

## ABSTRACT

Elhaddad, Mahmoud Shawky. Adaptive Multipath Traffic Allocation in TCP/IP Networks. (Under the direction of Dr. Injong Rhee and Dr. Munindar Singh.)

Balancing network traffic among multiple paths connecting ingress-egress pairs promises better utilization of network links and improved service quality for user-flows. Research into optimized adaptive traffic allocation yielded techniques that tradeoff convergence speed to stability and/or fail to incorporate knowledge about the behavior of congestion control mechanisms, making their evaluation a difficult, if possible task – both analytically and experimentally.

In this thesis, we consider load balancing in multipath IP networks where the number of flows between ingress-egress pairs is a slow-changing process and TCP-friendliness is adopted-by or imposed-on all flows. Provided that packet ordering is preserved, we present an analytical characterization of the throughput-optimal fractional allocations of flow packets in terms of the TCP-fair share along each candidate path; and show that optimally splitting all network flows results in efficient and globally fair sharing of network resources. For the estimation of the fair share along network paths, we demonstrate that TCP-fairness in sharing bottleneck bandwidth can be modeled as weighted max-min, where the weight vector corresponds to the Bulk Transfer Capacity (BTC) of the bottlenecked path. Policing at the ingress routers limits connections to their computed fair rates, thereby eliminating TCP bias against connections passing through multiple bottlenecks and minimizing packet losses at internal routers. Given minimal loss rate and small bounded delay skew among alternative paths, packet resequencing at the egress routers can be applied effectively.

We also introduce AMTA – a centralized traffic allocation service for MPLS networks based on the optimality and fairness results described above, and extended to handle flows with limited throughput demand. The robustness of AMTA under relevant scenarios is demonstrated through simulations.

# ADAPTIVE MULTIPATH TRAFFIC ALLOCATION IN TCP/IP NETWORKS

by

**Mahmoud S. Elhaddad**

A thesis submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the Degree of  
Master of Science

**Computer Science**

Raleigh

2001

**APPROVED BY:**

---

Chair of Advisory Committee

To my family

## BIOGRAPHY

Mahmoud Elhaddad was born on July 30, 1972 in Alexandria, Egypt. He received his bachelor degree in Computer Engineering (summa cum laude) from the Arab Academy for Science and Technology in 1996, where he spent the next three years as a teaching assistant before joining the computer science masters program at North Carolina State University.

## ACKNOWLEDGEMENTS

I would like to thank my advisors Dr. Injong Rhee and Dr. Munindar Singh for their support and encouragement. As his research assistant, Dr. Rhee taught me the tools of the networking research trade. His guidance and inspiring critique were essential for the completion of this research. Dr. Singh introduced me to the problem of adaptive traffic allocation and provided close supervision and advice throughout my stay at NC State.

I am grateful to Dr. George Rouskas and Dr. Wenke Lee for serving as members of my masters defense committee. During the thesis preparation, Dr. Rouskas provided valuable comments and useful literature pointers. His vigorous questioning also made the thesis defense an extremely satisfying experience. Many thanks to Dr. Davis, Mrs. Chery Rousseau, Mrs. Margery Page, and Mrs. Angie Barefoot for their help and courtesy.

In its final stages, this research was supported by a grant from Cisco University Research. I would like to thank Cisco's Technology Assessment Group of Research Triangle Park for their hospitality and their enlightening comments.

When it comes to family I am lucky to have one that highly values research and academia. I would like to thank my mother, sister, her husband and in-laws for there support and encouragement. I was also lucky to have come to Raleigh. Mr. and Mrs. Elmaghrabi made it feel like home. Dr. Abdelsalam "Solom" Heddaya has been guiding my steps since I was an undergraduate student in Egypt. I would like to thank him for his sincere advice and also for pointing out work on traffic engineering in MPLS networks which I later adopted as the target network platform for this research.

Throughout this research project, Nevine, my wife, patiently listened for hours and discussed claims that would later prove too simplistic or plain wrong. Her confidence helped me overcome frustration and maintain my goal of achieving useful results.

Friends made a big difference during my stay in Raleigh. I enjoyed the outings and discussions with my friends Volkan, Venu, Tu, Javan, Jason, Brent and Sarat just to name a few.

# Contents

<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Thesis Overview and Contributions . . . . .	6
1.2 Thesis Outline . . . . .	8
<b>2 Fair and Efficient Equilibria in Multipath Networks</b>	<b>9</b>
2.1 Steady-State Behavior of TCP Connections . . . . .	10
2.2 Optimal Multipath Traffic Allocation . . . . .	11
2.3 TCP Fair Share Estimation . . . . .	15
2.3.1 Modeling TCP Fairness as Weighted Max-Min . . . . .	16
2.3.2 Estimation of TCP Fair Shares in Multipath Networks . . . . .	20
2.4 Summary . . . . .	20
<b>3 AMTA: Adaptive Multipath Traffic Allocation</b>	<b>22</b>
3.1 The Network Environment . . . . .	22
3.1.1 Traffic Aggregation and Routing . . . . .	22
3.1.2 Packet Classification and Traffic Measurements . . . . .	23
3.1.3 Rate Policing, Path Selection, and Resequencing . . . . .	24
3.2 The Trunk Share Estimation Algorithm . . . . .	25
3.3 The Effectiveness of AMTA . . . . .	28
3.3.1 Coping with Traffic Swings . . . . .	30
3.3.2 Elimination of TCP Bias Against Flows with Multiple Bottlenecks	30
3.4 Handling Flows with Limited Throughput Demand . . . . .	33
3.5 Summary . . . . .	37
<b>4 Concluding Remarks</b>	<b>39</b>
<b>Bibliography</b>	<b>41</b>

## List of Figures

2.1	Experiment ECMP vs OPTIMAL. Simulated topology. . . . .	13
2.2	Experiment ECMP vs OPTIMAL. FEC 1 flow throughput obtained through simulation compared to that obtained analytically for ECMP and OPTIMAL. . . . .	15
2.3	Experiment FAIRNESS MODEL VALIDATION. Simulated topology. . . . .	18
2.4	Experiment FAIRNESS MODEL VALIDATION. Case 1: Connections with equal round-trip time. . . . .	19
2.5	Experiment FAIRNESS MODEL VALIDATION. Case 2: Trunk 1 connections have twice the RTT of trunk 2. . . . .	19
3.1	Algorithm for Trunk Share Estimation . . . . .	27
3.2	Experiment AMTA vs OPTIMAL. Simulated topology. . . . .	28
3.3	Experiment AMTA vs OPTIMAL. FEC 1 flow throughput. . . . .	29
3.4	Experiment MULTIPLE BOTTLENECKS. Network topology. . . . .	31
3.5	Algorithm for the estimation of effective number of FEC flows . . . . .	34
3.6	Experiment SMALL WINDOW. Simulated topology. . . . .	35
3.7	Experiment SMALL WINDOW. Overall throughput for each FEC and their respective rate bounds. . . . .	36
3.8	Experiment SMALL WINDOW. Effective volume for each FEC. . . . .	37

# List of Tables

2.1	Experiment ECMP vs OPTIMAL. FEC 1 Traffic allocation parameters under OPTIMAL . . . . .	14
3.1	Experiment AMTA-OPTIMAL. FEC 1 traffic allocation parameters obtained through AMTA for different volumes of FEC 2. . . . .	29
3.2	Experiment AMTA-OPTIMAL. Trunk weights for different volumes of FEC 2. . . . .	29
3.3	Experiment AMTA-OPTIMAL. Trunk shares for different volumes of FEC 2. . . . .	29
3.4	Experiment MULTIPLE BOTTLENECKS. Trunk weights in Mb/s. . . . .	32
3.5	Experiment MULTIPLE BOTTLENECKS. Trunk shares in Mb/s. . . . .	32
3.6	Experiment MULTIPLE BOTTLENECKS. Traffic allocation parameters. . . . .	32
3.7	Experiment MULTIPLE BOTTLENECKS. Flow throughput results. . . . .	33

# Chapter 1

## Introduction

Internet Protocol (IP) networks are invariably designed with multiple paths connecting router pairs to ensure robustness in the face of node or link failures. The ever increasing demand for higher throughput and lower delay led researchers to consider balancing the traffic load among the alternative paths as a departure from traditional single-path routing. Multipath forwarding has been proposed in the context of techniques for minimum-delay routing and throughput maximization through traffic dispersion. A multipath routing technique would normally be concerned with finding a set of directed paths (a multipath) connecting each pair of routers, however, many proposed techniques combine traffic allocation with routing. For instance, an off-line algorithm that minimizes the expected total network delay based on static traffic estimates was presented in [28]; in addition to specifying for every router a set of next-hops toward each destination, the algorithm also performs traffic allocation by dictating the fraction of load forwarded to each path or downstream neighbor. Similarly in practice, Equal-Cost MultiPath routing (ECMP), which is currently part of the widely deployed standard internet gateway protocol, OSPF [17], performs traffic allocation by splitting load evenly among the alternative paths connecting router pairs.

Naturally, traffic allocation based on long-term traffic estimates does not take into account load fluctuations on shorter time-scales. Research into adaptive and adaptive-multipath routing resulted in a variety of techniques (see for example [14],

[5], [27], and [25]). In contrast to static routing, algorithms based on dynamic cost metrics tradeoff convergence speed to route and traffic stability, and incur substantial communication overhead in disseminating the link cost changes to routers to prevent them from acting on outdated information [20].

The limitations of adaptive routing, in addition to the fact that a routes optimization criterion ultimately reflects multiple objectives that all need to be reflected in the link cost metric, and may be subject to policy as well as QoS constraints [23], leads us to conclude that a routing algorithm should only provide a stable set of loop-free paths connecting each router pair, and that the task of short-term load balancing should be handled separately by an adaptive traffic allocation mechanism.

In IP networks, end-to-end congestion control, together with router queue management mechanisms, are used to apportion network resources among competing flows of different delay and throughput demands. Traffic engineering, on the other hand, involves paths computation and traffic management such that user performance requirements and routing constraints are met while ensuring reliable operation and good utilization of network resources [3]. To achieve its goals, traffic engineering processes, such as measurements and traffic allocation must take into consideration the behavior of congestion control mechanisms.

The deployment of Transmission Control Protocol (TCP) which implements Additive Increase/Multiplicative Decrease (AIMD) flow/congestion control helped prevent the congestive collapse of the Internet[9]. Consequently, TCP behavior in sharing bottleneck capacity has become an accepted practical definition of fairness as opposed to other idealistic notions such as max-min fairness[5]. TCP however is bursty, making it ill-suited for multimedia streams. Protocols for smoother control of such flows have been developed and evaluated for “TCP-friendliness” [19, 10]. A congestion control protocol is said to be TCP-friendly if the controlled stream approximates the throughput of a concurrent TCP connection over the same path. Mechanisms for promoting the use of TCP friendly end-to-end congestion control and enforcing TCP friendliness through selectively dropping packets that belong to misbehaving flows have also been developed [9, 8, 15].

