit the number of logical links adjacent to a node.

Another TE method is Multipath Traffic Engineering (MPTE). The goal of multi-path routing is to increase the resource utilization of the network by intelligently splitting the traffic between a source-destination pair among multiple alternative paths. Multipath traffic engineering can be classified into two groups: Connection-oriented and connectionless. Connectionless techniques are based on improving the shortest-path algorithms or routing metrics in IP networks. Connection-oriented techniques use signalling for path setup, such as MPLS or ATM, based on virtual connections between a source destination pair.

1.1.1 Connectionless Multipath Traffic Engineering

One connectionless MPTE technique is ECMP (Equal Cost Multi-path) extension of OSPF [7]. ECMP evenly divides the traffic among all available shortest paths with equal lowest cost. This allows a good load distribution in some network topologies. Also it has robustness due to the good failure detection capability and efficiency of OSPF. The packets can be divided by using either packet based round robin or according to a hash function applied to the source and desination pair. Hashing based routing solves the packet reordering problem within flows. Also ECMP is integrated into OSPF, so it is readily available in OSPF routers. The main problem of this technique is that it requires multiple paths with equal lowest cost. In a typical network, usually there are a limited number of paths that satisfy this requirement. Also uneven traffic splitting is better in many cases. Another technique based on OSPF and capable of uneven traffic splitting is OSPF Optimized Multipath (OSPF-OMP) [8], which uses a hashing based splitting algorithm based on source and destination address. Routers generally do not know the congestion status of distant links in the network, so they do not know the best traffic splitting ratio. OSPF-OMP solves this problem by using a link-state protocol flooding mechanism for informing all routers in the network about the load level of each link in the network. By using this information, routers can calculate the best traffic splitting ratios in order to decrease the load in congested links and minimize the maximum link utilization in the network. However storing detailed information about all links in the network brings scalability problems. Also there is an increased signalling overhead for informing all routers in the network about the load level of each link. There is a recent proposal called Adaptive Multi-Path routing (AMP) in [9] that restricts the distribution of load information to a local scope, thus simplifying both signaling and load balancing mechanisms.

1.1.2 Connection-oriented Multipath Traffic Engineering

In connection-oriented MPTE, multiple logical connections with disjoint paths are established between edge nodes. These logical connections can be considered as explicitly routed paths that are readily implementable using standard-based layer 2 technologies such as ATM or MPLS or using source routed IP tunnels. These logical connections can be determined by using the long-term traffic matrix. One technique based on MPLS is MPLS Optimized Multipath (MPLS-OMP) [10]. It requires an Interior Gateway Potocol (IGP) such as OSPF or IS-IS to provide link state information. Like OSPF-OMP, it uses a hashing based algorithm based on source and traffic address for uneven traffic splitting. Splitting ratio is adjusted gradually for stability.

In [11], a dynamic multi-path routing scheme for connection oriented homogeneous high speed networks is proposed. In this scheme, the ingress node starts making use of multiple paths as the shortest path becomes congested in order to distribute the load and reduce packet loss in the network. If no alternate path exists it only uses the shortest path, because propagation delay is much larger than queuing and transmission delay in high speed networks. It uses source routing and the routing tables are calculated off-line. In [12, 13, 14, 15], a network architecture called "Cognitive Packet Networks (CPN)", which makes use of adaptive techniques to find routes based on user defined QoS criteria like packet loss or delay, is proposed. In this approach, smart packets explore and learn optimal routes by using reinforcement learning in an adaptive manner and inform the source with acknowledgment packets. Then dumb packets that carry actual payload follow these routes selected by smart packets.

There are also works that adapt multi-path routing methods to wireless networks. For ad-hoc wireless networks, a distributed QoS routing scheme that selects a network path with sufficient resources to satisfy a QoS requirement (delay or bandwidth) in a dynamic multihop mobile environment is proposed in [16].

In [17], a mechanism for adaptive computation of multiple paths with an objective to minimize end-to-end delay is proposed. In a wireless environment that has continuously changing topology and no infrastructure, a routing mechanism that uses multiple paths simultaneously by splitting the packets into smaller blocks and distributing the blocks over available paths based on the failure probabilities of paths is proposed in [18]. An on-demand routing scheme called Split Multipath Routing (SMR) that establishes and utilizes multiple routes that are the shortest delay route and the one that is maximally disjoint with the shortest delay route is proposed in [19]. However, choosing the multiple paths with link-disjoint criteria may not be enough for wireless networks. Route coupling, which occurs when two routes are located physically close enough to interfere with each other during data communication, must be considered. In [20], zone-disjointness of routes to minimize the effect of interference among routes in wireless medium is proposed besides link-disjointness. It proposes using directional antennas instead of omni-directional antennas to help decoupling interfering routes.

Recently, there have been a number of multi-path traffic engineering proposals specifically for MPLS networks that are amenable to distributed online implementation. One of them is [21], which transmits probe packets periodically in order to obtain one-way LSP statistics such as packet delay and packet loss. Based on these statistics, it uses a gradient projection algorithm for load balancing. In this approach, intermediate nodes do not perform traffic engineering or measurements, except for normal packet forwarding. Also it does not impose any particular scheduling, buffer management, or a priori traffic characterization on the nodes. However, it gives equal priority to all paths between an s-d pair, which may be problematic in scenarios in which some paths may have significantly longer hop lengths than their corresponding min-hop paths.

Additive Increase/Multiplicative Decrease (AIMD) feedback algorithms are used generally for flow and congestion control in computer and communication networks [22, 23]. In [24], a multipath-AIMD algorithm, which uses binary feedback information regarding the congestion state of each of the LSPs and assumes that each traffic source has a primary path and may utilize the capacity of other secondary paths, is proposed. It tries to minimize the total volume of traffic sent along secondary paths. However, it assumes that all sources have access to all LSPs, which is unrealistic in many networking contexts.

A critical problem in multi-path routing is the potential de-sequencing (or reordering) of packets of a flow due to sending successive packets of a flow over different paths with different delays. Some resequencing algorithms are analyzed in [25]. Their queuing analysis examine the end-to-end delay encountered in the network. Today, the majority of the traffic in the Internet is based on TCP, so the impact of packet reordering on TCP performance is crucial. TCP has a complex receiver behavior and there are many different TCP versions, so for a network with many TCP flows, it is not possible to apply a queuing analysis similar to [25]. Experiments must be carried out for more reliable results. The experiments in [26] on different TCP versions show that packet reordering, which produces false congestion signals, can cause unnecessary and significant throughput degradation. Therefore it is concluded that packet reordering should be prevented when splitting traffic. In [27], it is shown that when the traffic is split in a static manner (i.e., splitting ratios are fixed over time), hashing based splitting algorithms can give a good performance while preventing packet reordering and providing scalability.

The main problem of static traffic splitting is that it is not able to adapt to wide and rapid fluctuations in traffic from variations in traffic demand and changes in the network configuration. Static traffic splitting requires detection of the problem and manually adjusting the network configuration. However in dynamic traffic splitting algorithms, the problems are detected in a very short time by using the information coming from the network and the splitting ratios are changed adaptively by the algorithm in a short time without requiring manual configuration. Similar to static traffic splitting, it is better to prevent packet reordering when dynamically splitting traffic. Flow based multi-path routing algorithms in [28, 29] detect long-lived and short-lived flows and forward the long-lived flows to the shortest path and the short-lived flows to secondary paths. Such a differentiation between long-lived flows and short-lived ones is done, because it is suggested that short-lived flows have more bursty arrival characteristics than long-lived flows. Bursty behavior is shown to have a bad impact on network performance as it can abruptly increase the queue length at routers, causing packet losses. In [30, 31], flow-based routing of elastic flows by applying admission control for blocking flows under congested network conditions is proposed. They try to maximize the throughput of elastic flows at light loaded conditions and preserve the network efficiency at high loaded conditions. In [30], the Maximum-Utility Path algorithm is proposed where least loaded paths are preferred at low load and shortest paths are preferred at high load. In [31], trunk reservation technique of circuit-switched networks is compared with the Maximum-Utility Path algorithm. Unlike hashing based splitting, these dynamic traffic splitting algorithms have scalability problems, because they require flow aware nodes in order to do flow based splitting.

There are some recent dynamic traffic splitting methods proposed for optical networks. A suite of dynamic multipath traffic splitting strategies, each making use of a different type of information regarding the link congestion status is presented in [32]. It shows that traffic splitting decisions using information about the current state of the OBS network perform significantly better than shortest path routing. Some other load balancing traffic splitting methods are proposed in [33, 34].

In [35], a scalable flow-based multi-path TE approach for best-effort IP/MPLS networks is proposed. It uses max-min fair bandwidth sharing with an explicit rate control mechanism. Only the edge nodes of the MPLS network are flow aware, so it is scalable unlike other flow based dynamic traffic splitting algorithms. Its flow-based splitting solves the reordering problem. It is compared with single path routing and packet based multipath routing with streaming UDP flows and it is shown that it has much lower packet loss rates than single path routing and very close packet loss rates to packet based multipath routing. It can be used in networks where the rate of flow arrivals is large enough for performing traffic engineering via flow-based splitting. However, in this paper only UDP flows are considered, so the impact of packet reordering on TCP flow goodputs is not studied. Lower packet loss rate of packet-based splitting does not guarantee higher TCP goodput than flow-based splitting, because TCP receiver behavior is very complex and it also depends on other factors like packet reordering.

1.2 Proposed Traffic Engineering Framework and Contributions

In this thesis, the work in [35] is extended using elastic traffic (TCP flows) instead of UDP traffic, applying AIMD-based rate control instead of the explicit rate flow control and utilizing more realistic models for the Internet traffic. This TCP TE architecture is implemented over ns-2 (Network Simulator) version 2.27 [36]. During the implementation of this TE architecture, many improvements are introduced to ns-2 architecture. These optimizations made it possible to simulate a mesh network with much lower memory requirements.

In the proposed architecture, two link disjoint paths, one being the primary path (PP) and the latter being the secondary path (SP), are established between edge nodes, which have traffic between. Link disjointness is required because in case a congestion occurs on a link shared by the PP and SP of a source-destination (s-d) pair, it would effect the traffic routed over the both paths between this s-d pair and multipath routing would not help. For an s-d pair, PP is chosen as the shortest path found using Dijkstra's algorithm. SP is computed after pruning the links used by PP and using Dijkstra's algorithm in the remaining network graph. The traffic between these two edge nodes are split between PP and SP for load balancing. The splitting algorithm gives the decisions by using the information coming from the network and the local queue lengths. The information coming from the network is carried by probe packets that are periodically sent to the destination nodes by each edge node and sent back to the edge node by the destination nodes. In [35], the information carried in the probe packets is based on the explicit rate feedback mechanism that is motivated by the ABR (Available Bit Rate) service category used for flow control in ATM networks. However in this thesis, a binary feedback mechanism is used instead of the explicit rate feedback mechanism, because it is much simpler to implement. Also it can be implemented with little additional complexity with the help of standards-based ECN (Explicit Congestion Notification) of MPLS, in case it is used over MPLS. Edge nodes maintain two drop-tail queues, one for the PP and one for the SP. The drain rate of these queues change with an AIMD algorithm by using the congestion information provided by the binary feedback mechanism.

The splitting algorithm detects the individual flows and and perform flow-based splitting by probabilistically assigning each flow to one of the two paths based on the moving average difference between the delays of the corresponding queues. We propose the Random Early Reroute (RER) algorithm for traffic splitting, which is inspired by the Random Early Detect (RED) algorithm used for active queue management in the Internet. Flow based splitting is used instead of packet-based load balancing in order to prevent packet reordering within a flow.

This TE architecture is adaptive to the changes in traffic, so it does not require the availability of any prior information on the traffic matrix. However, we also show that its efficiency can be further improved by selecting PP and

SP optimized for the expected traffic load in case an estimated traffic matrix is provided.

When using multiple paths, queuing method used in the architecture has a big impact on its performance. It is well-known that giving equal priority to PPs and SPs may decrease the performance of PPs since SPs typically use longer paths (more hops) than PPs, i.e. they use more resources, and an SP may share links with PPs of other node pairs. Traffic increase on an SP may force sources of PPs sharing links with this SP to move traffic to their own SPs. This further decreases performance, because SPs typically use longer routes and this in turn forces other PPs to move traffic to their SPs. Therefore, this can move the network to an operating point where the performance is even lower than the single path routing. This fact is called the knock-on effect in literature, and precautions should be taken to minimize this effect [37]. For example in [37], the impact of knock-on effect is limited by preferring min-hop paths and discriminating against alternative paths. In [31], when the network is congested, Trunk Reservation is used to prevent the use of long paths in order to deal with the knock-on effect. In [35], a queuing architecture in the MPLS data plane is proposed that assigns Strict Priority to packets of PPs over those of SPs in order to deal with the knock-on effect. In this thesis, we compare the performances of the Strict Priority mechanism with the First-In-First-Out (FIFO) queuing discipline in dealing with the knock-on effect. We show that the Strict Priority queuing proposed in this thesis is more effective and relatively more robust with respect to the changes in the traffic demand matrix than FIFO queuing.

The rest of the thesis is organized as follows. In Chapter 2, we present our TE framework. Our numerical results are presented in Chapter 3 and conclusions and future work are provided in the final chapter.

Chapter 2

Traffic Engineering Framework

In this study, we envision an IP backbone network which consists of edge and core nodes (i.e., routers) and which is capable of establishing explicitly routed paths. In this network, edge (ingress or egress) nodes are gateways that originate/terminate explicitly routed paths. Core nodes carry only transit traffic. Edge nodes are responsible for per-egress and per-class based queuing, flow classification, traffic splitting, and rate control. Core nodes are responsible for per-class queuing and Explicit Congestion Notification (ECN) marking. In this architecture, only the edge nodes are flow aware, so the overall architecture scale better than some other flow-based architectures.

