An Integrated Framework for Management and Traffic Engineering of Best Effort Service in Next Generation Data Centric Networks

by

Christopher Marty

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2007

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted by the Graduate Faculty in Engineering in satisfaction of the requirements for the degree of Doctor of Philosophy.

_____ _____

Date **Prof. Mohamed A. Ali**
Chair of Examining Committee

_____ _____

Date **Dean Mumtaz K. Kassir**
Executive Officer

Prof. Samir Ahmed (Department of Electrical Engineering, The City College of New York, CUNY)

Prof. Tarek N. Saadawi (Department of Electrical Engineering, The City College of New York, CUNY)

Prof. M. Ümit Uyar (Department of Electrical Engineering, The City College of New York, CUNY)

Prof. Manish Parashar (Department of Electrical Engineering, Rutgers University College of Engineering, Piscataway, NJ)

Dr. Cheenu Srinivasan **(**Technology Research Department, Bloomberg L.P., New York, NY)

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

An Integrated Framework for Management and Traffic Engineering of Best Effort Service in Next Generation Data Centric Networks

by

Christopher Marty

Advisor: **Prof. Mohamed A. Ali**

This thesis addresses the previously unanswered question of how to handle best-effort traffic in Multiprotocol Label Switched (MPLS) based next generation data-centric networks. Specifically, this thesis examines the technical feasibility and assesses the performance of implementing an integrated framework for managing and traffic engineering best effort traffic in next generation data centric networks. Through proof and simulation, this thesis will show that the proposed integrated framework successfully enforces weighted fair rates for best-effort flows while simultaneously increasing

The implementation of the proposed system proceeded in two phases. In the initial phase, a system for signaling fair rate using unmodified RSVP messages is devised. Signaling overhead and legacy integration are analyzed. To calculate fair rates, a distributed WPMM fair rate algorithm was developed. The algorithm is tuned to work at low refresh rates and calculates weighted fair rates with O(1) computation complexity. An elegant proof based on bottleneck ordering is provided to show that the algorithm converges to fair rates in arbitrary network topologies.

Based on the results of phase one, a second phase was undertaken to introduce a new paradigm in queue management. The introduction of this new queue management was essential to ensure that the initial congestion control mechanism is compatible/friendly

with the native TCP congestion control mechanism, and will converge to fair rates with minimal signaling overhead. This new paradigm is based on Bettter than Best Effort (BBE) and Less than Best Effort (LBE) fair rate marking of packets and is shown to drive the network to fair rates with minimal overhead. This mechanism is shown to be vastly superior to the initial mechanism in both enforcing fair rates and in network throughput. We call this system TERM (TCP friendly Explicit Rate congestion control for MPLS networks).

Low frequency signaling coupled with efficient calculation of WPMM fair rates makes the TERM system a feasible solution for explicit rate congestion control in large scale networks. To our knowledge this is the first implementation of such a system.

ACKNOWLEDGEMENTS

I'd like to acknowledge the support of my advisor Mohamed A. Ali. His leadership, dedication to the art, and hard work made this thesis possible. His patience and guidance through the process have made him a mentor in the truest sense of the word.

I'd also like to thank Cheenu Srinivasan for the many hours he spent reviewing papers and discussing topics. His many comments and technical observations significantly enhanced this work. His constant encouragement helped me through many hard times.

So many people contributed to this work, it is impossible to name every one. My sincere thanks go out to everyone their technical contributions, support, and encouragement.

This Thesis is dedicated to all my Family. To my son Jonathan who spent countless hours sitting with me while I ran simulations and wrote papers. Watching him work through his first attempts at forming letters and doing basic math was an inspiration; nothing in this thesis compares to the outstanding work he did during those times. To my daughter Sarah, who doesn't remember a time when her father wasn't glued to his computer. The time she spent sitting on my lap, banging random characters into my documents were some of the most pleasurable parts of this long process. To my wife Rita who assumed responsibility above and beyond what was reasonable so that I could complete this work. Without her constant support and encouragement, this could not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 - Introduction

1.1 Introduction

We live in an Internet Protocol (IP) centric world. Twenty years ago few would have guessed, but IP has become a central fixture not only in our networking technology, but also in our culture. As the Internet pervades more and more aspects of our lives, most would agree that IP is here to stay.