Given the prevalence of TCP-style congestion control, using queuing delay or

buffer occupancy as a basis for traffic allocation is ineffective. Two paths may appear to be equally congested while the bottleneck on one path is saturated by a smaller number of flows. Since the majority of flows on the Internet are throughput-sensitive [24], we argue that in a multipath best-effort network, traffic allocation must be concerned with throughput maximization while preserving fairness among TCP flows. While the literature includes many techniques that attempts to solve similar problems, none studied the interaction between traffic allocation and congestion control mechanisms[13].

In [24], measurements also revealed a small variance in the number of active flows crossing an Internet link on a 5-min time scale. This observation is the key to the viability of adaptive traffic allocation. Given the quasi-static character of the number of flows and a model of TCP bandwidth sharing behavior, the available capacity on each path can be estimated and traffic allocation decision accordingly made.

Recognizing the negative effects of packet reordering on the throughput and competitiveness of TCP connections, many traffic allocation schemes perform load balancing at the connection level [28, 7, 25]. Flows between an ingress-egress pair are assigned to the candidate paths using a hash function based on the source and destination addresses in the corresponding packets. By appropriately setting the *hash thresholds*, a lightly loaded path covers a larger range of addresses and hence receives a larger number of connections. Unfortunately, even by exploiting the quasi-static nature of the traffic volume, and assuming TCP results in max-min fair allocations, finding an allocation of flows to paths that maximizes the minimum share or the total throughput is a combinatorial optimization problem. In addition, the resulting allocations are not necessarily fair; flows between the same ingress-egress pairs but follow different paths are not likely to achieve equal throughput.

In packet-level load balancing, each connection's stream is split among the candidate paths. The fractional allocations of flow packets to the candidate paths are called the flow *allocation parameters*. The flow allocations can be accurately achieved by assigning flow packets to paths using a round-robin mechanism. Generally, packets are of different sizes, therefore Deficit or Surplus Round-Robin (DRR and SRR respectively) are used instead of Weighted Round-Robin. Flows between the same

ingress-egress points share the same candidate paths (also called *trunks*) and share a common set of allocation parameters, hence forming a Forwarding Equivalence Class (FEC). Since the packets of these flows are multiplexed at the ingress in no particular order, path selection using the round-robin mechanism at the FEC level—that is without distinguishing between individual flows—results in the realization of the intended flow allocations.

For TCP-controlled flows, reordering of data packets (or forward-path reordering) results in the reception of duplicate acknowledgements at the sender. In NewReno, the most popular flavor of TCP, if the number of duplicate acknowledgments received at the sender reaches the DUPACK threshold which is commonly set to 3, the sender presumes that data following the last acknowledged byte have been lost, consequently halving the transmission window to alleviate congestion, and retransmitting the lost data. Called *Fast Recovery* and *Fast Retransmit* respectively, these procedure are intended for quick recovery from occasional packet losses thereby maintaining a continuous flow of TCP segments. In case the duplicate acknowledgements occur frequently due to packet reordering, unnecessary rate halving results in significant reduction in connection throughput. We call this phenomenon *false congestion indication*. The unnecessarily retransmissions are also a waste of network resources. Spurious fast retransmits also may affect the estimation of round-trip time in case TCP time-stamps are not in use. RTT samples from retransmitted segments are discarded. Naturally, the accuracy of the estimation algorithm is greatly affected by the reduction in number of samples. In addition, out of order packet reception incurs a significant overhead at the receiver in buffering and sorting packets before their delivery to the application. TCP-friendly protocols suffer from the same effects since they seek to emulate—yet in a smoother way—the behavior of TCP congestion control. Real-time flows such as audio and video suffer additionally since in applications where no appropriate buffering is implemented, missing packets are considered lost. Out-of-order packets are useless and the playout quality degrades sharply.

Reordering on the reverse-path can have equally serious effects. The sender increases the size of the transmission window when it receives original acknowledgements. Out-of-order packets are covered with a smaller number of cumulative ACKs

as the missing data arrives and hence the transmission window grows more slowly and the connection become less competitive in sharing the bottleneck capacity.

Packet reordering is widespread due to the parallelism in router architectures, link striping, and the proliferation of wireless devices. In response, modifications to TCP behavior have been proposed as well as extensions to add robustness in the face of frequent and persistent packet reordering. Notably, the D-SACK extension has been proposed to let the sender identify false congestion indications and undo the unnecessary rate reduction. ACK-clocking in SACK-TCP in general is not affected by repeated and cumulative acknowledgments since a duplicate ACK carries an identification of the segment that generated the acknowledgement at the receiver allowing the sender to augment its window size as original packets reach the destination.

Relying on the end-systems to provide smart buffering for real-time flows or adopt recent TCP standards however is not recommended given the slow nature of software releases and that changes have to be implemented at both the receiving and sending ends before their benefit is accrued. Packet resequencing at the edges is an alternative solution that has been proposed in the context of packet striping. In [21], authors describe algorithms for resequencing packets at the egress of the multipath network without the need for sorting or explicit packet numbering. Given that flow packet apportioning among candidate paths at the ingress is achieved using a causal load sharing mechanism such as SRR, the egress can determine the path on which the next in-order packet should arrive. Out-of-order packets are queued in a separate logical resequencing queue for each path and dequeued in the proper order of reception. This buffering is done at the routers' input line-cards. Pointer chaining is used to access packets that have been dequeued from resequencing queues and are hence eligible to access the switch fabric. This way, slow buffer copying operations are avoided.

The resequencing algorithm works correctly only in the absence of packet loss. Hence, resynchronization between the flow ingress and egress after each loss is necessary to resume proper operation. Detecting of packet losses at the egress requires the inspection of the transport headers in each packet, alternatively resynchronization is done periodically through the *marker* packets sent by the ingress. In networks with rare internal packet drops, the resynchronization overhead should be small. Another

factor for the effectiveness of resequencing is the amount of skew among the alternative paths. A bounded skew ensures bounded resequencing queues at the egress while a small skew means a small average packet delay in the resequencing queues. Skew boundedness is guaranteed through finite buffers at the intermediate routers. To minimize the skew, the propagation delay along the alternative paths must be approximately equal – a constraint to be imposed on the routing algorithm.

Load balancing is intended to improve the throughput of user flows by utilizing capacity on redundant paths. In contrast with connection-level load balancing, packet-level traffic allocation offers a simple constructive characterization of the desired allocations. By extending the notion of TCP-fair shares in single path networks to the multipath case, a flow allocation is said to be *throughput-optimal* if the connection acquires – on each candidate path – the fair share of a TCP flow forwarded entirely over that path. Moreover, the above definition implies that when all flows achieve optimal throughput, *global fairness* is also achieved. As this study shows, an efficient algorithm for computing optimal allocations exists, and optimal connection throughput can be achieved provided that reordering penalties are eliminated.

## 1.1 Thesis Overview and Contributions

In this thesis we present and validate an analytical characterization of globally-fair and efficient equilibria in multipath TCP/IP networks. The optimal allocation of a connection is expressed in terms of the TCP-fair share along each candidate path. Revising flow allocation based on direct measurements of the fair share may result in oscillations as many flows simultaneously shift their traffic load onto lightly loaded paths. A mathematical model of TCP fairness is thus needed to estimate the fair shares as all flows update their allocations. We present a model of TCP fairness based on the weighted max-min criterion. Specifically, we show the TCP is weighted max-min fair when the weight vector corresponds to the measured shares. We validate this model through simulations. The problem of computing efficient and globally fair equilibria is thus reduced to one of finding weighted max-min fair allocations, for which there is a known efficient centralized algorithm. The viability of this approach is

ensured the small variance in number of flows crossing a network link over a relatively large time interval. TCP has a well-known bias against connections passing through multiple bottlenecks, however we show that this bias can be eliminated through rate policing given knowledge of the fair throughput for each flow. Rate policing and traffic shaping at the ingress routers also minimize packet drops at the internal routers. Thereby allowing the proper and efficient operation of the resequencing mechanism at the egress nodes.

This thesis also introduces AMTA, a centralized traffic allocation service for MPLS networks based on the above optimality and fairness results. MPLS is chosen to as the target platform to leverage its support for traffic engineering processes, particularly, its explicit support for FECs and trunks, and per-packet path selection at the ingress. Since the volume of traffic crossing a link is the aggregation of multiple FECs, whose volumes can be modeled as independent random variables, small variance in the aggregate implies smaller variance in the volumes of its component FECs. As described earlier, in packet-level load balancing, traffic allocation and load distribution is performed at the FEC level, resulting in high efficiency of the bandwidth measurement, rate policing and traffic allocation processes in AMTA. The robustness of AMTA in the face of sudden traffic swings, and in the existence of multiple bottlenecks is demonstrated through simulations.

In practical settings, not all flows exhibit unlimited demand for bandwidth. The throughput of a connection may be constrained by the receiver, the application sending rate, or by congested links outside the AMTA network. On the aggregate, a FEC flows may behave as a smaller number of flows with unlimited demand. An algorithm for the estimation of the effective number of flows in each FEC is provided and its accuracy is demonstrated through simulations under difference scenarios.

Periodically the AMTA server collects the effective FEC volumes and trunk BTC values from the ingress routers and, given the set of paths for each FEC, solves the weighted max-min allocation problem yielding the optimal allocation parameters for every FEC. Therefore, unlike other proposed traffic allocation strategies, AMTA does not converge incrementally to the optimal allocations. Also, AMTA is computationally efficient and does not incur a large communication overhead while requiring traffic

management and measurement functionality only at the edge routers.

## 1.2 Thesis Outline

In the next chapter, we review the behavior of TCP congestion control and derive a condition for the optimal allocation of a TCP flow in terms of the TCP fair share along each of the candidate paths. We also provide a method for the estimation of the fair shares based on bandwidth measurements and the weighted max-min fairness criterion. In chapter 3, we present AMTA, a centralized adaptive traffic allocation service for MPLS networks that is built upon the above optimality and fairness results and address practical considerations in the design of AMTA. Finally, we summarize the thesis contributions and present an outline of future work in chapter 4.

## Chapter 2

# Fair and Efficient Equilibria in Multipath Networks

The objective of multipath forwarding is to increase the throughput of TCP connections by utilizing the capacity available on redundant network paths. Therefore, an optimal multipath traffic allocation strategy is one that enables each TCP connection to acquire its fair share of capacity along each routing path; thereby resulting in efficient and fair link capacity allocations to competing flows.

In this chapter, we present a characterization of the effects of multipath forwarding on TCP throughput and derive conditions for the optimal apportioning of a TCP flow packets to candidate paths in the absence of false congestion indication and other effects of packet reordering. The optimality condition is defined in terms of the connection's fair share along each path. We also propose and validate a method for the estimation of TCP fair share based on path measurements and a weighted max-min allocation model of TCP fairness. In the next chapter we use the TCP fairness model and the optimality result to define AMTA – a centralized solution for adaptive traffic allocation.