The proposed architecture is composed of four components:

- path establishment
- queuing in edge and core nodes
- feedback mechanism and rate control
- traffic splitting at the edge nodes

2.1 Path Establishment

We assume that edge nodes are single-homed, i.e., they have a link to a single core node. We set up one PP and one SP, which are link disjoint in the core network, from an ingress node to every other egress node for which there is direct TCP/IP traffic. Link disjointness is required because in case a congestion occurs on a link shared by the PP and SP of a source-destination (s-d) pair, it would affect traffic between this s-d pair independent of the path used for a particular flow, and multipath routing will not provide any performance enhancement.

2.1.1 Path Selection with no Traffic Knowledge

For an s-d pair, PP is chosen as the shortest path found using Dijkstra's algorithm. If there are multiple min-hop paths, the one with the minimum propagation delay is chosen as the PP. SP is selected as the shortest path obtained after pruning the links used by PP. If there are multiple min-hop paths, the one with the minimum propagation delay is chosen as the SP. In case the connectivity is lost after pruning the links from the graph, the SP is not established. As an example, PP and SP are shown in Figure 2.1 where PP is using the shortest path and SP is using a link-disjoint shortest path. In this framework, a-priori knowledge on traffic demands is not required when establishing the paths. However when an accurate estimate of the traffic demand matrix is known a-priori, more sophisticated algorithms might be used to select the routes. Next, we discuss how PP and SP can be determined when traffic estimates are available.

2.1.2 Path Selection with Traffic Knowledge

In this section, we optimize the selection of PPs and SPs based on the estimated traffic matrix applied to the network. Instead of using shortest path algorithm,

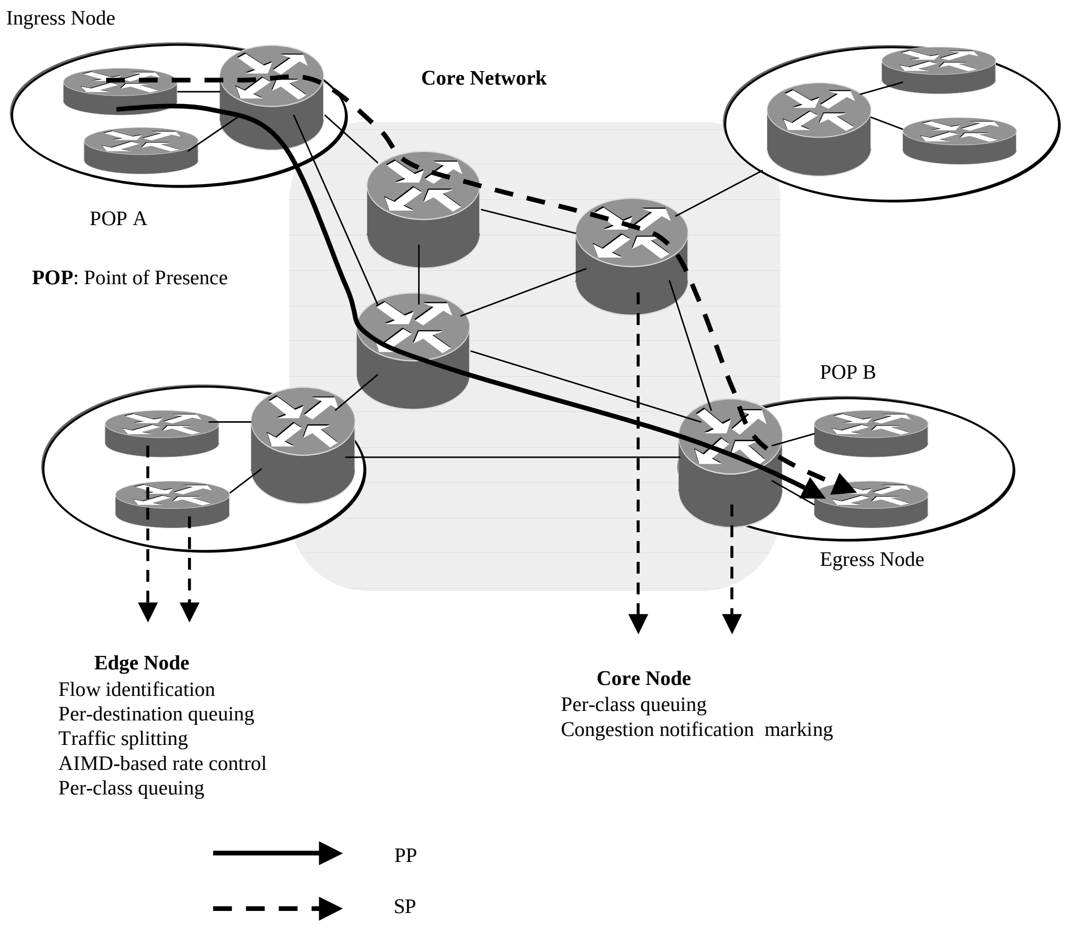

Figure 2.1: Example Architecture

we apply a lexicographic optimization by using the estimated traffic matrix information. Using shortest path can cause some of the links to be heavily congested as it does not consider the traffic distribution. However lexicographic optimization tries to balance the load in the network. It chooses the maximum loaded link in the network and first tries to reduce its load as much as possible. Then among all possible solutions that minimize the maximum load it tries to reduce the load of the next highest loaded link in the network, and goes on until all links are considered. The definition of *lexicographically smaller* is given in [38] as follows:

Given an n-dimensional real vector x define by $\Phi(x)$ the n-dimensional vector whose coordinates are those of x arranged in non-increasing order, i.e.,

$$\Phi(x) = (\Phi_1(x), \Phi_2(x), \ldots, \Phi_n(x)) = (x_{i_1}, x_{i_2}, \ldots, x_{i_n})$$

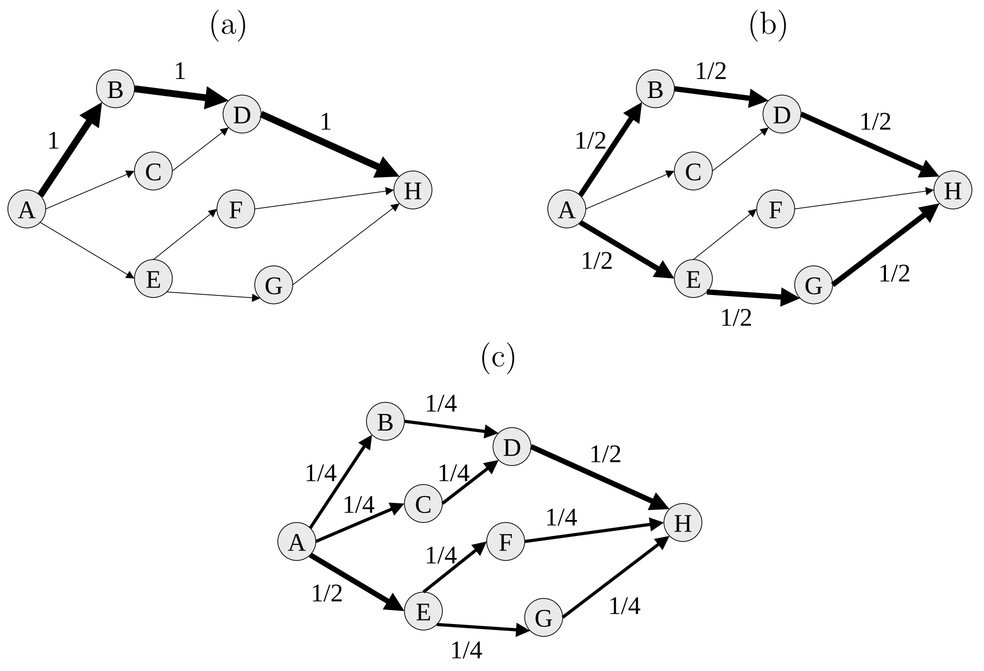

Figure 2.2: Lexicographic optimization: (a) Unbalanced load distribution, (b) After first step in lexicographical optimization, (c) Lexicographically optimal solution

where $x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_n}$. Vector x is called *lexicographically smaller* than or equal to vector y, if either $\Phi(x) = \Phi(y)$, or there exists a number l, $1 \leq l \leq n$ such that $\Phi_i(x) = \Phi_i(y)$, for $1 \leq i \leq l-1$ and $\Phi_l(x) < \Phi_l(y)$. We write $x \preceq y$, and if in addition $\Phi(x) \neq \Phi(y)$, $x \prec y$.

For example, an eight-node topology is given in Figure 2.2a [38]. There is traffic from node A to node H. The capacity of all links are the same and the traffic from A to H is equal to this capacity. The numbers assigned to each link correspond to the traffic load over that link. In Figure 2.2a, only a single path is used, so the load of the links on this path equal to 1 and the load of the other links equal to 0. We can split the the traffic between two paths as seen in Figure 2.2b. Now the maximum link utilization becomes 1/2. It is possible to further distribute the load by using four paths (not link-disjoint) as seen in Fig. 2.2c. Now two links have a load of 1/2 and the other links have a load of 1/4. This is the lexicographically optimal solution, because there is no other distribution that is *lexicographically smaller* than this distribution.

We applied lexicographic optimization to our topology and estimated traffic matrix with two conditions.

- The maximum number of paths for each s-d pair is two, as one PP and one SP.
- PP and SP are link-disjoint.

Lexicographic optimization gives a set of possible solutions. Inside these solutions, we chose the path set where the usage of SPs is the lowest, because the Strict Priority queuing gives higher priority to PPs.

2.2 Queuing in Edge And Core Nodes

In the proposed framework, core nodes 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. Strict priority scheduling is applied where gold queue has strict priority over the silver queue, and the bronze queue. The gold service is given to Resource Management (RM) packets used for gathering binary congestion status from the network and TCP ACK (i.e., acknowledgment) packets. RM packets are allowed to use gold service, because we want to protect RM packets from the possible side effects of a congestion caused by data packets in the network. TCP ACK packets are allowed to use gold service because we want to be able to provide prompt feedback to TCP end users. ACK packets are usually much smaller in size when compared with data packets, so they do not affect the transmission of RM pakets using the same queue as much as the data packets.

For the silver and bronze queues, two queuing models based on the work in [35] are studied. These are strict priority queuing and FIFO (first-in-first-out) queuing. In FIFO queuing, data packets of PPs and SPs join the same silver queue and we do not make use of the bronze queue at all. Therefore, there is no preferential treatment for PP packets that use fewer resources (i.e., traverse fewer hops) over SP packets that typically use more resources. However, it is well-known that giving equal priority to PPs and SPs may degrade the performance of PPs by causing a problem called the knock-on effect[37]. Traffic increase on an SP may force sources of PPs sharing links with this SP to move traffic to their own SPs. This further decreases performance, because SPs typically use longer routes and can in turn force other PPs to move traffic to their SPs. Therefore this can move the network to an operating point that has a performance even worse than the single path routing. In order to mitigate this cascading effect, longer secondary paths should be resorted to only if primary paths can no longer accommodate additional traffic. Based on the work described in [35, 39, 40], we propose to solve this problem by using strict priority queuing where silver service is used for data packets routed over PPs and bronze service is used for data packets routed over SPs. It is possible to implement these queuing models by marking packets using three bits in the packet header. For example, when MPLS is used, packet marking can be implemented by using the standards-based E-LSP (EXP-inferred-PSC LSP) method by using the three-bit experimental (EXP) field in the MPLS header. EXP bits can be used for marking the packet as a

1. Forward RM packet for a P-LSP,
2. Backward RM packet for a P-LSP,
3. Forward RM packet for an S-LSP,
4. Backward RM packet for an S-LSP,

5. TCP data packet for a P-LSP,

6. TCP data packet for an S-LSP,

7. TCP ACK packets.

2.3 Feedback Mechanism and AIMD Rate Control

In our proposed architecture, ingress nodes periodically send RM packets to egress nodes, one over the PP (P-RM) and the other over the SP (S-RM). Egress nodes send them back to the ingress nodes. These RM packets are sent every T_{RM} seconds. The direction of the RM packet must be specified in the packet header, because only the RM packets going towards the ingress node are processed at the core nodes. Also it allows the ingress and engress nodes to find out whether this RM packet is on its forward or backward path. If strict priority queuing is used and when an P-RM packet arrives at the core node on its forward path, the node compares the percentage queue occupancy of its silver queue on its outgoing interface with a threshold level parameter μ and sets the CE (Congestion Experienced) bit (if not already set) of the P-RM packet accordingly. Likewise, if strict priority queuing is used and when an S-RM packet arrives at the core node on its forward path, the node compares the percentage queue occupancy of its bronze queue on its outgoing interface with a threshold level parameter μ and sets the CE (Congestion Experienced) bit (if not already set) of the S-RM packet accordingly.

An ingress node maintains two per-egress queues, one for the PP and the other for the SP. These are drained at the rates determined by the AIMD-based rate control. When the ingress node receives back the RM packet, it invokes the AIMD algorithm in order to calculate the new ATR (Allowed Transmission Rate) value

Table 2.1: The AIMD algorithm

```
if RM packet marked as CE
    ATR := ATR − RDF × ATR
else
    ATR := ATR + RIF × PTR
ATR := min(ATR, PTR)
ATR := max(ATR, MTR)
```

of the path of the RM packet received. The AIMD algorithm is given in Table 2.1. In the AIMD algorithm, RDF and RIF denote the Rate Decrease Factor and Rate Increase Factor, and MTR and PTR denote the Minimum Transmission Rate and Peak Transmission Rate, respectively.