IP has survived, and in fact thrived due to its ability to handle load gracefully, and it that regard, it measures up so favorably that it has displaced mostly all competing network protocols. IP though, is not without its flaws. IP scales exceptionally well to large networks. One trip to Yahoo or Google across the public Internet is all that's necessary to prove this. IP, however, scales on its own terms.

IP was initially designed as a protocol that could survive the elimination (intentional or otherwise) of large parts of the network in an ad-hoc manner [1]. Attempts to provide any but the most basic attempts at Traffic Engineering (TE) are typically foiled by the

protocols propensity to survive. This lack of an effective means for controlling IP as it propagates through a network is a major hindrance, as effectively applied TE techniques are a powerful means for enforcing Quality of Service (QoS) constraints and increasing overall network efficiency and reliability.

Though initially created as a mechanism for rapid switching of IP packets, Multiprotocol Label Switching (MPLS) [2] has quickly become the dominant technology for providing TE and QoS in IP networks. Using MPLS, network operators can create tunnels by which QoS and TE parameters can be effectively enforced.

At the heart of MPLS is a connection oriented forwarding plane. This forwarding plane allows for the creation of tunnels on which explicit route and QoS parameters can be enforced. These MPLS tunnels are referred to as Label Switched Paths (LSPs) and are defined by a series of short, fixed length labels between Label Switched Routers (LSRs).

Though it is envisioned that MPLS will allow the next generation Internet to embrace real-time traffic such as voice and video that require specific QoS constraints, it is clear that best effort traffic will still dominate network traffic for the foreseeable future. While significant work has been done to address the Traffic Engineering and provisioning requirements for real-time traffic, the important problem of Traffic Engineering and active management of best-effort services has not been addressed; addressing this critical issue is the focus of this thesis.

1.2 Thesis Motivation

Though next generation networks will consist of multiple classes of service, best effort traffic will still be a dominant force. By its nature, sources of best-effort traffic compete for network resources. Managing the interaction between these sources during times of network congestion is referred to as congestion control and is the primary determinant of the efficiency and performance of best-effort traffic.

Explicit Rate (ER) congestion control describes a system whereby a flow is allocated an explicit rate at which it can offer traffic into the network. By assigning explicit rates to best-effort network flows, the network can actively manage congestion. Using a weighted fair rate algorithm to calculate explicit rates, best effort Label Switched Paths (LSPs) can be granted differentiated explicit rates thus giving network operators flexibility in offering premium services while ensuring network users are treated fairly.

Though significant work has been done to implement an explicit rate congestion control system for the Available Bit Rate (ABR) class of service in Asynchronous Transfer Mode (ATM) networks [3], this service has not gained wide popularity. ATM network creators focused on real-time voice and video as the “killer applications” for next generation networks. In the ATM model, the proper handling of packet-based protocols such as IP was a secondary consideration as it was envisioned that these protocols were unsuitable for transporting voice and video. While the idea that voice and video will be significant players in the next generation Internet slowly comes to realization, the hypothesis that packet based networks are incapable of properly serving these networks is consistently being disproved. IP is, and continues to be, the protocol of choice for transmitting voice and video traffic.

ABR ER congestion control service imposes significant signaling and computational overhead, and has a limited ability to efficiently handle IP, and more specifically TCP traffic. This is due to the paradigm on which it was built. In the ABR class of service, ATM Virtual Circuits (VCs) are assigned a fair rate that imposes a hard upper limit on the rate at which the VC can offer traffic into the network. Assigning fair rates in real time for typically bursty network connections requires extremely high frequency signaling and fair rate calculations. The network bandwidth and computing power required for this task is considerable. Unfortunately, it has been shown that, in addition to adding significant network and computational overhead, the ABR congestion control mechanism reduces TCP throughput [4, 5].