## 2.1 Steady-State Behavior of TCP Connections

TCP implements Additive Increase/Multiplicative Decrease (AIMD) congestion control[6]. At steady state, a TCP connection remains in the congestion avoidance phase. The TCP congestion window size  $w$  is increased by  $1/w$  each time an ACK is received. Upon inferring a packet loss through duplicate acknowledgments, the window is halved.

In [6], Chiu and Jain showed that AIMD congestion control results in nearly efficient equilibrium throughput allocations where the transmission rate of connections sharing a bottleneck oscillate around the optimal fair allocations. Their analysis assumed complete synchronization among the competing flows. Recently, a similar result was proved for connections with heterogeneous round-trip times, in that case without the synchronized rate update assumption [26]. Practically, drop-tail router queues can cause synchronized losses among many connections, forcing them to reduce their rates simultaneously, hence resulting in inefficient utilization of the bottleneck capacity. Loss synchronization can be eliminated by changing the packet-drop policy to Random Early Detection (RED)[11]. Therefore we consider henceforth that the equilibria achieved by TCP in the case of unipath connections are efficient; or, alternatively, that the effective capacity of a bottleneck link is the sum of the rates of the TCP connections passing through it.

At the sender, the TCP window size is increased additively as described above until it reaches a maximum window size  $W$ , at which a packet loss occurs and the window size is reduced to  $W/2$ . The process is then repeated, resulting in a sawtooth-like variation in the window-size. A stochastic model of single-path TCP connections was introduced in [18]. Assuming congestion-induced packet loss over a path can be represented as a Bernoulli process with rate  $l$ , the model is used to derive the following expected maximum window size at which a loss occurs on the path

$$\bar{W} \approx B \sqrt{\frac{8}{3bl}}$$

where  $B$  is the packet size of the connection in bytes,  $RTT$  is the round-trip time in seconds,  $l$  is the connection loss rate and the receiver generates an ACK every  $b$

received packet. In the next section we use the deterministic model to characterize the optimal flow allocations of multipath connections.

## 2.2 Optimal Multipath Traffic Allocation

According to the stochastic model, in a multipath network, a TCP sender receives synchronized loss indications from the bottlenecked path(s) when the transmission window reaches a size corresponding to the maximum window size on the bottleneck(s). The sender reduces its transmission rate in response to only one loss signal per round-trip time, therefore, aside from the penalties of reordering, multipath forwarding does not adversely affect the throughput of a TCP connection.

Consider a TCP-controlled flow split among a set of paths  $P$ , we define the *flow density* on a path  $p_i \in P$  as the ratio  $\phi_i/c_i$ , where  $0 < \phi_i \leq 1$  is the fraction of the flow assigned to path  $p_i$ , and  $c_i$  is the flow's fair capacity share on that path. The following lemma formally characterizes the behavior of TCP flow-control in multipath-routed networks in terms of path flow density.

**Lemma 1** *In a multipath-routed network, the maximum throughput of a TCP-controlled flow is stochastically bounded by the minimum of the inverse flow densities along its paths.*

*Proof.* Let  $\overline{W}$  be the expected size of the transmission window of the TCP flow when a loss occurs. As described in section 2.1, at steady state, the transmission window grows linearly over time from  $\overline{W}/2$  to  $\overline{W}$  before receiving one or more simultaneous packet drops and repeating the cycle. It follows that if  $p_i \in P$  is a bottlenecked path with flow density  $\phi_i/c_i$ , then a packet drop is likely to occur on  $p_i$  when the transmission window reaches  $\overline{W}$ , hence

$$c_i = \phi_i \frac{3\overline{W}}{4RTT} = \phi_i R \tag{2.1}$$

where  $RTT$  is the round-trip time and  $R$  is the expected connection throughput.

Moreover, if  $p_i$  is not a bottleneck then the connection always receives a packet drop before reaching the available capacity on the path, and

$$c_i > \phi_i R \tag{2.2}$$

The case  $c_i < \phi_i R = \phi_i \frac{3\bar{W}}{4RTT}$  is unlikely to occur or persist since  $\phi_i \bar{W}$  is the expected maximum window size on  $p_i$  in steady state. From (2.1) and (2.2)

$$R \leq \frac{c_i}{\phi_i}, \quad \forall i : p_i \in P \tag{2.3}$$

■

From the definition of optimal allocations, each flow must fully acquire its capacity share on every path. Disregarding the packet reordering penalties, the following theorem establishes the optimality condition.

**Theorem 1** *A flow allocation is optimal, if and only if, the flow fraction assigned to each path is equal to its relative capacity contribution.*

*Proof.* Under optimal allocation, the flow acquires its full share on every path, hence according to lemma 1

$$\begin{aligned} \sum_{p_j} R_j &= \sum_{p_j} c_j = c_i / \phi_i \\ \text{or,} & \\ \phi_i &= c_i / \sum_{p_j} c_j, \quad \forall p_i \in P \end{aligned} \tag{2.4}$$

■

**Corollary 1** *A traffic allocation strategy is fair and efficient if it results in optimal allocations for all network flows.*

*Proof.* From theorem 1, given optimal allocation parameters, each flow acquires exactly its fair share along every candidate path. As we define the fair share along a path to be the throughput of a TCP flow forwarded entirely over that path, the corollary follows from the convergence properties of TCP congestion control (section 2.1).

■

Let  $R_i = \phi_i R$  be the maximum flow throughput in steady state on path  $p_i \in P$ , from (2.3), the relative forfeited capacity share by the flow on  $p_i$  due to congestion on some other path(s) is given by

$$\frac{c_i - R_i}{c_i} = 1 - \frac{[c/\phi]^*}{c_i/\phi_i} \quad (2.5)$$

where  $[c/\phi]^* = \min\{\frac{c_i}{\phi_i} | p_i \in P\}$ .

The ongoing characterization of optimal allocation parameters calls for adaptive traffic allocation. In case of traffic allocation using static allocation parameters (as in ECMP), the throughput of a flow is determined by how far, perturbations in traffic load, drives the ratio  $[c/\phi]^*$  away from the sum of the available capacity on all paths. Next, we validate the optimality condition and demonstrate this necessity of adaptive traffic allocation.

### Experiment ECMP vs OPTIMAL

Figure 2.1 shows the topology used to conduct the simulation experiments.

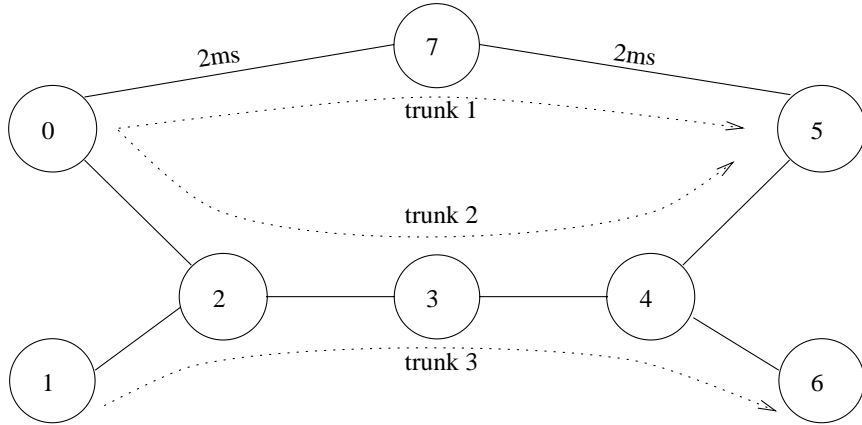


Figure 2.1: Experiment ECMP vs OPTIMAL. Simulated topology.

All links are 20 Mb/s with 1ms propagation delay, except as noted. FEC 1 is composed of 20 FTP flows that originate at node 0 and terminate at 5. Its packets are forwarded along trunks 1 and 2. FEC 2 flows are forwarded along trunk 3. ACK packets are forwarded along the reverse direction of the same trunks using ECMP

FEC 2 Volume ( $n_2$ )	0	10	20	30	40	50	60	70	80
$\phi_1^1$	0.5	0.6	0.6667	0.7143	0.75	0.7778	0.8	0.8182	0.8333
$\phi_2^1$	0.5	0.4	0.3333	0.2857	0.25	0.2222	0.2	0.1818	0.1667

Table 2.1: Experiment ECMP vs OPTIMAL. FEC 1 Traffic allocation parameters under OPTIMAL

parameters. Each FTP flow is controlled by TCP NewReno with a receiver window larger than the sum of the bandwidth-delay products on all trunks. All connections transmit 1000 byte packets. Links are full-duplex and have RED output queues with default parameter values and a buffer size of 100 packets.

We compare the analytical model with the simulation results for three strategies: ECMP, OPTIMAL and OPTIMAL/R. For each allocation strategy, we vary the number of flows in FEC 2 from 0 to 80 flows and run the simulator with the corresponding traffic allocation parameters. ECMP splits FEC 1 packets equally among trunks 1 and 2 while OPTIMAL and OPTIMAL/R apportion FEC 1 traffic according to the optimal traffic allocation parameters in (2.4). In OPTIMAL/R resequencing of FEC 1 packets is performed at the egress (router 5) before their delivery to the TCP receiver. Since both trunks 2 and 3 share the same path, the fair capacity share along each is  $\frac{20}{n_1+n_2}$  Mb/s, where  $n_1 = 20$  and  $n_2$  are the volumes of FEC 1 and FEC 2 respectively. The fair share along trunk 1 is  $\frac{20}{n_1}$  Mb/s. The resulting optimal parameters for FEC 1 are listed in table (2.1) where  $\phi_i^1$  denote the allocation of FEC 1 to trunk  $i$ .

From lemma 1, the flow throughput for FEC 1 under ECMP is at most  $\min \left\{ \frac{1}{1/2}, \frac{20/(n_1+n_2)}{1/2} \right\} = \frac{40}{n_1+n_2}$  Mb/s. Similarly, disregarding the reordering penalties the fair flow throughput under OPTIMAL should be  $1 + \frac{20}{n_1+n_2}$  Mb/s. The plot in figure 2.2 compares these values to those obtained through simulation.

The analytical model does not capture the effect of packet reordering evident in the difference between the analytically calculated throughput and that obtained experimentally through OPTIMAL. In OPTIMAL/R, the effect of packet reordering is eliminated and FEC 1 flows achieve their optimal throughput thus validating the optimality conditions.