2.4 Traffic Splitting At The Edge Nodes

We propose flow-based splitting, so the edge nodes detect flows and keep a list of active flows. For each egress node, there are two drop-tail queues, namely the PP and SP queues that are maintained at the edge nodes and drained at a rate calculated by the AIMD algorithm given in Table 2.1. As in Figure 2.3, when a packet arrives, which is not associated with an existing flow, a decision on which path to forward the packets of this new flow needs to be made. The delay estimates for the PP and SP queues (denoted by D_{PP} and D_{SP}, respectively) in the edge nodes are used for this purpose. These are calculated by dividing the occupancy of the corresponding queue with the current drain rate ATR. The notation d_n denotes the exponential weighted moving averaged difference between the delay estimates, D_{PP} and D_{SP}, at the epoch of the nth packet arrival which is updated as follows:

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

where β is the smoothing parameter. When the first packet of a new flow arrives at the ingress node, if $d(n) \leq min_{th}$ ($d(n) \geq max_{th}$), then we forward the flow

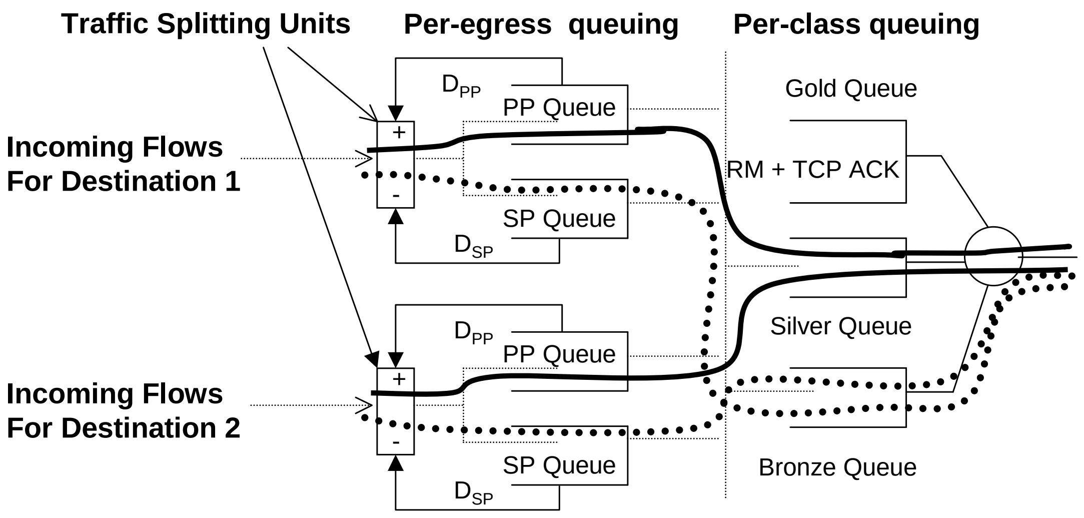

Figure 2.3: Traffic Splitting

over the PP (SP). When $min_{th} \leq d_n \leq max_{th}$, then the new flow is forwarded over the SP with probability $p_0(d_n - min_{th})/(max_{th} - min_{th})$ where min_{th}, max_{th} and p_0 are algorithm parameters to be set. If the delay estimates of the PP or the SP queues exceed a pre-determined threshold, the packets destined to these queues are dropped. The traffic splitting probability is shown in Figure 2.4, which is similar to the Random Early Detect (RED) curve used for active queue management [41]. We call this policy for multi-path traffic engineering as the Random Early Reroute (RER) policy. 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. RER uses a proportional control ($max_{th} > min_{th}$) rather than a simple threshold policy in order to control the potential fluctuations in the controlled system. RER gives priority to the PP (i.e., $min_{th} > 0$), which usually uses less network resources than SP, and resorts probabilistically to the SP when the PP queue builds up. Once a path 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 path.

An example network with three edge nodes (0-2) and three core nodes showing the proposed architecture is given in Figure 2.5. In this figure, the internals of only the edge node 0 are shown. For each egress node, two link-disjoint paths (PP and SP) are created prior to data transmission as described in Section

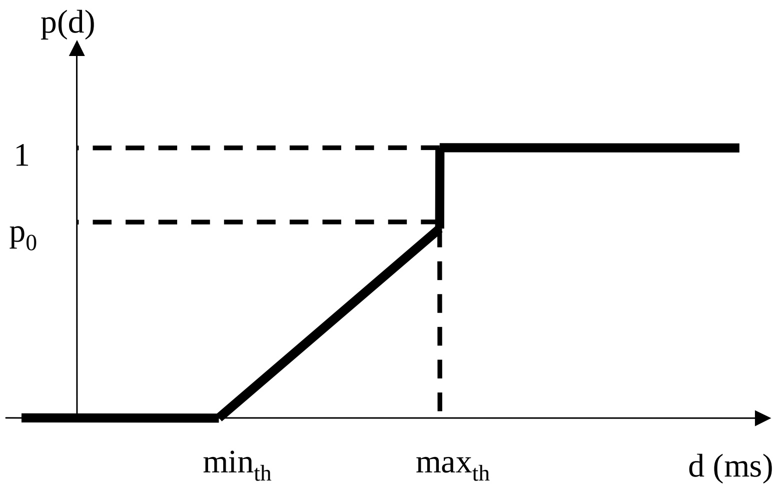

Figure 2.4: Random Early Reroute

2.1. The PP(n) queue, n=1,2, refers to the queue maintained for TCP data packets destined for the egress node 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 node. The SP(n) queue, n=1,2, is similarly defined for packets to be routed over the SP. If strict priority is used, TCP data packets using the secondary paths will join the bronze queue in the second stage. If FIFO queuing were employed instead of the Strict Priority queuing, TCP data packets routed over the SP would also join the silver queue as those packets routed over the PP. 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 directly join the gold queue of the second stage by bypassing the first stage.

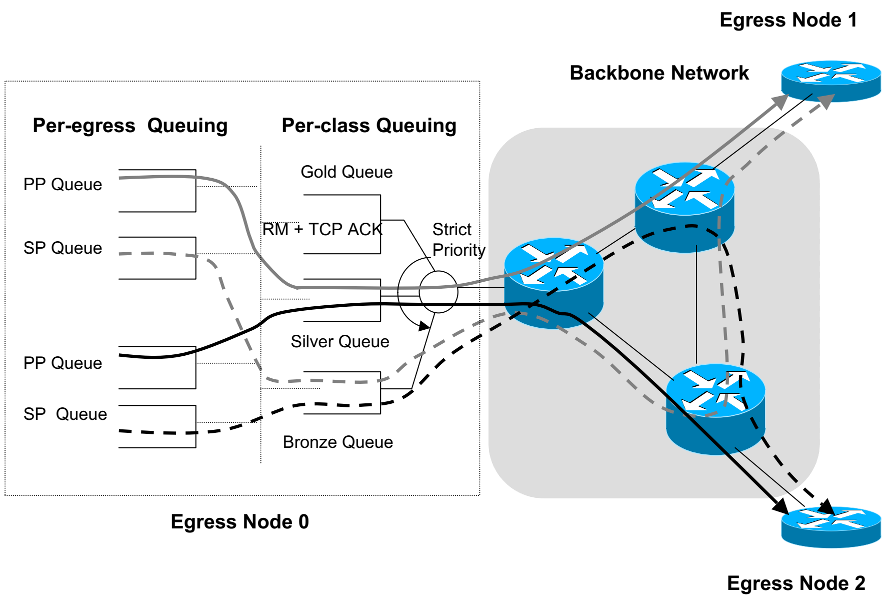

Figure 2.5: Queuing Architecture

Chapter 3

Numerical Results

The proposed TCP TE architecture is implemented over ns-2 (Network Simulator) version 2.27 [36]. During the implementation of this TE architecture, many improvements are introduced to ns-2 architecture. In this chapter, we will first present our simulator architecture. Then, we will present the simulation results to show the performance of our multipath TE architecture. First, the results on a simple three node network will be presented for showing the basic results. In these simulations, the proposed methods are applied over MPLS architecture. Then simulation results on a meshed network will be presented for more realistic results. In these simulations, the proposed methods are generalized and made suitable for applying over any architecture that supports explicit routing. Although it is not necessary to know the traffic matrix for applying the proposed TE architecture, its efficiency can be further improved by selecting PP and SP optimized for traffic load in case a prior traffic matrix is available. The simulation results for both cases are presented for comparision with the mesh network.

3.1 Simulator Architecture

Some new modules required by the new architecture are implemented for ns-2. For the output links of ingress nodes, a new per destination based queuing system, where on the same link many queues drain according to their ATR and adapt to updates in their ATR independent of link speed and other queues, is implemented. For routing of packets, a new source routing module accepting multiple possible paths for flows is implemented. The link agent on these links stores and updates the ATR of queues and delay differences by checking the CE bit of returning probe packets. This agent also decides on whether primary or secondary path will be used upon a flow arrival.

In order to be able to simulate mesh topologies, we introduced a number of optimizations to the ns-2 simulator. The default source routing module in ns-2 does the routing of flows by using tables on source nodes which contain a different route entry for each flow id. This table becomes too large in case of large number of flows. We minimized and made its size independent of number of flows by using a hashing based on source-destination addresses and path numbers.

The input traffic is created offline by calculating the arrival time, size and s-d pair of all flows according to the traffic demands of s-d pairs and chosen distribution of flow sizes. Each run of the simulator accepts this scenario file as input. Therefore, the flow arrival sequence is the same in all simulations. In ns-2, the approach of creating all the flows at the beginning of the simulation brings the problem of high memory requirements. Also the high number of flows used in the simulation brings the problem of simulation speed due to slowness of ns-2 in creating new flows. Therefore, direct simulation of mesh networks for a long duration brings high memory and processing power requirements. We solved these problems by implementing a new architecture that optimizes the usage of existing flows. In our architecture, when a flow finishes sending its data, it informs the simulator. The simulator resets the variables of the flow object, detaches its source and sink from s-d nodes and puts it into a list of unused flows. Upon a new flow arrival information, simulator checks the list of unused flows. If there is a flow available in the list, it takes the flow, attaches its source and sink to the new s-d pair and sets the amount of the data it must transfer according to the offline created input traffic information. The simulator creates a new flow, only if there is no flow left in the list of unused flows. Unless there is an accumulation of flows for an s-d pair, the peak amount of flows required in the simulations becomes fixed independent of the duration of the simulation after the traffic load in network reaches an equilibrium. Also re-using the previously created but finished flows, further improves the speed of the simulation as it solves the problem of the slowness of ns-2 in creating new flows. For example, in some of the simulations given in the next sections, over 1.000.000 flows are applied to the network in each simulation. Our method allowed us to do the simulation by creating only at most 10.000-20.000 flows in most of the simulations independent of simulation duration and number of flows listed in the offline created traffic.

These optimizations make it possible to simulate 5 minutes of an offline traffic injected meshed network with 12 nodes and 19 links by using only around 300 Megabytes of memory that does not increase much with increase in simulation duration and most of which was used by the scheduler for storing the events. Without optimizations, it would require around 5 Gigabytes of memory and this amount increases proportionally to the simulation duration.

Flows do not stop until transferring the amount of data it was given to, and there is no limit on the maximum number of possible flows between a s-d pair, so it is possible to observe the accumulation of flows on a s-d pair in case of a congestion on a link.

Calendar Scheduler [42], which is the default scheduler of ns-2 and the scheduler used in our simulations, is known to have important performance problems in case the time distribution of events in its event list is highly non-uniform. Populating the event list at the beginning of the simulation with the arrival times of all flows which will be applied throughout the simulation, causes such non-uniform distribution as flow arrivals are spread over a long period of time while events created during the simulation are usually spread over a short period of time. In order to solve this problem, in our architecture the list of flows, which will be applied, is divided into small time blocks like 0.1 seconds and stored inside functions responsible for that time block. Each function schedules the execution time of the function carrying the flows of next time block to the beginning time of that block, so the event table of the scheduler is not populated at the beginning of the simulation. This increases the speed by decreasing the initial size of event list and solving possible the performance problems of calendar queues on non-uniform distributions caused by applying offline traffic. Also some enhancements are made to optimize the selection of parameters like bucket width and number of calendar queues for simulation of mesh topologies.

3.2 Three-node Topology Simulations

The performance of our TE algorithm is evaluated first for the three-node topology shown in Figure 2.5. In these simulations, the proposed methods are applied over MPLS architecture. Bandwidth of each link between core nodes is 50 Mbit/s and each has a propagation delay of 10 msec. Also bandwidth of each link between edge nodes and core nodes is 1 Gbit/s. Therefore the potential bottleneck links in the network are the core-to-core links.

In the simulations, flow arrivals occur according to a Poisson process. Flow sizes have a bounded Pareto distribution [43]. The bounded Pareto distribution is used as opposed to the normal Pareto (similar to [44]) because the latter distribution has infinite variance requiring excessively long simulations 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. Therefore, it is much suitable for simulations. The distribution of bounded Pareto 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 increased, the tail gets shorter, and the ratio of long flows decreases. The probability density function for the $BP(k, p, \alpha)$ is given by

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

The average flow size, m, for the $BP(k, p, \alpha)$ distribution is given by [43]

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

The parameters used for bounded Pareto in our simulations are as follows: k = 4KBytes, p = 50MBytes, and $\alpha = 1.20$ or 1.06, corresponding to a mean flow size of m = 20,362 Bytes for $\alpha = 1.20$ and m = 30,544 Bytes for $\alpha = 1.06$.

The average outgoing traffic from each edge node is fixed to 70 Mbit/s in our simulations. The offered traffic from ingress node i to egress node j is denoted by $T_{i,j}$. For simplicity, we assume that $T_{i,((i+1)\ mod\ 3)} = \gamma T_{i,((i-1)\ mod\ 3)}$ for all $0 \leq i \leq 2$. The traffic spread parameter, γ, is introduced in order to characterize the traffic distribution on multi-path TE. $\gamma = 1$ corresponds to fully symmetric traffic and $\gamma = 0$ corresponds to totally asymmetric traffic. In the case of $\gamma = 1$, we have 35 Mbit/s average outgoing traffic in each direction, whereas all the outgoing traffic takes the counter-clockwise direction in the $\gamma = 0$ scenario.

The performance of the flow-based multi-path TE algorithm is compared with single-path routing and packet-based TE algorithms. In packet-based TE, the RER mechanism splits the packets to the PP or the SP, irrespective of the flow they belong to. Therefore it can cause out-of-order packet delivery at the destination, and this may adversely affect the TCP performance [28, 29]. We study this packet reordering effect on TCP-level goodput in our simulations. 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 buffer threshold parameters for the RER curve are used in this study:

- Shortest Delay (SD): $min_{th} = max_{th} = 0$ msec and $p_0 = 1$.
- RER: $min_{th} = 1$ msec, $max_{th} = 15$ msec and $p_0 = 1$.

SD forwards each flow or packet simply to the path with the shorter estimated queuing delay at the ingress node, and thus it does not favor the PP. SD is used in conjunction with the FIFO queuing discipline where there is no preferential treatment between the PP and the SP at core nodes. We experimented extensively with different RER parameters but we observed that in the neighborhood of the chosen RER parameter set, the performance of RER is quite robust. The delay averaging parameter is selected as $\beta = 0.3$. If the delay estimate of either the PP or the SP queue exceeds 360 msec, 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 nodes, including per-destination (primary and secondary) and per-class queues (gold, silver and bronze), have a size of 104,000 Bytes each. The TCP receive buffer is of length 19,840 Bytes.

The following parameters are used for the AIMD algorithm:

- $T_{RM} = 0.1$ s
- $RDF = 0.0625$
- $RIF = 0.125$
- $PTR = 50$ Mbit/s
- $MTR = 0$
- $\mu = 50\%$

TCP-Reno is used in our simulations. The simulation runtime is selected as 300 s. In the calculation of simulation results, only the flows arrive in the period [95 s, 295 s] are used. 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 bit/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 Figure 3.1 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.

Based on the simulation results on this three-node topology, the following observations can be made:

- It is seen that the average goodputs generally increase with the flow size since larger flows have the advantage of achieving larger TCP congestion windows. However the shorter flows cannot reach large TCP congestion windows due to the slow-start mechanism of TCP.
- The RER policy and Strict Priority queuing always gave the highest average goodput for all tested values of the traffic spread parameter γ and all flow size ranges. For asymmetrical traffic ($\gamma = 0$), the Shortest Path policy has a very poor performance. Even for fully symmetrical traffic ($\gamma = 0$), it is slightly outperformed by the proposed flow-based TE with RER and Strict Priority. As the traffic becomes more asymmetric, its performance decreases sharply and gives worse performance than also other TE algorithms tested.
- Due to the packet reordering problem, both packet-based TE algorithms, i.e., Strict Priority/RER and FIFO/Shortest Delay give bad performance when compared with their flow-based counterparts. The negative impact of the packet reordering on TCP performance is more on large flows that are active for a longer period, because they have large window sizes. Its impact on the shorter flows is much less due to their small TCP window sizes during their lifetimes. This effect is much more visible when the packet-based TE algorithm with Shortest Delay and FIFO is compared with the packet-based algorithm with RER and Strict Priority, because the former causes relatively larger number of out-of-order packet arrivals

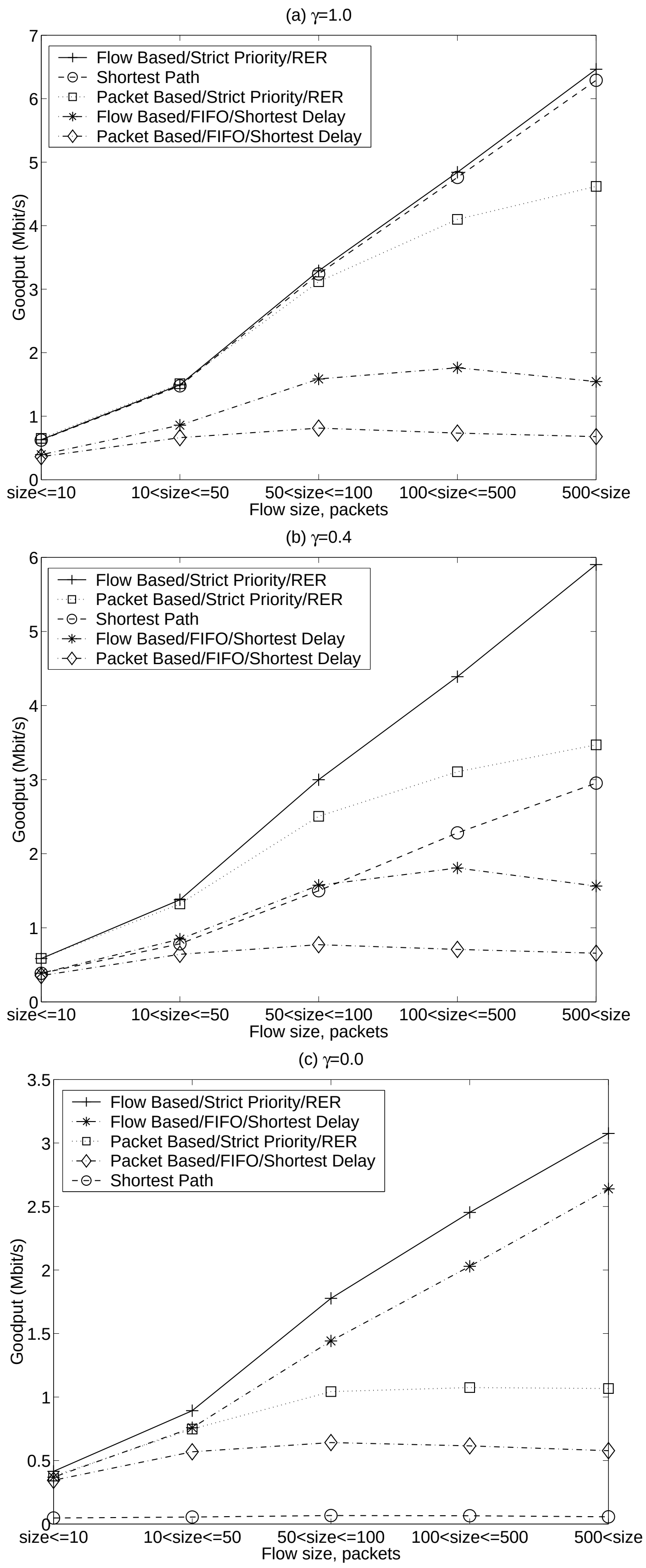

Figure 3.1: Goodput as a function of flow size for $\alpha = 1.06$ and (a) $\gamma = 1.0$, (b) $\gamma = 0.4$, and (c) $\gamma = 0.0$.

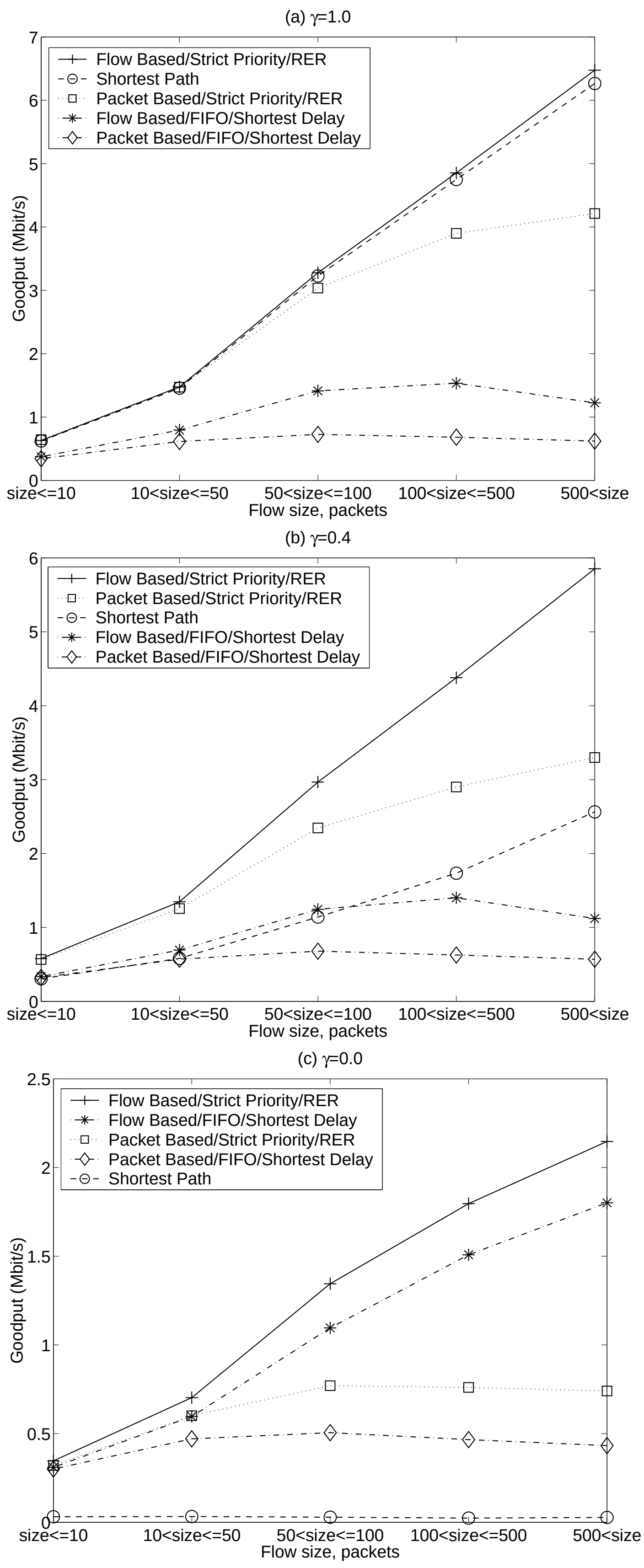

Figure 3.2: Goodput as a function of flow size for $\alpha = 1.20$ and (a) $\gamma = 1.0$, (b) $\gamma = 0.4$, and (c) $\gamma = 0.0$.

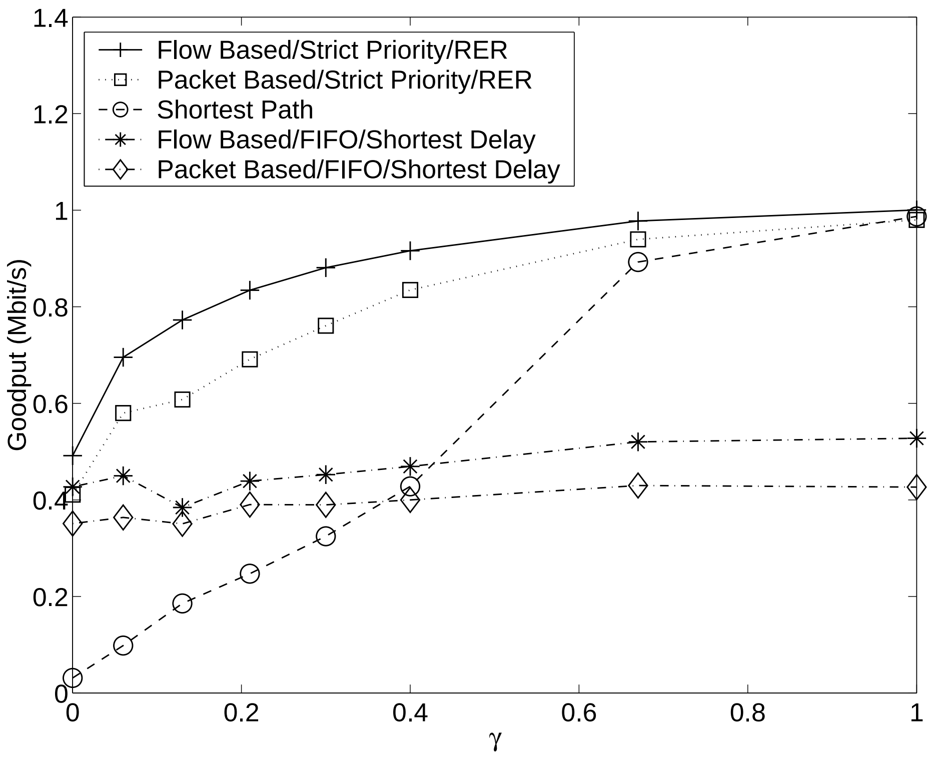

Figure 3.3: Average per-flow goodput as a function of γ for $\alpha = 1.20$.

as it alternates packets between the PP and SP as d_n fluctuates around zero.

- RER, Strict Priority queuing, and flow-based splitting are three important components of the proposed architecture. Joint use of all them makes the architecture more robust and effective, because each of them solves some of the possible problems of the architecture under different conditions.
- When Figures 3.2 and 3.1 are compared, it is seen that there is not a big difference between the results of the relative performances of the five algorithms for flow size parameter $\alpha = 1.20$ and $\alpha = 1.06$.