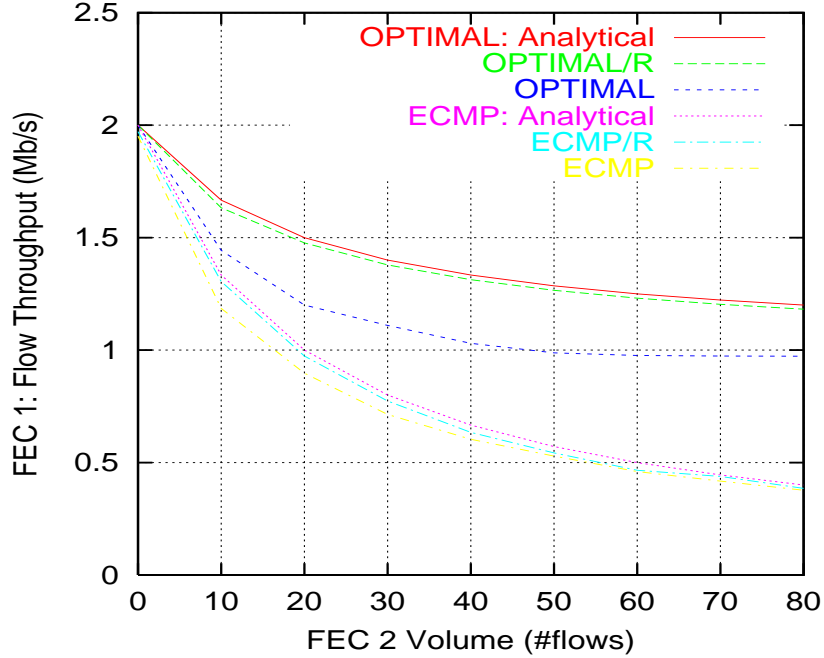


Figure 2.2: Experiment ECMP vs OPTIMAL. FEC 1 flow throughput obtained through simulation compared to that obtained analytically for ECMP and OPTIMAL.

The results also validate the bound in lemma 1 and demonstrate the necessity of adaptive traffic allocation in multipath networks. As congestion escalates on trunk 2, OPTIMAL and OPTIMAL/R forward FEC 1 packets increasingly on trunk 1. In the limit, all FEC 1 packet will exclusively follow trunk 1, efficiently utilizing its capacity; whereas under ECMP, the throughput of FEC 1 flows will decrease to zero leading to an inefficient equilibrium where trunk 1 is not well utilized.

According to (2.4), optimal traffic allocation requires the estimation of the TCP fair capacity share along every path. We attempt modeling the TCP equilibria in the next section.

## 2.3 TCP Fair Share Estimation

In multipath settings, estimation of TCP fair share along a path is not a trivial task. Direct measurements, for example using probe flows, may result in synchronized

measurements leading to oscillations. Many flows suffering congestion on alternate paths may simultaneously move a large fraction of load to the light-loaded links. The estimation of fair shares requires knowledge of the number of flows along each link and a mathematical model of TCP fairness.

An informal characterization of TCP fairness in single path networks is, that flows competing for capacity in a bottleneck link get equal shares if they have identical round-trip times, packet sizes, and suffer from equal path loss rates. In this section, we provide an approximate characterization of TCP fairness in terms of weighted max-min fair allocations and demonstrate its validity.

### 2.3.1 Modeling TCP Fairness as Weighted Max-Min

Consider a set of flows sharing a single bottleneck link. According to the deterministic model of TCP throughput [18], at steady state the sending rate of a TCP connection  $i$  is given by

$$\lambda_i [\text{bytes/sec}] = \frac{B_i}{RTT_i} \sqrt{\frac{3}{2l_i}} \quad (2.6)$$

where  $B_i$ ,  $RTT_i$ , and  $l_i$  are the connection's packet size in bytes, round-trip time, and loss rate respectively. We assume a single bottleneck where all packet losses occur and later relax this condition. For simplicity, we also assume that the RTT is insensitive to fluctuations in the number of flows and that the loss events witnessed by each flow follow IID random processes (e.g., Poisson) with rate  $l_i = l$ . These assumptions are not far from reality. Even a reduction in the number of flows across the bottleneck would not reduce the queueing delay because the remaining flows keep probing for more bandwidth. Moreover, the IID distribution of loss events is realistic in networks deploying randomized packet discard mechanisms such as RED. Given these assumptions, flow throughput is determined by its packet size and RTT.

A change in the number of flows crossing the bottleneck results in a change in the loss event rate  $\Delta l$ . Let  $\sigma_i$  be the throughput of connection  $i$  after the change, then the throughput of all connections increase (decrease) by a constant factor  $\alpha$ , that is

$$\alpha = \frac{\sigma_i}{\lambda_i} = \sqrt{\frac{l + \Delta l}{l}} \quad (2.7)$$

An increase in the number of flows keeps the link congested. Equally, a decrease therein results in higher throughput for the remaining flows, thus maintaining the state of the link as a bottleneck. Let  $C$  denote the bottleneck capacity, then  $\alpha \sum_i \lambda_i = C$  or,

$$\alpha = \frac{C}{\sum_i \lambda_i}$$

and substituting in (2.7),

$$\sigma_i = C \frac{\lambda_i}{\sum_j \lambda_j} \quad (2.8)$$

Given the throughput of TCP flows through the bottleneck before the change and given the new flows are known to have the same end-to-end path characteristics as some existing flows, Each flow's new share of bottleneck capacity can be predicted using (2.8).

If paths with multiple bottlenecks exist, a subset of the flows may not be able to acquire their fair share in full, in which case, the unclaimed shares are divided among the unsaturated flows. The problem is therefore one of computing the weighted max-min fair allocations where the weight vector is  $\lambda$  and the allocation vector corresponds to  $\sigma$ . This model however does not capture the compounded effect of multiple path bottlenecks on TCP throughput: losses occur at each bottleneck, reducing the throughput below the fair share on the most congested link. For example, if loss events at each bottleneck along a path follow a Poisson process, the rate of the overall loss process witnessed by a flow along that path is the sum of the loss event rate at each bottleneck. Accordingly, from (2.6), the throughput of such flow must be less than the share on the most congested bottleneck. Bias against flows passing through multiple bottlenecks is an undesirable property of TCP congestion control. Assuming no significant change in traffic volume occurs over a certain interval of time, rate policing can be applied to enforce the globally fair allocations and rectify the bias without trading-off network utilization.

### Experiment fairness model validation

To validate the fairness model in case of a single bottleneck per path, we performed two sets of  $ns$  simulation experiments using the topology shown in figure 2.3.

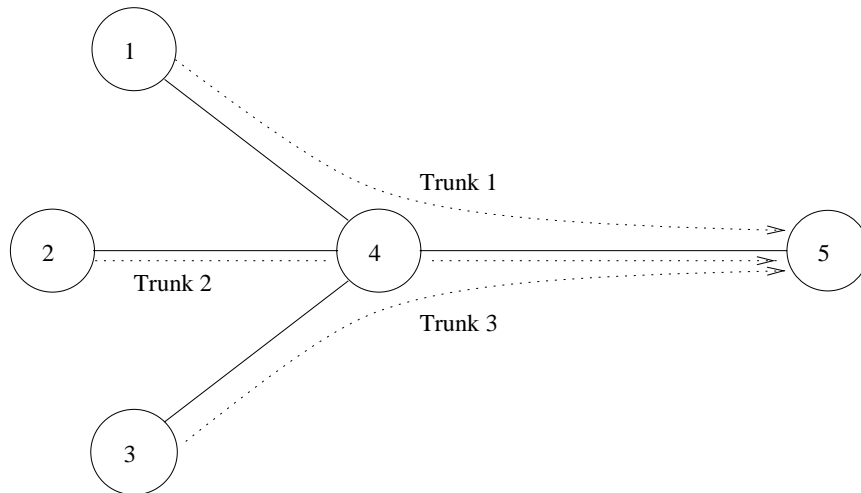


Figure 2.3: Experiment FAIRNESS MODEL VALIDATION. Simulated topology.

All links are 10Mb/s - 1ms except for link 1-4. Traffic consists of 20 FTP connections from each of sources 1 and 2 to node 5 (called trunks 1 and 2 respectively). In both experiments the number of flows from node 3 to node 5 (trunk 3) is varied from 10 to 40. All connections transmit 1000 bytes packets and are based on TCP NewReno with the receiver-advertised window being much larger than the bandwidth-delay product of the network. Output interfaces have RED queues with the default *ns* parameters.

The objective of the experiments is to compare per-flow throughput obtained from equation (2.8) against the measured throughput. The weight vector is taken to be the measured throughput along each trunk in the case traffic from 3 to 5 consists of 20 flows. In the first set of experiments, the propagation delay of link 1-4 is 1ms. Consequently, all flows have equal round-trip times. In the second set, the propagation delay of link 1-4 is set to 25ms to verify that the model captures the bias of TCP against flows with larger RTT. The resulting throughput plots are shown in figures 2.4 and 2.5.

Flow throughput of trunks 1 and 2 is plotted against the number of flows in trunk 3. In figure 2.4, the RTT is the same for all flows. In figure 2.5, flows in trunk 1 have approximately twice the RTT of trunks 2 and 3. The plots suggest that the weighted

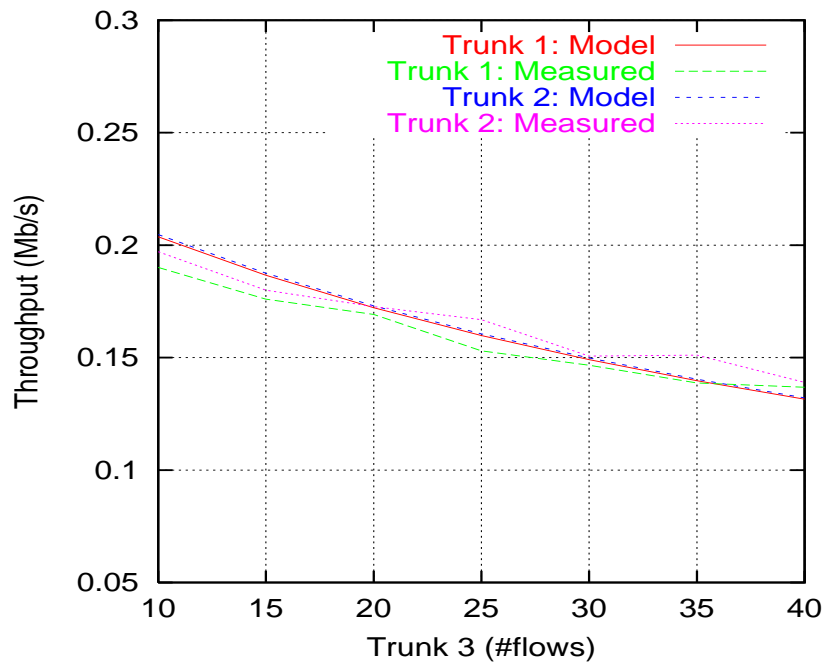


Figure 2.4: Experiment FAIRNESS MODEL VALIDATION. Case 1: Connections with equal round-trip time.