Figure shows the average goodputs calculated as the arithmetic mean of all flow goodputs for the five routing algorithms as a function of the traffic distribution parameter γ. It is seen that flow-based TE algorithm with RER and Strict Priority gives the highest performance. The performance of the flow and packet-based TE algorithms with RER and Strict Priority decrease as γ decreases

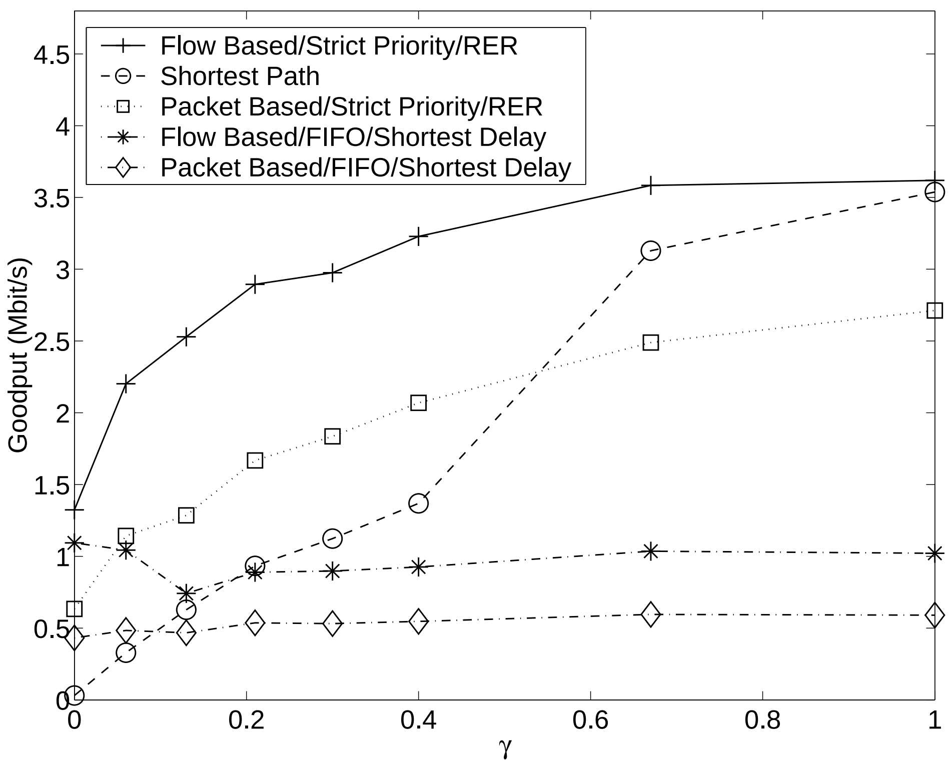

Figure 3.4: Normalized goodput as a function of γ for $\alpha = 1.20$.

because the traffic becomes more asymmetrical and the traffic load on some links increase. We see that the performance of flow-based TE algorithm with RER and Strict Priority and the shortest path routing algorithm are almost the same for very large γ, because flow-based TE algorithm with RER and Strict Priority behaves like the shortest path routing when γ is large as PP becomes lightly loaded. When we look at the performances the flow and packet-based TE algorithms with Shortest Delay and FIFO, we see that they are almost constant as γ changes because of the equal treatment of the PP and the SP with these algorithms.

In order to have a more fair representation of the goodputs achieved by individual flows by also considering the flow lengths, we compute the normalized goodput performance metric which is defined as

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

where G_i is the average goodput of flow i, and n_i is the number of packets successfully delivered by flow i. This metric gives a normalized goodput average weighted by the flow lengths. Figure 3.4 shows the normalized goodputs of all flow goodputs for the five routing algorithms as a function of the traffic distribution parameter γ. It gives more weight to the performance of large flows, so the effect of our TE algorithm is seen more clearly.

Table 3.1: Relative increase/decrease of normalized goodput, Δ^{TE}, for four TE algorithms with respect to shortest path routing.

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

In order to show the performance difference between algorithms 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 3.1. This relative change, Δ^{TE}, is computed for a generic TE method as

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

where $G^{ShortestPath}_{norm-avg}$ is the normalized goodput with the shortest path routing, and $G^{TE}_{norm-avg}$ denotes the normalized goodput with one of the four TE algorithms used for the calculation of the corresponding Δ^{TE}. The highest normalized goodputs is achieved by the flow-based TE algorithm with RER and Strict Priority compared with the other TE algorithms. Although the flow-based TE algorithm with Shortest Delay and FIFO has higher goodput relative to the shortest path algorithm for small values of γ, for large values of γ, i.e., with more symmetric traffic distribution and less congested PP, its performance degrades to worse than the shortest path routing. The packet-based TE algorithms also perform worse than the shortest path routing for large values of γ.

3.3 Mesh Topology Simulations

The performance of our TE algorithm is evaluated for the mesh topology shown in Figure 3.5. This topology and the traffic matrix used in our simulations are taken from [45]. This mesh network is called the hypothetical US topology and has 12 POPs (Point of Presence).

In our simulations, we scaled the speed of 155 Mbit/s links to 45 Mbit/s and the speed of 310 Mbit/s links to 90 Mbit/s for increasing the simulation speed. Also the traffic demands are scaled down accordingly. An edge node is connected to each core node in the topology as there is a traffic demand between all nodes. We assume that edge nodes are connected to the core nodes with 1 Gbit/s links, so they do not create any bottleneck.

First, we will present the simulation results when a prior traffic matrix is not available. However the efficiency our algorithm can be further improved by selecting PPs and SPs optimized for traffic load in case a prior traffic matrix is available. Therefore, we will also present the simulation results when an estimated traffic matrix is available.

3.3.1 Simulations Without Prior Traffic Matrix

Like the three-node topology simulations in previous section, in these simulations we used a traffic model where flow arrivals occur according to a Poisson process and flow sizes have a bounded Pareto distribution. The following parameters

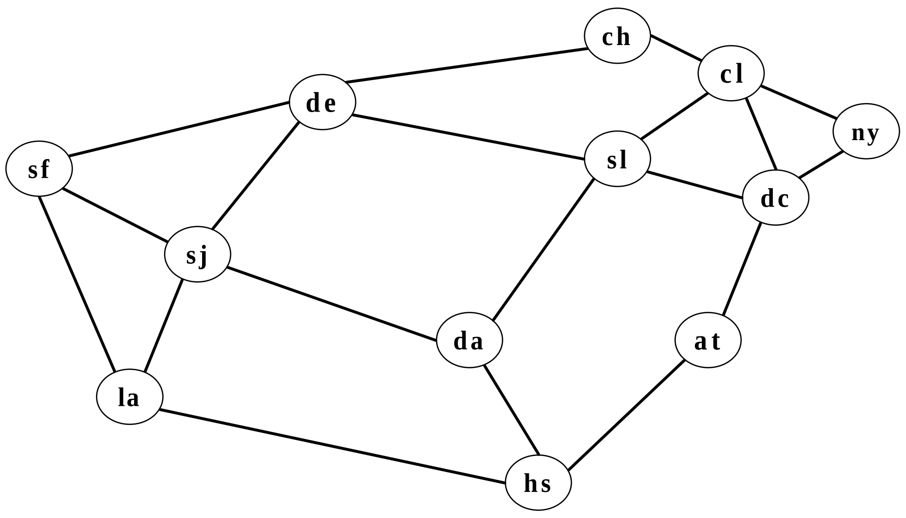

Figure 3.5: Hypothetical US Topology.

are used for the bounded Pareto distribution in this study: $k = 4000$ Bytes, $p = 50 \times 10^6$ Bytes, and $\alpha = 1.20$, corresponding to a mean flow size of $m =$ 20,362 Bytes.

The delay averaging parameter is selected as $\beta = 0.3$. TCP data packets are assumed to be 1040 Bytes long. We assume that the RM packets are 50 Bytes long. All the buffers at the edge and core nodes, including per-destination (primary and secondary) and per-class queues (gold, silver and bronze), have a size of 104,000 Bytes each. The TCP receive buffer is of length 20,000 Bytes.

The following parameters are used for the AIMD algorithm:

- $T_{RM} = 0.02$ s
- $MTR = 1$ bit/s
- $\mu = 20\%$

PTR is chosen as the speed of the slowest link on its path. MTR is chosen as 1 bit/s, in order to eliminate cases causing division by zero in the simulations. If the expected delay of a buffer exceeds 0.36 s, the packets destined to this queue are dropped due to its high delay. The simulation runtime is selected as 300 s. In the calculation of simulation results, only the flows arrive in the period [90 s, 250 s] are used.

Again, we use the $G_{norm-avg}$ as a performance metric. However, we note that some flows are not fully carried due to overloading of certain links in the network. In order to take this effect into account, we introduce a new performance measure, called the net average goodput, denoted by G_{net} (bit/s)

$$G_{net} = \frac{\sum_i \Delta_i G_i}{\sum_i S_i},$$

where Δ_i is the number of bits successfully delivered to the application layer by the TCP receiver for flow i and S_i is the real flow size. If flow i terminates before the end of the simulation, then Δ_i will be equal to the flow size S_i. G_{net} equates the service rate of uncarried packets to zero. In order to show the same effect, we suggest a new measure, called the Byte Rejection Ratio (BRR), which shows the portion of data that cannot be delivered within the simulation duration, in percentage. It is denoted by BRR

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

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

In Figure 3.6a and 3.6b, the effects of AIMD parameters RIF and RDF on $G_{norm-avg}$ are shown. Similarly, in Figure 3.6c and 3.6d the effect of these AIMD parameters on BRR is depicted. In these simulations, RER parameters are chosen as $min_{th} = 1$ msec, $max_{th} = 15$ msec and the strict-priority policy is used. It is seen that the performance of multi-path strict-priority with RER is better in both means than single-path policy. $RDF = 0.0625$ and $RIF = 0.0625$

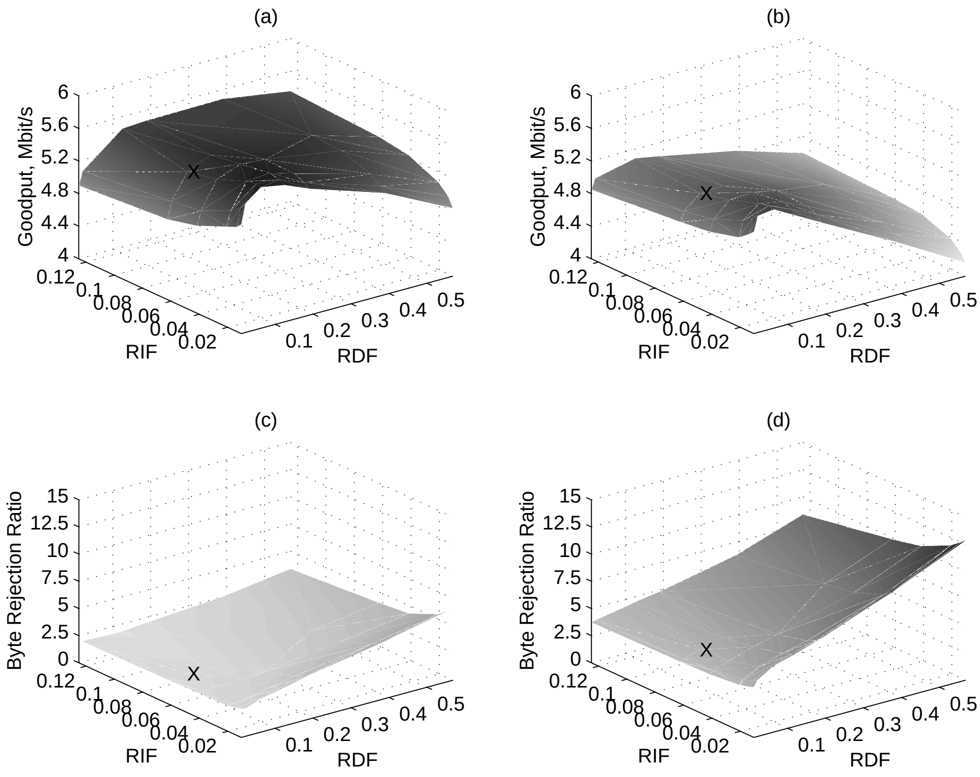

Figure 3.6: As a function of RIF and RDF: (a) G_{net} for the multi-path TE with strict-priority and RER, (b) G_{net} for the shortest-path routing, (c) BRR for the multi-path TE with strict-priority and RER (d) BRR for the shortest-path routing

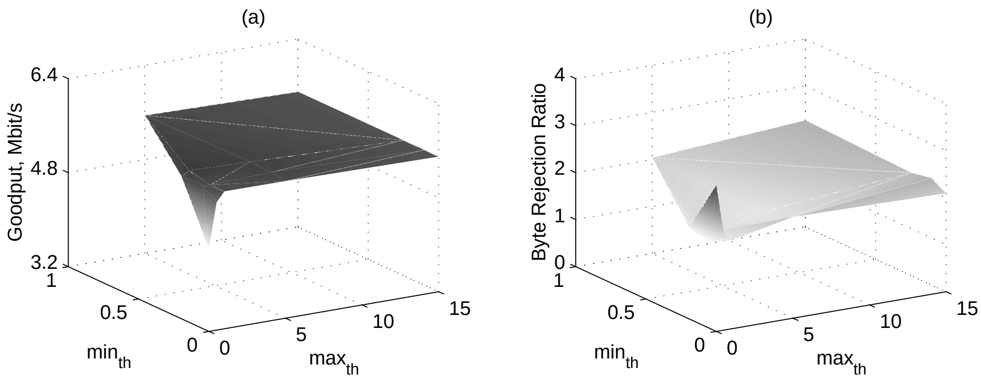

Figure 3.7: As a function of min_{th} and max_{th}: (a) G_{net} for the multi-path TE with strict-priority and RER (b) BRR for the multi-path TE with strict-priority and RER

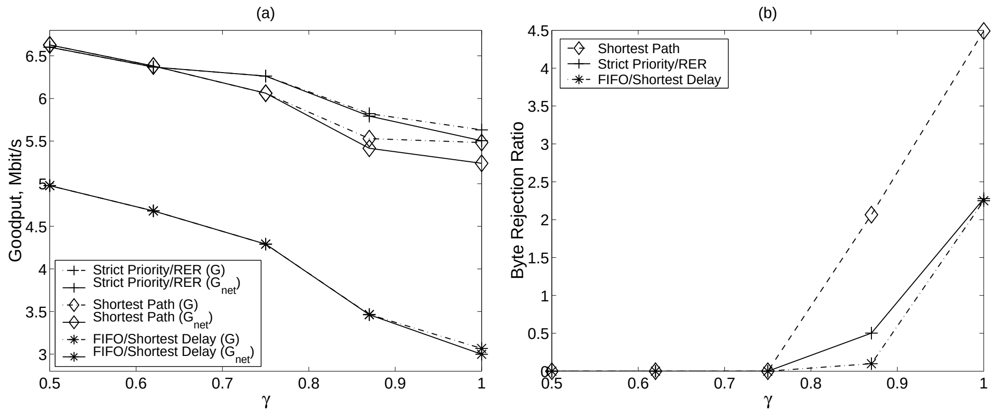

Figure 3.8: As a function of traffic scaling parameter γ: (a) G_{net} and $G_{norm-avg}$ (denoted by G in the figure) (b) Byte Rejection Ratio

point gives good performance in both figures and it is in a robust region, so we use these parameters in the rest of the simulations.

The effects of RER parameters on $G_{norm-avg}$ and BRR are shown in Figures 3.7a and 3.7b, respectively. We observe that, except for the points close to $min_{th} = max_{th} = 0$, which basically corresponds to the SD policy, the performance of the RER is high. When SD policy is used, the performance deteriorates. This performance degration is because of the knock-on effect. As we increase min_{th} and max_{th}, we observe that the performance of RER converges to the single-path routing. In the rest of the simulations, the RER parameters are used as $min_{th} = 1$ msec and $max_{th} = 15$ msec.

In order to show the effect of the total amount of the traffic demand, the traffic is scaled by multiplying the flow sizes with a traffic scaling parameter γ where $0.5 \leq \gamma < 1$, while keeping the flow arrival times same. As seen in Figure 3.8a, at high traffic rates the multi-path TE with strict-priority and RER achieves the highest $G_{norm-avg}$. In fact, there are node pairs, that have the maximum traffic demand in the network, for which the increase in goodput is more than 10 times with the multi-path TE with strict-priority and RER compared to the single-path routing. For these node pairs the PP is heavily congested, and the SP substantially improves the performance. On the other hand, for many node pairs multi-path routing does not improve the goodput since the PP is not congested. The overall performance, represented by $G_{norm-avg}$ which is the average normalized goodput taken over 132 node pairs, still shows a significant improvement for the congested cases.

It is also observed from Figure 3.8a that at high traffic rates the multi-path TE with strict-priority and RER achieves the highest $G_{norm-avg}$. This shows that the multi-path TE with strict-priority and RER not only carries more traffic, but also the carried traffic is transported faster.

In Figure 3.8b, we observe that the BRR of the policy of multi-path routing with strict-priority and RER is approximately half of the BRR for the single-path routing. This indicates a drastic improvement in the performance of congested paths when multi-path routing with strict-priority and RER is used. As the traffic demand decreases, we see that the gap between the multi-path routing with strict-priority and RER and the single-path routing disappears. This is due to the fact that at light traffic loads, PP is not congested, and the multi-path routing effectively behaves as single-path routing. In Figure 3.8b, BRR for the multi-path routing with SD and FIFO is less than the multi-path routing with strict-priority and RER, but the net goodput of the multi-path routing with SD and FIFO queuing is 25-50% lower than the proposed TE approach when γ is changing between 0.5 and 1.0, as shown in Figure 3.8(a).

Since a-priori knowledge of traffic demands is not considered during path set selection, paths are not optimized in terms of minimizing the link utilizations. Consequently, we observe in the simulations that many s-d pairs use multiple bottleneck links on their PPs. Also many of them have SPs that traverse heavily congested links, which limits efficient usage of SPs. In spite of these limitations, our proposed architecture is shown to give better or equal results than the single-path routing policy in both normalized goodputs and BRR.

Simulations With Flow Rerouting

A known problem of using strict priority is the possible starvation of low priority flows. For example, assume that the PP of a s-d pair is heavily congested, but its SP is not congested. In such a case, the traffic splitting algorithm will start forwarding some of the new flows to the SP as expected. However it is not possible to guarantee that the SP will never be congested. In case congestion occurs on a link, the flows that have their PPs traversing this link will have strict priority, so they will not be affected from this congestion unless the total traffic of PP flows exceeds the link capacity. On the other hand, flows that have their SPs traversing this link will be affected by the congestion and the bronze queue on that link will start building up and possibly start dropping SP packets. Consequently, s-d pairs that have their SPs traversing this link, will stop forwarding new flows over their SPs. However, the flows previously assigned to SP will continue using this path irrespective of the status of PP and suffer from this congestion. In case the total traffic of PPs is equal to or more than the link capacity, PP flows use the whole link capacity and the link stops transferring SP packets resulting in starvation of SP flows using that link.

When we check the goodputs of the s-d pairs in the mesh network simulations, we see that this starvation occurs for some s-d pairs such as ny-sf. Therefore we consider a mechanism for avoiding negative effects of starvation. A possible solution is periodically rerouting of existing flows. We store the arrival time of each flow and apply flow rerouting to each flow every T_r seconds. Rerouting must be applied carefully, because choosing a low T_r value may decrease the flow goodput due to frequent packet reordering for that flow. Choosing a high T_r value decreases the number reroutings resulting in prolonged adverse effects of starvation. For limiting the extra complexity due to flow rerouting, we apply flow rerouting to only long flows. This can be achieved by considering the average flow

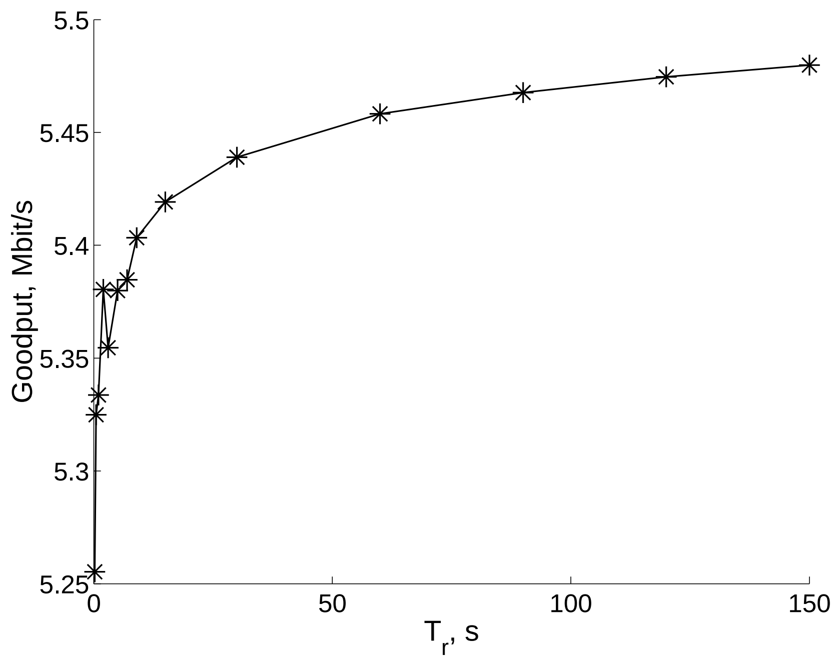

Figure 3.9: $G_{norm-avg}$ as a function of traffic scaling parameter T_r

lengths so that short flows will finish transmitting their data before a rerouting occurs.

The parameters used in the simulations are the same as the parameters in the previous section. As seen in Figure 3.9, applying flow rerouting decreases the overall speed. Overall speed decreases, because when we apply flow rerouting, in case the SP gets worse than PP, we forward SP flows to PP. This increases the load on PPs and queue lengths on PP links and therefore decreases the performance of other PP flows. In other words, we decrease performances of several PP flows in order to increase the performance of some SP flows so that we have a more fair distribution of goodputs among flows. We see a sharp decrease near the point $T_r = 0$ s, because choosing a very low T_r value decreases the flow goodput due to frequent packet reordering in that flow. As seen in Figure 3.10, improvement in BRR is very low, because this path set is not optimized for applied traffic matrix, so its capability of being improved in terms of BRR is very limited. However we will show that there is a big improvement in case a path set optimized with prior traffic matrix is used.

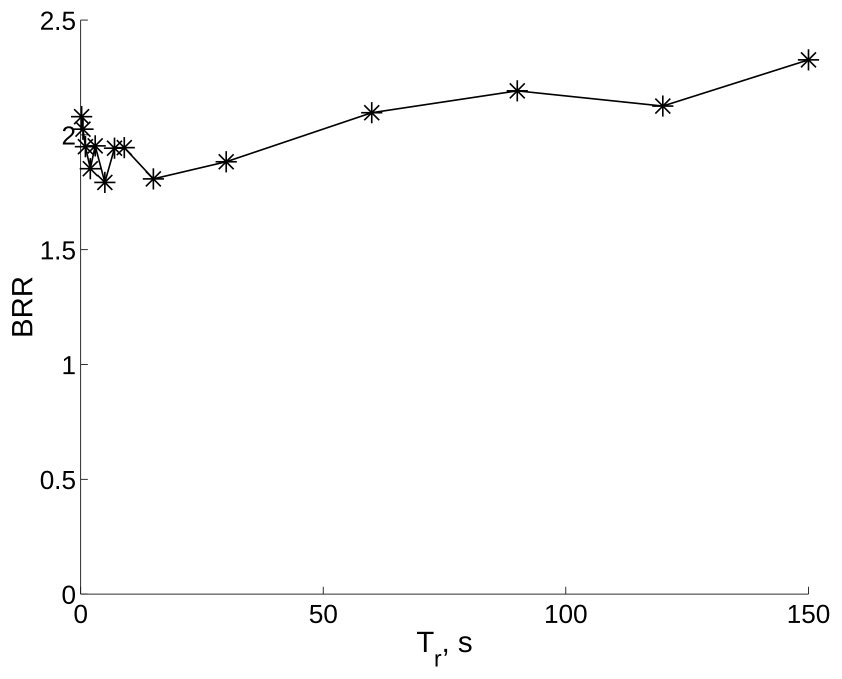

Figure 3.10: BRR as a function of traffic scaling parameter T_r

3.3.2 Simulations for the Case With Estimated Traffic Matrix Available

The parameters used in the simulations are the same as the parameters in the simulations without prior traffic information. All simulations, including the single-path simulations, use the optimized paths. Flow splitting is not used.

In Figure 3.11a and 3.11b, the effects of AIMD parameters RIF and RDF on $G_{norm-avg}$ are shown. Similarly, in Figure 3.11c and 3.11d the effect of the AIMD parameters on BRR is depicted. In these simulations, RER parameters are chosen as $min_{th} = 1$ msec, $max_{th} = 15$ msec and the strict-priority policy is used.

The same observations as in the case of without estimated traffic information can be made. It is seen that the performance of multi-path strict-priority with RER policy is better in both considered metrics than the single-path policy.