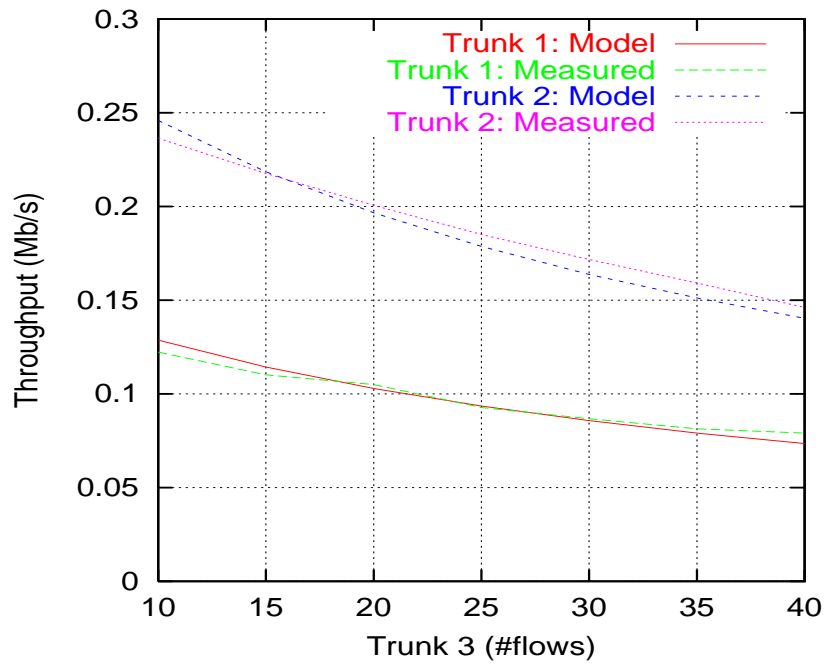


Figure 2.5: Experiment FAIRNESS MODEL VALIDATION. Case 2: Trunk 1 connections have twice the RTT of trunk 2.

max-min fair model yields accurate predictions of the capacity shares in case of a single bottleneck. In all cases, the measured throughput slowly oscillates around the values obtained from the model.

### 2.3.2 Estimation of TCP Fair Shares in Multipath Networks

We now discuss the application of the fairness model to multipath networks. Up-to-date measurements reflect the actual number of concurrent flows in each FEC and hence the number of flows passing through each link, as well as, trunk weights  $\lambda$ . The problem, thus, is not one of predicting flow throughput given a change in the number of competing flows, nevertheless, synchronized trunk weight measurements may result in overestimating the capacity available to some competing trunks along shared links, hence resulting in a similar effect. Traffic allocation based on inflated measurements causes the affected links to become overloaded, consequently increasing the packet drop rate and causing the the actual shares to deviate from the measured ones. Equation (2.7) describes the proportionality relation between the measured weight and actual flow shares through a bottleneck given a change in loss rate. The actual trunk shares  $\sigma$  can thus be predicted using the weighted max-min fair allocation model by simply substituting the corresponding trunk weights into equation (2.8).

## 2.4 Summary

In this chapter, we presented a constructive characterization of fair and efficient traffic allocation that maximize the throughput of TCP connections by optimally splitting each flow along multiple paths. The optimality condition is defined in terms of the TCP fair share along each path. We also introduced and validated a method for the accurate estimation of TCP fair share based on path measurements and a weighted max-min allocation model of TCP.

Trunk share estimation and the optimality condition are the basis for AMTA—a centralized solution for adaptive multipath traffic allocation that we introduce the next chapter. For the optimality condition to hold in practice, AMTA must ensure

packet reordering penalties are rectified.

## Chapter 3

# AMTA: Adaptive Multipath Traffic Allocation

AMTA is a centralized service that performs adaptive multipath traffic allocation within a routing domain based on the optimality and fairness results of chapter 2. The main task of a traffic allocation service is the estimation of the fair share along network paths and hence the computation of traffic allocation parameters. We describe AMTA in the context of a network environment that provides support functions such as routing, packet classification, resequencing and, traffic measurements and policing. At first, we limit the discussion to the case where traffic sources and sinks are connected to the edges of the local domain and have large bandwidth demand, then extend AMTA to deal with flows that have limited window size.

### 3.1 The Network Environment

#### 3.1.1 Traffic Aggregation and Routing

AMTA is designed for best-effort networks with explicit routing support (for example, using MPLS). Routing and traffic allocation at the user-flow level is not scalable, therefore, through packet classification, traffic with common ingress and egress routers is logically grouped into a Forwarding Equivalence Class (FEC). The volume of a

FEC is the number of concurrently active flows belonging to that FEC. A multipath routing service provides, for each FEC, a set of paths connecting the corresponding ingress-egress pair. Although the paths for a particular FEC may not be disjoint, in case a common link suffers from increased competition. A constraint on routing is the small delay skew among the multiple paths connecting any two routers. Efficient packet resequencing at the egress routers requires bounded small skew. The routing service may revise routes periodically, or in response to link failures or topology changes. The route service provides the traffic allocation server with a copy of the route database, with trunk identifiers matching those used by the ingress routers. The role of a traffic allocation algorithm is the computation of the traffic allocation parameters for each FEC.

### 3.1.2 Packet Classification and Traffic Measurements

Packet classifiers [12] at the ingress routers map each packet to the corresponding FEC for path selection. Packet classification is also used to maintain an estimate of the average number of concurrently active flows in each FEC, to track the average packet size and to enforce the computed allocations. In addition, ingress nodes periodically measure or estimate the trunk shares, that is, the throughput of a TCP flow routed exclusively over each trunk, also called the bulk transfer capacity (BTC) of the path. Upon request, ingress routers communicate the number of flows and trunk shares of each FEC to the AMTA server.

AMTA does not depend on the particular technique used for measuring the FEC volumes<sup>1</sup> or trunk shares as long as it is provided with accurate estimates. For trunk shares, probe flows are a good candidate since the interval between measurements is in the order of minutes and TCP probes reach steady state in a short time. The bandwidth consumed by probe flow becomes a substantial overhead only when the number of user-flows sharing the links is small. It is worth noting that there are many tools for measuring BTC [2], as well as an ongoing standardization effort within the

---

<sup>1</sup>Off-the-shelf flow measurement tools that perform similar functions are available, see [22] for an example.

IETF [16].

### 3.1.3 Rate Policing, Path Selection, and Resequencing

In section 3.3, we demonstrate the necessity of rate policing in order to protect the flows against the inherent TCP bias against flows passing through multiple bottlenecks. To maintain scalability, ingress routers perform policing at the FEC level. It should be noted that policing induced packet drops must be distributed uniformly across the flows of the misbehaving FEC. A candidate scheme is to associate with each FEC at the ingress a variant of token bucket with packet buffers that emulate a RED link with negligible propagation delay. The rate of token generation is set at the beginning of each epoch, to the FEC throughput limit calculated by the AMTA server.

Ingress routers perform trunk selection such that the correct fraction of each FEC's packets is forwarded on each trunk according to the optimal parameters. This is achieved by selecting a path at packet arrival, using Surplus-Round Robin (SRR). Deficit round-robin offers slightly better accuracy in implementing the traffic allocations however, it is not a causal fair queuing mechanism hence does not allow effective packet resequencing at the egress [1, 21].

At the egress, out-of-order packets are kept in FIFO queues (called the resequencing queues) associated with FEC trunks. Resequencing of FEC packets is achieved by simulating the trunk selection function of the ingress to determine on which trunk to expect the next in-order packet. SRR is causal since its trunk selection function is independent from the size of the packet being assigned. As long as the selected trunk has positive credit stored in a deficit counter, it receives the packet even if its size is larger than the credit. A round ends when the deficit counters of all trunks are non-positive. For the new round, all trunks receive credits proportional to their traffic allocations from which the deficit is deducted. The resequencing algorithm operates correctly as long as there is no loss or replication of packets within the network. Packet replication is a rare event, whereas loss at the internal routers can be prevented through rate policing at the ingress. The traffic through a link is nor-

mally the superposition of traffic from multiple FECs. Simultaneous arrivals from many well-behaving FECs on a certain link may overflow its buffer and induce losses. However, since the the number of FECs is naturally small, simultaneous arrivals pose no actual problem. Another source of losses is the use of unpoliced probe flows for BTC measurements. Since BTC measurements are brief and infrequent, a simple solution is to have the ingress nodes of all affected trunks send marker packets after the end of each measurement session. An ingress ending a BTC measurement session informs a central service which looks-up the affected trunks in a routes database and in turn informs their ingress routers to resynchronize with the egress nodes. More sophisticated schemes are beyond the scope of this thesis.

The benefits of resequencing come at a cost. The resequencing function incurs buffering overhead, inefficient utilization of the switch fabric and the outgoing links due to the non-work conserving nature of the egress. To reduce this cost, resequencing queues can be implemented to share the same memory with the virtual output queues at the input of the egress router and using pointers to avoid the expensive packet copying operation. On the other hand, small delay skew among FEC trunks helps avoid packet buildup in resequencing queues and ensure efficient utilization of the switch fabric and the outgoing links.

## 3.2 The Trunk Share Estimation Algorithm

Measurements in [24], indicate that the instantaneous number of concurrent user-flows on Internet links does not deviate significantly from its mean over 5-minute periods. This is necessarily due to little variance in traffic between ingress-egress pairs. In AMTA, traffic measurements are averaged over 2.5 minute intervals called *epochs* and the averages are used to compute the routing parameters for the subsequent epoch.

At each invocation, the AMTA server collects the measured trunk shares and FEC volumes from the ingress routers and computes the bandwidth allocations using to the weighted max-min fair model of TCP presented in section 2.3. The resulting WMM fair allocation vector represents estimates the trunk shares in the subsequent

interval. The traffic allocation parameters for each FEC are then computed according to equation 2.4, and downloaded to the corresponding ingress routers together with the computed maximum FEC throughput which is used for rate policing as explained in section 3.3.

Assume a set of active FECs  $F$  and a set of trunks  $T$ . Let  $n_f$  denote the set of flows comprising a FEC  $f \in F$  and  $T_f \subseteq T$ , the set of trunks along which  $f$  is routed. Suppose the weight vector is  $\lambda$  with an element for each trunk  $t \in T$  representing the measured trunk share, per-flow throughput  $\sigma$  for each trunk is computed using the WMM fair allocation algorithm in figure 3.1 which is an adaptation of the one presented in [5, p.527]. In the algorithm  $A^k$  denotes the set of network links not saturated at the beginning of iteration  $k$ , and  $T^k$  denotes the set of trunks (across all FECs) not passing through any saturated link at the beginning of iteration  $k$ . Also,  $C_a$  denotes the nominal capacity of link  $a$  while  $U_a^k$  denotes the allocated capacity on link  $a$  at the end of the  $k^{\text{th}}$  iteration. Each iteration begins by calculating the maximum rate increment per unit weight  $r^k$ . In step 3, the share of each unsaturated trunk is incremented in proportion to its weight. Step 4 updates the capacity allocated on each link in preparation for the following iteration.

According to theorem 1 and given the accuracy of the allocation vector  $\sigma$ , the optimal throughput of a FEC  $f$  is  $n_f \cdot \sum_{t \in T_f} \sigma_t$  and the corresponding traffic allocation parameters are given by

$$\phi_\tau^f = \frac{\sigma_\tau}{\sum_{t \in T_f} \sigma_t} \quad \forall \tau \in T_f \quad (3.1)$$

The complexity of computing the traffic allocation parameters is determined by that of the trunk share estimation algorithm which can be shown to have  $O(MH^2)$  worst-case time complexity, where  $M = \max\{|T|, |A|\}$  and  $H = \min\{|T|, |A|\}$ . Naturally, one do not expect the routing service to provide a large number of paths for each FEC. Nonetheless, in case the number of trunks or the network size is large, the accuracy of allocation vector can be traded off to alleviate the time complexity. Given  $B$  as the maximum link bandwidth, Awerbuch and Shavitt [4] provide an  $O(\log B)$  quiescent time algorithm to approximate the max-min vector using a constant rate increment in each iteration.

**Algorithm** Trunk Share Estimation

Initial conditions:  $k = 1$ ,  $U_a^0 = 0$ ,  $\sigma_t^0 = 0$ ,  $T^1 = T$ ,  $A^1 = A$ .

1.  $T_a^k := \{t \mid t \in T \text{ and } t \text{ crosses link } a\}$
2.  $r^k := \min_{a \in A^k} (C_a - U_a^{k-1}) / \sum_{t \in T_a^k} n_t \lambda_t$
3.  $\sigma_t^k := \begin{cases} \sigma_t^{k-1} + \lambda_t r^k & \text{for } t \in T^k \\ \sigma_t^{k-1} & \text{otherwise} \end{cases}$
4.  $U_a^k := \sum_{t \in T_a^k} n_t \sigma_t^k$
5.  $A^{k+1} := \{a \mid C_a - U_a^k > 0\}$
6.  $T^{k+1} := \{t \mid t \in T \text{ and } t \text{ does not cross any link } a \notin A^{k+1}\}$
7.  $k := k + 1$
8. If  $T^k$  is empty, then stop; else goto 1.

Figure 3.1: Algorithm for Trunk Share Estimation

Next we use simulation to demonstrate that AMTA results in the optimal traffic allocation parameters.

**Experiment AMTA vs OPTIMAL**

AMTA vs OPTIMAL is a rerun of the experiment ECMP vs OPTIMAL (section 2.2) except for applying AMTA instead of ECMP.

The topology used for the experiment is the same as the one used for ECMP-OPTIMAL and is reproduced in figure 3.2 for convenience. We again vary the volume of FEC 2 from 0 to 80 flows and measure the throughput achieved by AMTA and OPTIMAL. The resulting throughput plots in figure 3.3 for AMTA and OPTIMAL coincide indicating that AMTA computes optimal traffic allocation parameters. Both strategies result in lower throughput for FEC 1 flows than the one obtained analytically because of the reordering penalty. For all volumes of FEC 2, its flows exceed their intended shares, resulting in efficient yet unfair equilibria on subpath

2-3-4. Recall that we define the fair share along a path as the throughput of a TCP flow entirely routed over that path.

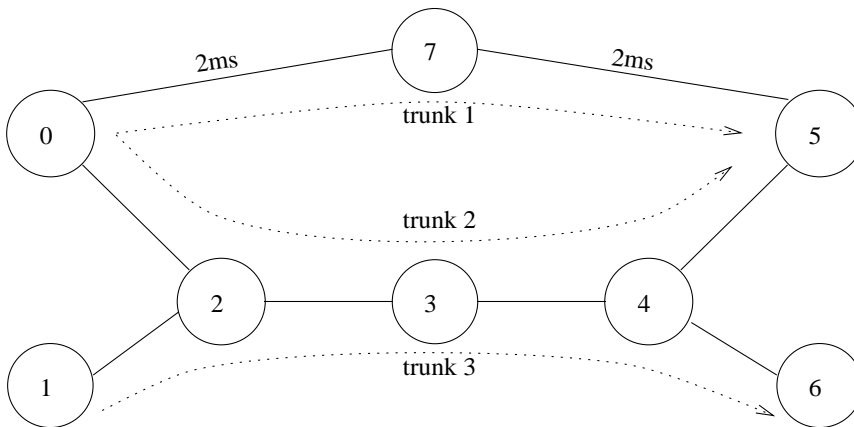


Figure 3.2: Experiment AMTA vs OPTIMAL. Simulated topology.

Table 3.1 lists the FEC 1 traffic allocation parameters obtained through AMTA at some FEC 2 volumes. Note that the parameters are identical to those obtained through OPTIMAL and listed in table 2.1. For each value of FEC 2 volume, simulation is run for two AMTA epochs. In the first epoch the traffic allocation parameter for FEC 1 trunks are set to 0.5. During that epoch probe flows are used along each trunk to measure the elements of weight vector  $\lambda$ . The resulting vector from each experiment is shown in table 3.2. At the beginning of the following epoch, the AMTA trunk share estimation algorithm (fig. 3.1) yields the trunk allocations shown in table 3.3. Given the trunk share vector  $\sigma$ , the allocation parameters are calculated from equation (3.1). Note that weight measurements renders AMTA independent from the initial allocation parameters. For large volumes of FEC 2, these measurements indicate that trunk 1 is underutilized during the first epoch, leading to a shift in traffic during the following epoch.

### 3.3 The Effectiveness of AMTA

Given the traffic allocation parameters, FEC flows will effectively acquire the computed fair share along each trunk under the following conditions:

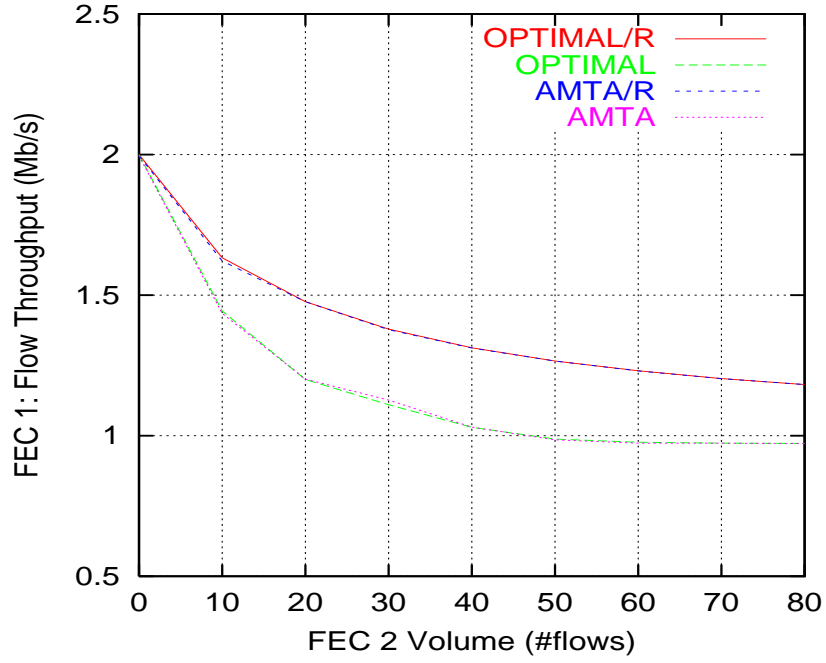


Figure 3.3: Experiment AMTA vs OPTIMAL. FEC 1 flow throughput.

FEC 2 volume (#flows)	0	20	40	60	80
$\phi_1^1$	0.4998	0.6658	0.747	0.802	0.8347
$\phi_2^1$	0.5002	0.3342	0.253	0.198	0.1653

Table 3.1: Experiment AMTA-OPTIMAL. FEC 1 traffic allocation parameters obtained through AMTA for different volumes of FEC 2.

FEC 2 volume	0	20	40	60	80
$\lambda_1$ (Mb/s)	2.564	11.401	14.1048	15.487	16.1
$\lambda_2$ (Mb/s)	2.521	0.554	0.3573	0.2523	0.198
$\lambda_3$ (Mb/s)	-	0.5503	0.348	0.2563	0.2007

Table 3.2: Experiment AMTA-OPTIMAL. Trunk weights for different volumes of FEC 2.

FEC 2 volume	0	20	40	60	80
$\sigma_1$ (Mb/s)	1	1	1	1	1
$\sigma_2$ (Mb/s)	1.0008	0.5019	0.339	0.2471	0.198
$\sigma_3$ (Mb/s)	-	0.4985	0.3306	0.25117	0.2

Table 3.3: Experiment AMTA-OPTIMAL. Trunk shares for different volumes of FEC 2.

1. The accuracy of the measured trunk share vector  $\lambda$ ,
2. the accuracy of FEC volume measurements,
3. the quasi-stationarity of FEC volumes,
4. the insensitivity of TCP flow control to packet reordering, and
5. the elimination of bias against flows with multiple bottlenecks.

In this study, we assume conditions 1 and 2 are trivially satisfied. In this section, we discuss solutions to cope with the rare yet possible sudden traffic swings and to eliminate the packet reordering penalty, as well as the bias against flows with multiple bottlenecks.

### 3.3.1 Coping with Traffic Swings

While condition 3 is covered by the findings in [24], rare events resulting in substantial traffic surges or ebbs do occur (e.g., traffic rerouting due to remote link failure or repair, or the end of a congestion period). The competing FECs may suffer severe throughput reduction since increased congestion on a path makes them unable to acquire their fair share on others for the remaining of the AMTA epoch. Therefore, AMTA must exhibit robustness in the face of sudden traffic changes by localizing the effects of traffic fluctuation to the affected FEC until the beginning of the following AMTA epoch. FEC isolation can be achieved through rate policing at the ingress points as described in 3.1.3. As we argue next, FEC isolation also promotes the convergence to fair equilibria in the lack of conditions 4 and 5.

### 3.3.2 Elimination of TCP Bias Against Flows with Multiple Bottlenecks

The very process of traffic reallocation may create multiple bottlenecks along the path of a trunk that had not existed while the BTC measurement were taking place. The compound effect of these bottlenecks is not reflected in the trunk weight measured

during the previous epoch. Unless the competing FECs on the bottlenecks are limited to their fair shares, the TCP bias described in section 2.3 will allow them to overrun the bottlenecked FEC, which in turn will not be able to acquire its fair share along other trunks. In case all the trunks of a FEC have room to grow (due to reduction in competition, or competition with trunks with multiple bottlenecks) then they the bias will take place. In contrast, if the traffic allocation parameters for a FEC are optimal, all trunks are bottlenecked hence limiting each other to their respective fair shares and the bias is eliminated. Rate policing at the FEC level ensures the elimination of the bias in any case. The following simulation experiment demonstrates the role of rate policing in eliminating TCP bias against flows with multiple bottlenecks.