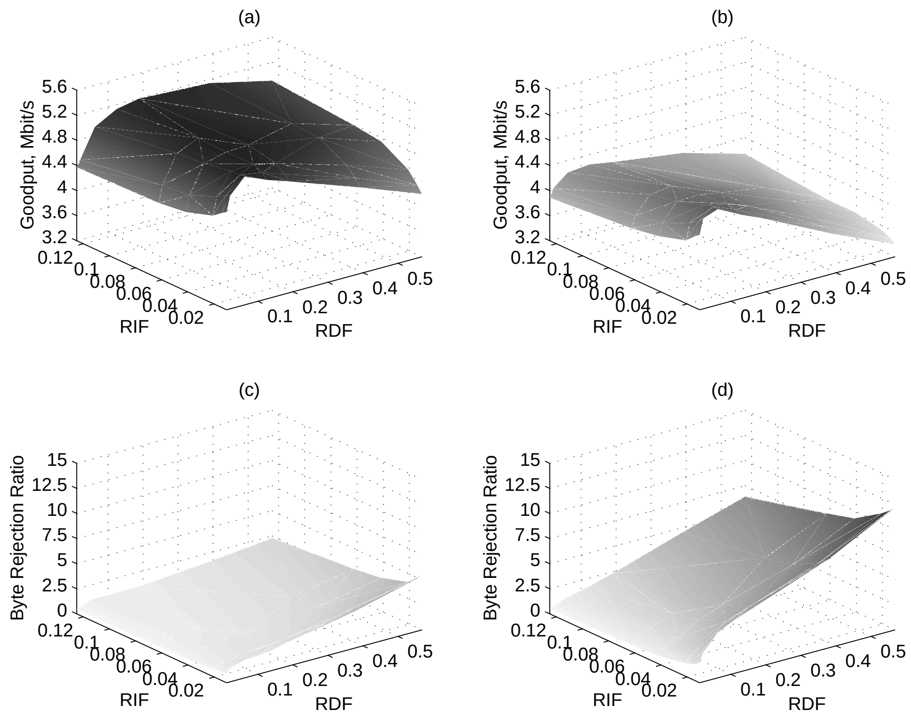

Figure 3.11: As a function of RIF and RDF: (a) G_{net} for the multi-path TE with strict-priority and RER, (b) G_{net} for the shortest-path routing, (c) BRR for the multi-path TE with strict-priority and RER (d) BRR for the shortest-path routing

When compared with the results of simulations without prior traffic information, it is seen that simulations without prior traffic information have a lower overall speed. However, the BRR performance with prior traffic information is more than two times better compared with the case without prior traffic information. The higher overall goodput for the case without prior traffic information is due to the unbalanced load distribution in the network. Some of the links are heavily congested, while some of them have a very low load. The flows using the low loaded links get very high goodput results and increase the overall average rate. Also the path set in the simulations with prior traffic information is not optimized in terms of path lengths. Many s-d pairs use paths longer than their min-hop paths. This increases the end-to-end delay that decreases the performance. However, it balances the load in the network, so it the goodputs are

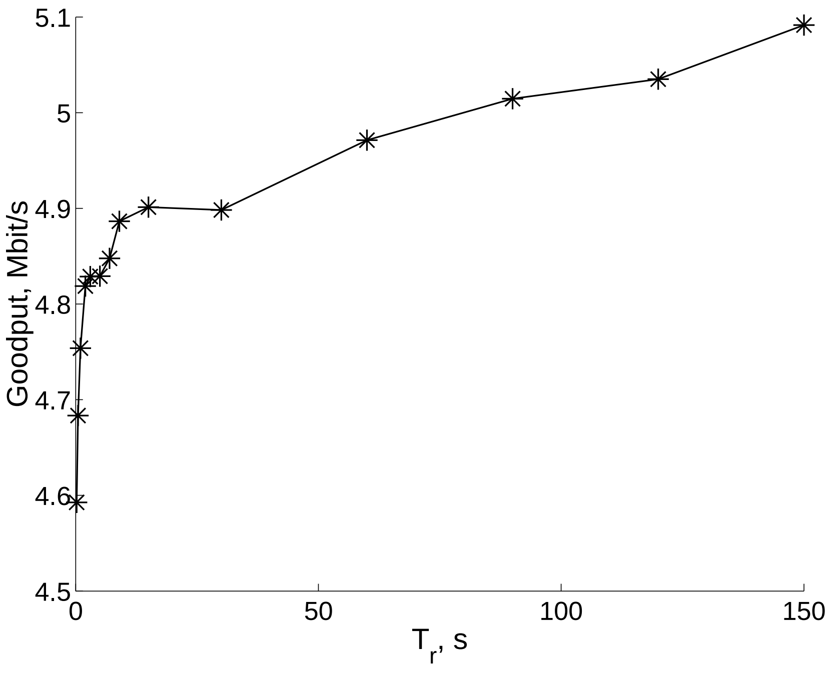

Figure 3.12: $G_{norm-avg}$ as a function of traffic scaling parameter T_r

more fair distributed among flows compared with the case without prior traffic information. Consequently, BRR decreases significantly since overall BRR is dominated by a few flows with very high congestion

As a result, if traffic matrix is available, lexicographic optimization can be used for a much fair load distribution and improved BRR results.

Simulations With Flow Rerouting

Flow rerouting can be used for the case with prior traffic matrix just like the case where there is no prior traffic matrix available. Even though the estimated traffic matrix is available and the paths are lexicographically optimized, due to the bursty nature of Internet traffic, it is not possible to guarantee that the SPs will never be congested.

The parameters used in the simulations are the same as the parameters in the previous section. As seen from in Figures 3.12 and 3.13, applying flow rerouting

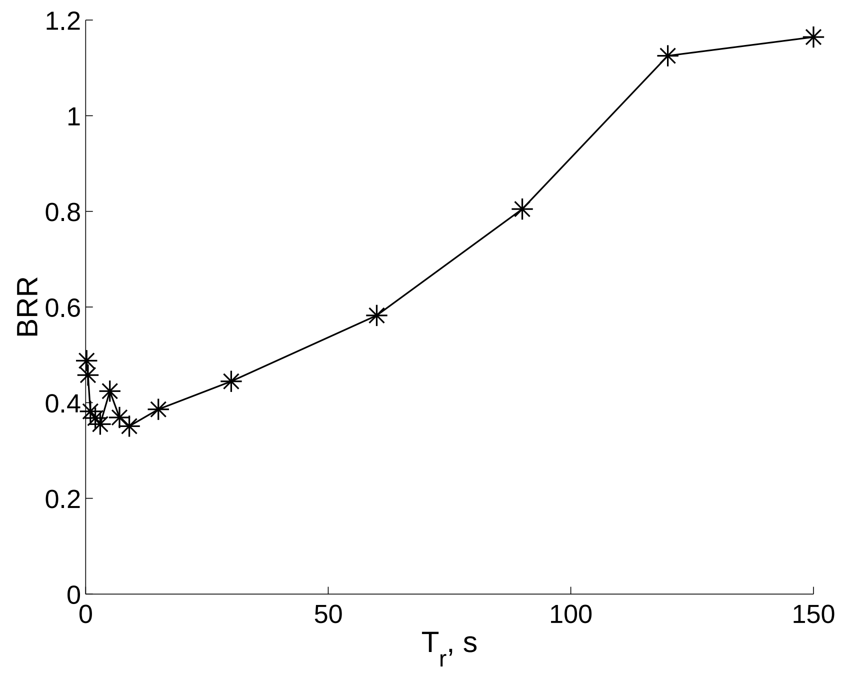

Figure 3.13: BRR as a function of traffic scaling parameter T_r

on optimized path set decreases the overall goodput, but improves the BRR. Without flow rerouting, in case of a problem in SP, we penalize the SPs for the benefit of PPs. In other words, we decrease performance of many PP flows in order to increase the performance of SP flows, which are much less. We see a sharp decrease near the point $T_r = 0$ s, because choosing a very low T_r value decreases the flow goodputs due to frequent packet reorderings. BRR value of flow rerouting is more than two times better than without rerouting. Here there is a big improvement on BRR, unlike the simulations on unoptimized path set. By using flow rerouting on optimized path set, we get the lowest BRR value in the mesh network simulations. It is around 10 times lower compared with the single-path routing without estimated traffic matrix. However, we see that there is a trade-off between goodput and BRR.

Chapter 4

Conclusions

In this thesis, we proposed a multi-path TCP load balancing traffic engineering methodology in IP networks. In this architecture, TCP traffic is split at the flow level between the primary and secondary paths. Flow based splitting prevented packet reordering problem of TCP flows. Traffic splitting is done by using a random early rerouting algorithm that controls the queuing delay difference between the two alternative paths. Probe packets are used for getting congestion information from the output queues of the links along the paths and AIMD-based rate control is applied to the paths by using this congestion information. Strict-priority is applied to the queues in the network in order to eliminate the knock-on effect. By using a three-node network and a publicly used mesh network, we show that our proposed architecture consistently outperforms the case of a single path in terms of goodput and the byte rejection ratio, and the performance of the algorithm is good for relatively large networks. We also show that load balancing with FIFO queuing and shortest delay policies does not always produce better results than that of a single path due to the knock-on effect. We showed that incorporating a-priori knowledge of the estimated traffic demand matrix into the proposed architecture can further improve its performance in terms of load balancing and byte rejection ratio. For the simulations, we improved the mesh topology simulation capability of ns-2 simulator by applying many optimizations. As a future work, its performance can be compared with other rate control algorithms like ERICA and the path set can be extended to the case where more than two paths are considered for some s-d pairs.

Bibliography

[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] 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.

[7] J. Moy, "OSPF version 2," http://www.ietf.org/rfc/rfc2328.txt, Network Working Group, RFC 2328, Apr. 1998.

[8] C. Villamizar, “OSPF Optimized Multipath (OSPF-OMP), Internet Draft ⟨draft-ietf-ospf-omp-02.txt⟩, 1998.

[9] I. Gojmerac, T. Ziegler, and P. Reichl, “Adaptive Multipath Routing Based on Local Distribution of Link Load Information,” in *Proceedings of QoFIS*, 2003.

[10] C. Villamizar, “MPLS Optimized Multipath (MPLS-OMP), Internet Draft ⟨draft-ietf-mpls-omp-01.txt⟩, 1999.

[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] S. K. Das, A. Mukherjee, S. Bandyopadhyay, K. Paul, and D. Saha, "Improving Quality-of-Service in Ad hoc Wireless Networks with Adaptive Multi-path Routing," in *Proceedings of GLOBECOM 2000*, San Francisco, California, Nov. 27-Dec 1, 2000.

[18] A. T. Z. J. Haas, and S. S. Tabrizi, "Multi-path Routing in mobile ad hoc networks or how to route in the presence of frequent topology changes," *Military Communications Conference*, 2001.

[19] S.J. Lee and M. Gerla, "Split Multi-path Routing with Maximally Disjoint Paths in Ad Hoc Networks," in *Proceedings of the IEEE ICC*, 2001, pp. 3201–3205.

[20] S. Roy, D. Saha, S. Bandyopadhyay, T. Ueda, and S. Tanaka, "Improving End-to-End Delay through Load Balancing with Multipath Routing in Ad Hoc Wireless Networks using directional Antenna," *5th International Workshop on Distributed Computing (IWDC 2003)*, 27-30 December 2003, IIM Calcutta, India.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

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

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

[32] L. Yang and G. N. Rouskas, "Path Switching in Optical Burst Switched Networks," http://www.csc.ncsu.edu/faculty/rouskas/Ar0ra/Submitted/Submitted-Yang-2004.pdf.

[33] G. Thodime, V. Vokkarane, and J. P. Jue, "Dynamic Congestion-Based Load Balanced Routing in Optical Burst-Switched Networks," *IEEE Globecom 2003*, San Francisco, CA, December 2003.

[34] F. Farahmand, V. Vokkarane, and J. P. Jue, "Practical Priority Contention Resolution for Slotted Optical Burst Switching Networks," in *Proceedings, First International Workshop on Optical Burst Switching (WOBS 2003)*, Dallas, TX, Oct. 2003.

[35] 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.

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

[37] 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.

[38] L. Georgiadis, P. Georgatsos, S. Sartzetakis, and K. Floros, "Lexicographically Optimal Balanced Networks," *IEEE Infocom 2001*, April 2001.

[39] O. Alparslan, N. Akar and E. Karasan, "Combined Use of Prioritized AIMD and Flow-Based Traffic Splitting for Robust TCP Load Balancing," *Lecture Notes in Computer Science*, Vol. 3266, pp. 124-133, Sep. 2004.

[40] O. Alparslan, N. Akar and E. Karasan, "AIMD-Based Online MPLS Traffic Engineering for TCP Flows via Distributed Multi-Path Routing," in *Annales Des Telecommunications*, To Appear.

[41] 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.

[42] R. Brown, "Calendar queues: a fast 0(1) priority queue implementation for the simulation event set problem," *Communications of the ACM*, vol. 31, no. 10, pp. 1220-1227, 1998.

[43] 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.

[44] 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.

[45] "Optimized Multipath," Internet drafts, simulations, examples, and tutorials available at www.faster-light.net/omp, 2002.

[46] The multi-path routing project at Bilkent University Information Networks Laboratory (BINLAB). Web page: http://www.binlab.bilkent.edu.tr/onur/index.html, July 2004.

Appendix A

Simulator

A.1 Installing And Using The Simulator

A.1.1 Installing The Module

1. First download the extension from [46]. Create a new directory called "multipath" under the "ns-2.27" directory of ns-2. Untar the files and copy them to the "multipath" directory.

2. Module requires modifications in some of the existing files of ns-2. First download the modifications file from [46]. It is the output of "diff" command of linux applied to a modified ns-2 version 2.27 and an unmodified ns-2 version 2.27. The output shows the name of the files and the line numbers that must be modified, added or deleted.

3. In the Makefile, add the following lines to the end of OBJ_CC

```
multipath/drop-tail2.o\
multipath/drop-tail3.o\
multipath/drop-tail4.o\
multipath/hdr_mp.o\
multipath/mpdelay.o\
multipath/mpdelay2.o\
multipath/mpsragent.o\
multipath/hdr_mpsrc.o\
multipath/cprobe.o\
```

4. In case a problem with patching the TCP source files with the diff file, just replace the tcp.cc, tcp.h, tcp-sink.cc, tcp-sink.h files under “ns-2.27/tcp” directory with the modified versions from [46]. There are some speed optimizations and functions for logging the stats of tcp flows in these files.

A.1.2 Simulation Scripts

Simulation script set can be downloaded from [46]. It includes routes of first and second paths, flow arrivals and a sample simulation script. A directory called “log” must be created under the directory where ns-2 is running. Simulation log files will be created in and written to that directory. Also a file called “out.tr” will be created and written to the same directory you are running ns-2. “out.tr” file shows the starting and ending times of the flows and their source, destination, number of successfully carried packets information. The performance of the system can be learned by processing “out.tr” file by using another program like Matlab.

A.2 Unoptimized Path Set For Hypothetical US Topology

In these tables, the primary and the secondary paths for s-d pairs are given. This path set is used in the mesh network simulations in which prior traffic matrix is not available. The path from a source node to destination node is the same as the path from the destination node to source node, so we give only the half of the paths for s-d pairs.