### Experiment multiple bottlenecks

The topology for the experiment is shown in figure 3.4. All links are 20 Mb/s with 1ms propagation delay. Output queues are RED with the default parameters except for random drop instead of drop tail. All FECs consist of 20 FTP flows over TCP NewReno. FEC 1 packets are forwarded along trunks 1 and 2; FEC 2 along trunks 3 and 4; and FEC 3 along trunks 5 and 6.

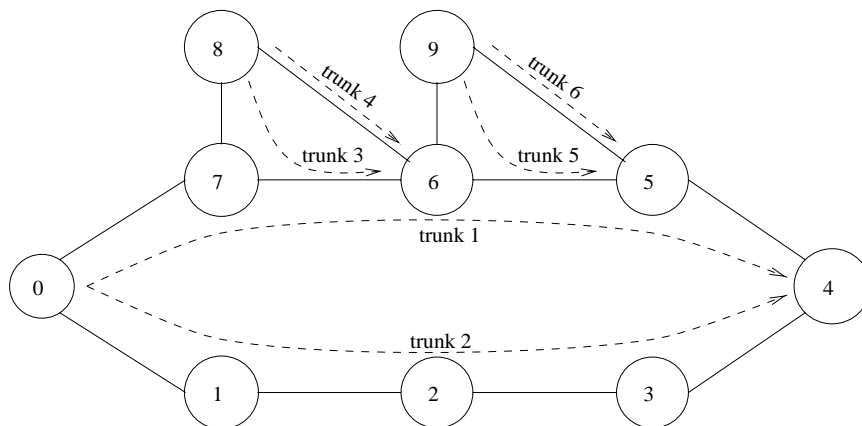


Figure 3.4: Experiment MULTIPLE BOTTLENECKS. Network topology.

Initially the network is at an inefficient equilibrium where FEC 1 is entirely routed through trunk 2, FEC 2 through trunk 4 and, FEC 3 through trunk 6; therefore

$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_5$	$\lambda_6$
20	0.935	20	0.934	20	0.933

Table 3.4: Experiment MULTIPLE BOTTLENECKS. Trunk weights in Mb/s.

$\sigma_1$	$\sigma_2$	$\sigma_3$	$\sigma_4$	$\sigma_5$	$\sigma_6$
0.5	0.935	0.5	0.934	0.5	0.933

Table 3.5: Experiment MULTIPLE BOTTLENECKS. Trunk shares in Mb/s.

leaving trunks 1, 3 and 5 unutilized. During the initial AMTA epoch, all trunks are probed to construct the weight vector  $\lambda$ . In our experiment we did not allow the probe flows to interfere, resulting in the weight vector listed in table 3.4. The trunks shares and traffic allocations parameters are shown in tables 3.5 and 3.6 respectively.

According to the revised allocation, trunk 1 passes through two bottleneck links, namely 7-6 and 6-5. Row 1 of table 3.7 shows the throughput of all FECs in case FECs 3 and 5 are limited to their fair shares by trunks 4 and 6 respectively. In this case there is no significant bias against FEC 1 flows. Next, we emulate a sudden reduction in competition along trunks 5 and 6 by doubling the capacity of links 8-6 and 9-5 to 40 Mb/s at the beginning of the second epoch. Row 2 of the same table lists the resulting throughput for all FECs without the deployment of rate policing at the ingress. The results indicate that FEC flows are severely affected by the bias as they are not able to acquire their fair shares on either trunks 1 or 2. In contrast, the deployment of rate policing effectively eliminates the bias as indicated by the results in row 3. We emulate the effect of policing by inserting a “virtual” RED link with negligible propagation delay between the FTP sources and the ingress routers

FEC 1		FEC 2		FEC 3	
$\phi_1^1$	$\phi_2^1$	$\phi_3^2$	$\phi_4^2$	$\phi_5^3$	$\phi_6^3$
0.348	0.652	0.349	0.651	0.349	0.651

Table 3.6: Experiment MULTIPLE BOTTLENECKS. Traffic allocation parameters.

$C_{8 \rightarrow 6}$ and $C_{9 \rightarrow 5}$ (Mb/s)	FEC 1 (Mb/s)	FEC 2(Mb/s)	FEC 3(Mb/s)
20	1.332	1.534	1.534
40	0.8163	2.07	2.076
40 (w/ Rate policing)	1.4565	1.4115	1.418

Table 3.7: Experiment MULTIPLE BOTTLENECKS. Flow throughput results.

8 and 9. The virtual link capacity (i.e., the maximum permitted rate) for FEC 2 is  $20(\sigma_3 + \sigma_4) = 28.7$  Mb/s and the same for FEC 3. Table shows that under rate policing all FECs acquire their fair shares on all trunks.

### 3.4 Handling Flows with Limited Throughput Demand

In this section we relax our assumptions about flow characteristics. Specifically we extend AMTA to handle flows with small transmission window due to congestion on their routes through other domains as well as those with modest throughput demand due to application characteristics, small receiver window or in the near future TCP implementations without the D-SACK extension.

In every traffic allocation epoch, ingress routers are responsible for monitoring the number of active flows for each of their FECs and communicating it to the traffic allocation server which in turn computes the FEC throughput and the optimal allocation parameters and returns them to the ingress router for use during the following epoch. Under the relaxed assumptions, some of the FEC flows will not acquire their fair shares, while competing flows in other FECs are limited to their fair shares thus resulting to inefficient or unfair equilibrium. Consequently, it is necessary to estimate the effective number of flows based on the FEC throughput demand in the previous epochs. For instant consider a FEC  $f$ , routed along a set of trunks  $T_f$  with  $\sigma_t$  being the trunk share for all  $t \in T_f$  computed at the end of previous epoch. Let  $R_f$  be the measured throughput of FEC  $f$  during the current epoch, the effective number

**Algorithm** FEC Volume Estimation

1.  $\overline{n}_f^{prev}$  := effective number of FEC flows during the previous epoch
2.  $n_f$  := number of active flows in the current epoch
3.  $R_f$  := FEC throughput in the current epoch
4.  $slack = 0.15$
5. if ( $R_f = \overline{n}_f^{prev} \cdot \sum_{t \in T_f} \sigma_t$ )
  6.  $\overline{n}_f := \frac{\overline{n}_f^{prev} + n_f}{2}$
- else
  7.  $\overline{n}_f := \frac{R_f}{\sum_{t \in T_f} \sigma_t} (1 + slack)$

Figure 3.5: Algorithm for the estimation of effective number of FEC flows

of flows during that epoch can be estimated using the following formula

$$\overline{n}_f = \frac{R_f}{\sum_{t \in T_f} \sigma_t}$$

which has  $\overline{n}_f^{prev}$  as an upper bound due to rate policing. That is, in case the throughput demand or the actual number of flows increases, rate policing will not allow the increase to be reflected in the effective number of flows. One way to get around this limitation is to overestimate the effective number of by a small percentage called the *slack*, if the the FEC traffic grows to use the slack, it is an indication that rate policing rather than external congestion or small client window is the limiting factor of the throughput of the FEC flows. In such case the estimate must be increased such that it converges rapidly on the correct value. The FEC volume estimation algorithm in figure 3.5 achieves this goal by converging on the correct number of flows in logarithmic number of epochs. Even if the number of flows in the FEC keeps increasing, its maximum increase rate (as was shown in [24]) is much slower than that of the convergence speed of the algorithm. The following experiments demonstrate

the effectiveness of the the volume estimation algorithm in allowing AMTA to achieve efficient equilibria where rate policing does not leave links underutilized.

### Experiment small window

In the topology of figure 3.6, FECs 1 and 2 consist of 20 FTP flows each based on New Reno TCP. FEC 1 packets are forwarded along trunks 1 and 2, while FEC 2 packets are forwarded along trunks 3 and 4. Once again all links are 20Mb/s and 1ms propagation delay. Output queues are RED with default parameters except for using random drop instead of drop tail. The capacity and propagation delay of the external link 0-1 is varied to emulate congestion on the end-to-end path of FEC 2 outside the local domain.

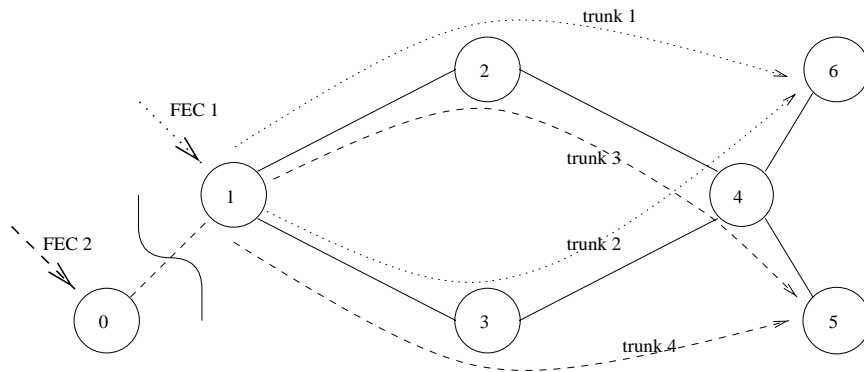


Figure 3.6: Experiment SMALL WINDOW. Simulated topology.

Initially each FEC's load is split equally among the corresponding trunks and the effective number of flows in each FEC is considered to be average number of concurrently active flow in that FEC. At the beginning of epoch 1 the capacity of link 0-1 is set to 4Mb/s and its propagation delay to 20ms. The original capacity and propagation delay are restored at the beginning of epoch 3. The plots in figures 3.7 and 3.8 show the reaction of AMTA to the changes in FEC 2 traffic load. At the end of epoch 1, AMTA reacts by reducing the effective number of flows for the FEC and hence its fair share. Note that by the beginning of FEC 3, the sum of the maximum allowed rates for both FEC amounts to an efficient equilibrium. FEC 1 however is not

able to fully utilize its share due to significant reordering penalty at higher throughput and AMTA slightly reduces its effective volume. At the beginning of epoch 4, AMTA realizes that FEC 2 throughput has grown to fill the slack and adjust the effective volume accordingly. By the beginning of epoch 6, AMTA had correctly estimated the volume of each FEC. We conclude that the FEC volume estimation algorithm responds correctly to traffic changes and allows AMTA to converge quickly to an efficient equilibrium by revising the throughput bound for the competing FECs.

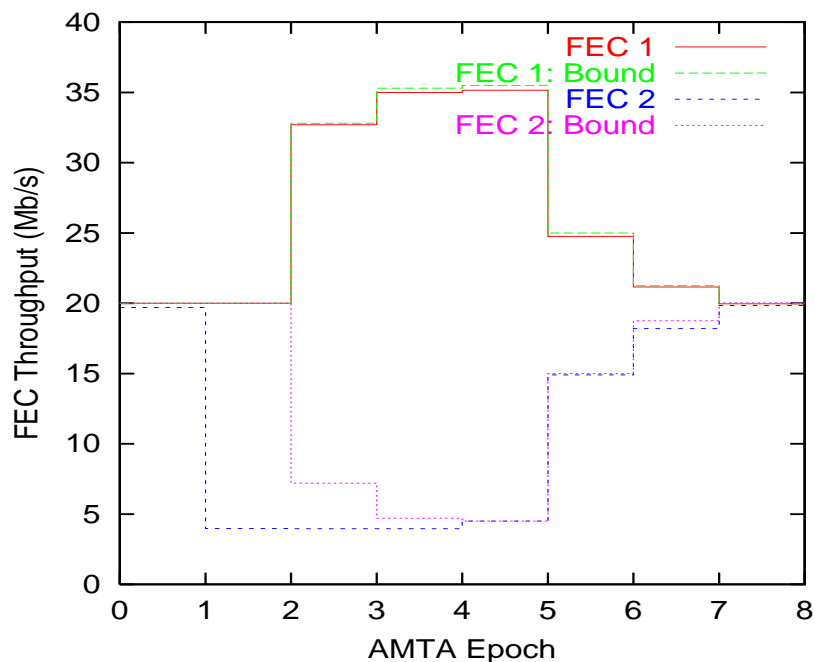


Figure 3.7: Experiment SMALL WINDOW. Overall throughput for each FEC and their respective rate bounds.

The value of the `slack` parameter should be set so as to prevent unnecessary fluctuations in the available rate for the competing FECs from one epoch to the next. Consider a FEC whose size increases according to the findings in [24], that is, at a maximum rate of 15% every epoch. If the value of the `slack` is less than 0.15, then the effective number of flows is computed from step 6 potentially overestimating the effective FEC volume one or more epochs. The overestimate is reflected in the bandwidth reserved for the FEC on each trunk and deducted from that reserved for the competing FECs. A too large `slack` on the other hand may result in inefficient

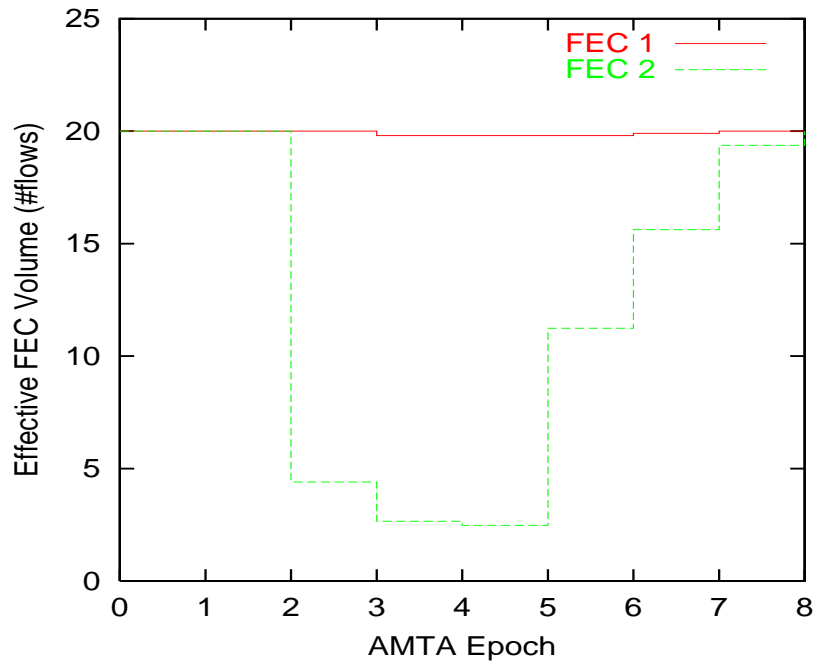


Figure 3.8: Experiment SMALL WINDOW. Effective volume for each FEC.

utilization of network links.

### 3.5 Summary

In this chapter we described AMTA, a centralized adaptive traffic allocation service for networks with explicit routing and other traffic engineering support functions. FEC and path measurements are performed during each epoch by the ingress routers, which, upon its request at the epoch's end, send them to the AMTA server. The trunk share estimation algorithm computes the fair share for each trunk, and hence the optimal traffic allocation parameters and the throughput limit for each FEC to be used during the following epoch. Under AMTA, and given the validity of the models in chapter 2, each flow should acquire its fair share along all candidate paths. However, packet reordering and TCP bias against connections passing through multiple congested routers were not captured in our models. We demonstrate that FEC isolation through rate policing can eliminate the TCP biases and show that FEC

packets resequencing at the domain borders with proper resynchronization through marker packets eliminates packet reordering. We finally extend AMTA to deal with flows having limited bandwidth demand by introducing an algorithm for estimating the effective number of flows in each FEC and demonstrate its effectiveness through simulation.

# Chapter 4

## Concluding Remarks

Multipath forwarding potentially improves the throughput of user flows and the utilization of network links. To implement multipath forwarding, two problems have to be addressed, namely, multipath routing and traffic allocation. Recognizing the prevalence of TCP-style traffic on the Internet, we consider the effect of traffic allocation on the throughput of TCP flows. We have shown in section that static traffic allocation based on long term load estimates can result in poor performance for TCP flows, concluding that traffic allocation should adapt to load fluctuation so that the throughput of TCP flows is maximized. To avoid the problems associated with adaptive routing such as loops and routes oscillation, we call for separating routing from traffic allocation as a departure from earlier effort that combined them into a single function.

Given a multipath network with a set of routes between ingress and egress nodes, our objective has been to devise an efficient method for computing the optimal allocation parameters for each FEC while maintaining the traffic allocation process transparent to the network internal routers and the transport protocol endpoints.

In chapter 2, we presented a constructive characterization of fair and efficient traffic allocation that maximizes the throughput of TCP connections by optimally splitting each flow along multiple paths. The optimality condition is defined in terms of the TCP fair share along each path. We also introduced and validated a method for the accurate estimation of TCP fair share based on path measurements and a

weighted max-min allocation model of TCP.

In chapter 3 we described AMTA, a centralized adaptive traffic allocation service for MPLS networks. FEC and path measurements are performed during each epoch by the ingress routers, which, upon its request at the epoch's end, send them to the AMTA server. The trunk share estimation algorithm computes the fair share for each trunk, and hence the optimal traffic allocation parameters and the throughput limit for each FEC to be used during the following epoch. Under AMTA, each flow should acquire its fair share along all candidate paths. To avoid the negative effects of packet reordering and the TCP bias against connections passing through multiple congested routers, we provide solutions to alleviate there effects—we demonstrated that FEC isolation through rate policing can eliminate the TCP biases and permits effective resequencing of FEC packets at the egress. We extended AMTA to deal with flows having limited bandwidth demand by introducing an algorithm for estimating the effective number of flows in each FEC and demonstrated its effectiveness through simulation.

Extending AMTA to DiffServ environments have not been considered in this thesis and is left for future work. Bandwidth reservation can be reflected in AMTA simply by modifying the link capacity available for best-effort flows. In case of service differentiation through preferential packet dropping, ingress-egress traffic should be split into a FEC for each service level.

The exact mechanisms for implementing support functions such as packet classification, rate policing have not been considered. A realization of AMTA may uncover some potential research problems. A testbed implementation of AMTA would increase its practical value and make its deployment likely as the major service providers begin to deploy MPLS.

# Bibliography

- [1] H. Adishesu, G. Parulkar, and G. Varghese. A reliable and scalable striping protocol. In *Proc. of the ACM SIGCOMM*, pages 131–141, 1996.
- [2] Mark Allman. Measuring end-to-end bulk transfer capacity. February 2001. Submitted for review. URL: <http://www.citeseer.nj.nec.com/418148.html>.
- [3] D. Awduche, A. Chiu, I. Elwalid, A. and Widjaja, and Xiao X. A framework for Internet traffic engineering. Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.
- [4] B. Awerbuch and Y. Shavitt. Converging to approximated max-min flow fairness in logarithmic time. In *Proc. of the 17th IEEE INFOCOM*, pages 1350–57, March 1998.
- [5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall Inc., 2nd edition, 1992.
- [6] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, pages 1–14, june 1989.
- [7] A. Elwalid, C. Jin, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, April 2001.
- [8] S. Floyd and K. Fall. Router mechanisms to support end-to-end congestion control. Technical report, LBL, February 1997.

- [9] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):459–472, 1999.
- [10] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM*, pages 43–56, 2000.
- [11] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.
- [12] Pankaj Gupta and Nick McKeown. Algorithms for packet classification. *IEEE Networks*, 15(2):24–32, March/April 2001.
- [13] E. Gustafsson and G. Karlsson. A literature survey on traffic dispersion. *IEEE Network*, 11(2):28–36, March/April 1997.
- [14] A. Khanna and J. Zinky. The revised ARPANET routing metric. In *Proceedings of ACM SIGCOMM*, pages 45–56, September 1989.
- [15] R. Mahajan and S. Floyd. Controlling high-bandwidth flows at the congested router. Technical report tr-01-001, ICSI, April 2001.
- [16] M. Mathis and M. Allman. A framework for defining empirical bulk transfer capacity metrics. Internet RFC 3148, Internet Engineering Task Force, July 2001.
- [17] J. Moy. OSPF version 2. RFC 2328, Internet Engineering Task Force, April 1998. Standards track.
- [18] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, pages 303–314, Vancouver, B.C., September 1998.
- [19] I. Rhee, V. Ozdemir, and Y. Yi. TEAR: TCP Emulation At Receivers - flow control for multimedia streaming. Technical report, NCSU, April 2000.

- [20] A. Shaikh, J. Rexford, and K. G. Shin. Load-sensitive routing of long-lived IP flows. In *Proceedings of ACM SIGCOMM*, pages 215–226, Cambridge, MA, August 1999.
- [21] M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. In *Proceedings of ACM SIGCOMM*, pages 231–242, 1995.
- [22] Cisco Systems. NetFlow FlowCollector documentation. URL: <http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nfc/>.
- [23] E. Crawley *et al.* A framework for QoS-based routing in the Internet. RFC 2386, Internet Engineering Task Force, August 1998.
- [24] K. Thompson, G. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, 1997.
- [25] C. Villamizar. OSPF optimized multipath. Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.
- [26] M. Vojnovic, J. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip time. In *IEEE INFOCOM*, March 2000.
- [27] S. Vutukury and J.J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *ACM SIGCOMM*, pages 227–237, Cambridge, Massachusetts, September 1999.
- [28] S. Vutukury and J.J. Garcia-Luna-Aceves. A traffic engineering approach based on minimum-delay routing. In *Proceedings of the Ninth International Conference on Computer Communications and Networks*, pages 42–47, Las Vegas, Nevada, October 2000.