From-to	**Primary Path**	**Secondary Path**
sf-de	sf-de	sf-sj-de
sf-ch	sf-de-ch	sf-sj-de-sl-cl-ch
sf-cl	sf-de-ch-cl	sf-sj-de-sl-cl
sf-ny	sf-de-ch-cl-ny	sf-sj-de-sl-dc-ny
sf-sj	sf-sj	sf-la-sj
sf-sl	sf-de-sl	sf-sj-da-sl
sf-dc	sf-de-sl-dc	sf-la-hs-at-dc
sf-la	sf-la	sf-sj-la
sf-da	sf-sj-da	sf-la-hs-da
sf-at	sf-la-hs-at	sf-de-sl-dc-at
sf-hs	sf-la-hs	sf-sj-da-hs

From-to	Primary Path	Secondary Path
de-ch	de-ch	de-sl-cl-ch
de-cl	de-ch-cl	de-sl-cl
de-ny	de-ch-cl-ny	de-sl-dc-ny
de-sj	de-sj	de-sf-sj
de-sl	de-sl	de-ch-cl-sl
de-dc	de-sl-dc	de-ch-cl-dc
de-la	de-sj-la	de-sf-la
de-da	de-sl-da	de-sj-da
de-at	de-sl-dc-at	de-sj-la-hs-at
de-hs	de-sl-da-hs	de-sj-la-hs

From-to	Primary Path	Secondary Path
ch-cl	ch-cl	ch-de-sl-cl
ch-ny	ch-cl-ny	ch-de-sl-dc-ny
ch-sj	ch-de-sj	ch-cl-sl-da-sj
ch-sl	ch-cl-sl	ch-de-sl
ch-dc	ch-cl-dc	ch-de-sl-dc
ch-la	ch-de-sj-la	ch-cl-sl-de-sf-la
ch-da	ch-cl-sl-da	ch-de-sj-da
ch-at	ch-cl-dc-at	ch-de-sl-da-hs-at
ch-hs	ch-cl-sl-da-hs	ch-de-sj-la-hs

From-to	**Primary Path**	**Secondary Path**
cl-ny	cl-ny	cl-dc-ny
cl-sj	cl-ch-de-sj	cl-sl-da-sj
cl-sl	cl-sl	cl-dc-sl
cl-dc	cl-dc	cl-ny-dc
cl-la	cl-ch-de-sj-la	cl-sl-de-sf-la
cl-da	cl-sl-da	cl-dc-at-hs-da
cl-at	cl-dc-at	cl-sl-da-hs-at
cl-hs	cl-sl-da-hs	cl-dc-at-hs

From-to	**Primary Path**	**Secondary Path**
ny-sj	ny-cl-ch-de-sj	ny-dc-sl-da-sj
ny-sl	ny-cl-sl	ny-dc-sl
ny-dc	ny-dc	ny-cl-dc
ny-la	ny-dc-at-hs-la	ny-cl-ch-de-sj-la
ny-da	ny-cl-sl-da	ny-dc-at-hs-da
ny-at	ny-dc-at	ny-cl-sl-da-hs-at
ny-hs	ny-dc-at-hs	ny-cl-sl-da-hs

From-to	**Primary Path**	**Secondary Path**
sj-sl	sj-de-sl	sj-da-sl
sj-dc	sj-de-sl-dc	sj-da-sl-cl-dc
sj-la	sj-la	sj-sf-la
sj-da	sj-da	sj-la-hs-da
sj-at	sj-da-hs-at	sj-de-sl-dc-at
sj-hs	sj-da-hs	sj-la-hs

From-to	**Primary Path**	**Secondary Path**
sl-dc	sl-dc	sl-cl-dc
sl-la	sl-de-sj-la	sl-da-hs-la
sl-da	sl-da	sl-de-sj-da
sl-at	sl-dc-at	sl-da-hs-at
sl-hs	sl-da-hs	sl-dc-at-hs

From-to	**Primary Path**	**Secondary Path**
dc-la	dc-at-hs-la	dc-sl-de-sj-la
dc- da	dc-sl-da	dc-at-hs-da
dc-at	dc-at	dc-sl-da-hs-at
dc-hs	dc-at-hs	dc-sl-da-hs

From-to	**Primary Path**	**Secondary Path**
la-da	la-hs-da	la-sj-da
la-at	la-hs-at	la-sj-de-sl-dc-at
la-hs	la-hs	la-sj-da-hs

From-to	**Primary Path**	**Secondary Path**
da-at	da-hs-at	da-sl-dc-at
da-hs	da-hs	da-sj-la-hs

From-to	**Primary Path**	**Secondary Path**
at-hs	at-hs	at-dc-sl-da-hs

A.3 Optimized Path Set For Hypothetical US Topology

This path set is used in the mesh network simulations in which prior traffic matrix is available. In some cases, the path from a source node to destination node is the different from the path from the destination node to source node, so we give the paths for all s-d pairs.

From-to	**Primary Path**	**Secondary Path**
sf-ny	sf-de-ch-cl-ny	sf-sj-de-sl-dc-ny
sf-dc	sf-de-sl-dc	sf-sj-de-ch-cl-dc
sf-cl	sf-sj-de-sl-cl	sf-de-ch-cl
sf-at	sf-la-hs-at	sf-de-sl-dc-at
sf-ch	sf-de-ch	sf-sj-de-sl-cl-ch
sf-sl	sf-de-sl	sf-sj-da-sl
sf-da	sf-sj-da	sf-la-hs-da
sf-hs	sf-sj-da-hs	sf-la-hs
sf-de	sf-sj-de	sf-de
sf-sj	sf-sj	sf-la-sj
sf-la	sf-la	sf-sj-la

From-to	**Primary Path**	**Secondary Path**
de-ny	de-ch-cl-ny	de-sl-dc-ny
de-dc	de-ch-cl-dc	de-sl-dc
de-cl	de-sl-cl	de-ch-cl
de-at	de-sl-dc-at	de-ch-cl-sl-da-hs-at
de-ch	de-ch	de-sl-cl-ch
de-sl	de-sl	de-ch-cl-sl
de-da	de-sl-da	de-ch-cl-dc-at-hs-da
de-hs	de-sl-da-hs	de-ch-cl-dc-at-hs
de-sj	de-sj	de-sf-sj
de-la	de-sf-la	de-sj-la
de-sf	de-sj-sf	de-sf

From-to	**Primary Path**	**Secondary Path**
ch-ny	ch-cl-ny	ch-de-sl-dc-ny
ch-dc	ch-cl-dc	ch-de-sl-dc
ch-cl	ch-cl	ch-de-sl-cl
ch-at	ch-cl-dc-at	ch-de-sl-da-hs-at
ch-sl	ch-cl-sl	ch-de-sl
ch-da	ch-cl-sl-da	ch-de-sl-dc-at-hs-da
ch-hs	ch-de-sl-da-hs	ch-cl-dc-at-hs
ch-de	ch-de	ch-cl-sl-de
ch-sj	ch-de-sj	ch-cl-sl-de-sf-sj
ch-la	ch-de-sj-la	ch-cl-sl-de-sf-la
ch-sf	ch-de-sf	ch-cl-sl-de-sj-sf

From-to	Primary Path	Secondary Path
cl-ny	cl-ny	cl-dc-ny
cl-dc	cl-dc	cl-sl-dc
cl-at	cl-dc-at	cl-sl-da-hs-at
cl-ch	cl-ch	cl-sl-de-ch
cl-sl	cl-sl	cl-dc-sl
cl-da	cl-sl-da	cl-dc-at-hs-da
cl-hs	cl-sl-da-hs	cl-dc-at-hs
cl-de	cl-ch-de	cl-sl-de
cl-sj	cl-sl-da-sj	cl-ch-de-sj
cl-la	cl-ch-de-sj-la	cl-sl-da-hs-la
cl-sf	cl-sl-de-sf	cl-ch-de-sj-sf

From-to	Primary Path	Secondary Path
ny-dc	ny-dc	ny-cl-dc
ny-cl	ny-cl	ny-dc-cl
ny-at	ny-dc-at	ny-cl-sl-da-hs-at
ny-ch	ny-cl-ch	ny-dc-sl-de-ch
ny-sl	ny-dc-sl	ny-cl-sl
ny-da	ny-dc-sl-da	ny-cl-dc-at-hs-da
ny-hs	ny-dc-at-hs	ny-cl-sl-da-hs
ny-de	ny-cl-ch-de	ny-dc-sl-de
ny-sj	ny-cl-sl-de-sj	ny-dc-cl-ch-de-sf-sj
ny-la	ny-dc-at-hs-la	ny-cl-sl-de-sf-la
ny-sf	ny-cl-ch-de-sf	ny-dc-sl-de-sj-sf

From-to	Primary Path	Secondary Path
sj-ny	sj-de-ch-cl-ny	sj-sf-de-sl-dc-ny
sj-dc	sj-da-sl-cl-dc	sj-de-sl-dc
sj-cl	sj-de-ch-cl	sj-sf-de-sl-cl
sj-at	sj-da-hs-at	sj-de-sl-dc-at
sj-ch	sj-de-ch	sj-sf-de-sl-cl-ch
sj-sl	sj-de-sl	sj-sf-de-ch-cl-sl
sj-da	sj-da	sj-la-hs-da
sj-hs	sj-da-hs	sj-la-hs
sj-de	sj-de	sj-sf-de
sj-la	sj-la	sj-sf-la
sj-sf	sj-sf	sj-la-sf

From-to	Primary Path	Secondary Path
sl-ny	sl-dc-ny	sl-cl-ny
sl-dc	sl-dc	sl-cl-dc
sl-cl	sl-cl	sl-dc-cl
sl-at	sl-dc-at	sl-da-hs-at
sl-ch	sl-cl-ch	sl-de-ch
sl-da	sl-da	sl-dc-at-hs-da
sl-hs	sl-da-hs	sl-dc-at-hs
sl-de	sl-de	sl-cl-ch-de
sl-sj	sl-da-sj	sl-de-sj
sl-la	sl-da-sj-sf-la	sl-de-sj-la
sl-sf	sl-da-sj-sf	sl-de-sf

From-to	Primary Path	Secondary Path
dc-ny	dc-ny	dc-cl-ny
dc-cl	dc-cl	dc-sl-cl
dc-at	dc-at	dc-sl-da-hs-at
dc-ch	dc-cl-ch	dc-sl-de-ch
dc-sl	dc-sl	dc-cl-sl
dc-da	dc-sl-da	dc-at-hs-da
dc-hs	dc-sl-da-hs	dc-at-hs
dc-de	dc-sl-de	dc-cl-ch-de
dc-sj	dc-cl-ch-de-sj	dc-sl-de-sf-sj
dc-la	dc-at-hs-la	dc-cl-ch-de-sf-la
dc-sf	dc-sl-de-sf	dc-cl-ch-de-sj-sf

From-to	Primary Path	Secondary Path
la-ny	la-hs-at-dc-ny	la-sj-de-ch-cl-ny
la-dc	la-hs-at-dc	la-sj-da-sl-dc
la-cl	la-sj-de-sl-cl	la-sf-de-ch-cl
la-at	la-hs-at	la-sf-de-sl-dc-at
la-ch	la-sj-de-ch	la-sf-de-sl-cl-ch
la-sl	la-sj-de-sl	la-sf-de-ch-cl-sl
la-da	la-sj-da	la-hs-da
la-hs	la-hs	la-sj-da-hs
la-de	la-sf-sj-de	la-sj-sf-de
la-sj	la-sj	la-sf-sj
la-sf	la-sf	la-sj-sf

From-to	Primary Path	Secondary Path
da-ny	da-sl-cl-dc-ny	da-sj-de-ch-cl-ny
da-dc	da-sl-dc	da-sj-de-ch-cl-dc
da-cl	da-sl-cl	da-sj-de-ch-cl
da-at	da-hs-at	da-sl-dc-at
da-ch	da-sl-cl-ch	da-sj-de-ch
da-sl	da-sl	da-sj-de-sl
da-hs	da-hs	da-sj-la-hs
da-de	da-sl-de	da-sj-de
da-sj	da-sj	da-hs-la-sj
da-la	da-sj-la	da-hs-la
da-sf	da-sj-sf	da-hs-la-sf

From-to	Primary Path	Secondary Path
at-ny	at-dc-ny	at-hs-da-sl-cl-ny
at-dc	at-dc	at-hs-da-sl-dc
at-cl	at-dc-cl	at-hs-da-sl-cl
at-ch	at-dc-cl-ch	at-hs-da-sl-de-ch
at-sl	at-dc-sl	at-hs-da-sl
at-da	at-hs-da	at-dc-sl-da
at-hs	at-hs	at-dc-sl-da-hs
at-de	at-dc-cl-ch-de	at-hs-da-sl-de
at-sj	at-hs-da-sj	at-dc-cl-ch-de-sj
at-la	at-hs-la	at-dc-cl-ch-de-sf-la
at-sf	at-hs-la-sf	at-dc-cl-ch-de-sf

From-to	Primary Path	Secondary Path
hs-ny	hs-at-dc-ny	hs-da-sl-cl-ny
hs-dc	hs-da-sl-dc	hs-at-dc
hs-cl	hs-da-sl-cl	hs-la-sf-de-ch-cl
hs-at	hs-at	hs-da-sl-dc-at
hs-ch	hs-da-sl-de-ch	hs-la-sf-de-sl-cl-ch
hs-sl	hs-da-sl	hs-la-sf-de-sl
hs-da	hs-da	hs-la-sj-da
hs-de	hs-da-sl-de	hs-la-sf-de
hs-sj	hs-da-sj	hs-la-sj
hs-la	hs-la	hs-da-sj-la
hs-sf	hs-la-sf	hs-da-sj-sf