A properly designed and constructed ER congestion control system will allocate rates and manage traffic such that packets are dropped as far upstream as possible. This important feature significantly increases network aggregate throughput. ER congestion control can also solve problems such as unconstrained UDP traffic affecting the native TCP congestion control mechanism, or malicious users modifying TCP congestion control parameters to get unfair shares of bandwidth, thus further increasing the quality and fairness of best-effort traffic transport.

To manage and traffic-engineer best effort service in the envisioned multi-service next generation Internet, several critical issues must be addressed. These include:

- How to remedy the scalability issues that plagued the ATM ABR service while retaining the valuable benefits to network fairness and throughput that are the promise of ER congestion control. Implementing such a service is a key component to managing and transporting best-effort traffic in next generation data-centric networks.

- How to properly devise and implement an explicit rate congestion control mechanism that is compatible with IP/TCP bursty On/Off traffic while simultaneously ensuring that the devised mechanism is indeed compatible and does not conflict with TCP own congestion control mechanism. This issue is expected to have a profound impact on the overall performance of current and future data-centric networks where IP/TCP packets constitute more than 90% of the traffic crossing today's long-haul backbones. This is because IP applications are the fastest growing segment of a service provider's network traffic. This growth is expected to continue well into this century.

1.3 Thesis Statement

This thesis addresses the previously unanswered question of how to handle best effort traffic in MPLS-based next-generation data-centric networks. Specifically, this thesis examines the technical feasibility and assesses the performance analysis for implementing an integrated framework for managing and traffic engineering best effort traffic in next generation data centric networks. We devise and demonstrate a practical and scalable strategy that will enable TCP compatible best effort congestion control with limited overhead in a large-scale network.

Through proof and simulation, this thesis will show that the proposed integrated congestion control framework successfully enforces weighted fair rates for sessions while simultaneously increasing aggregate network throughput. Additionally, the work outlines a method for integrating ER based best-effort traffic into a multi-service class network.

This comprehensive system is composed of four components which include a method for using Resource Reservation Protocol (RSVP) [6] to signal explicit rate information, a Weight Proportional Max-Min [7] (WPMM) fair rate algorithm, a priority based packet marking and active queue management system, and an integrated method for coupling best effort service with real-time and guaranteed bandwidth services in a unified framework.

The implementation of the proposed system proceeded in two phases. In the initial phase [8], a system for signaling fair rate using unmodified (RSVP) messages is devised. Signaling overhead and legacy integration are analyzed. To calculate fair rates, a distributed WPMM fair rate algorithm was developed. The algorithm is tuned to work at low refresh rates and calculates weighted fair rates with O(1) computation complexity.

An elegant proof based on bottleneck ordering is provided to show that the algorithm converges to fair rates in arbitrary network topologies.

In the initial implementation, the RSVP signaling mechanism was used to drive the WPMM fair rate algorithm. The fair rates were then enforced by limiting LSPs to injecting traffic up to the fair rate into the core network. This traffic shaping is done at the ingress node of the network.

Based on the results of phase one, a second phase was undertaken to introduce a new paradigm in queue management. The introduction of this new queue management was essential to ensure that the initial congestion control mechanism is compatible/friendly with the native TCP congestion control mechanism. This new paradigm is based on fair rate marking and is shown to drive the network to fair rates without edge queuing. This mechanism is shown to be vastly superior to the initial mechanism in both enforcing fair rates and in network throughput. We call this system TERM (TCP friendly Explicit Rate congestion control for MPLS networks) [9].

The TERM components are designed to work together, and are shown to increase aggregate throughput of mixed TCP and UDP traffic while enforcing weighted fair rates. The key difference between the TERM system and implementation of the initial phase is the method by which explicit rates are enforced. In the initial phase, the network edge device injects traffic into the network at precisely the explicit rate, with excess traffic queued at the network edge.

LSPs in the TERM system are not limited to transmitting at the explicit rate. In TERM, traffic up to the explicit rate is marked with a drop priority of Better than Best-Effort (BBE), and traffic over the explicit rate with a drop priority of Less than Best-Effort

(LBE) [10]. TERM includes a queue management system designed to enforce fair packet scheduling based on BBE/LBE drop priority. This allows for statistical multiplexing of sources that may be using more or less than their explicit fair rate at a point in time. The method enables LSPs to make use of the queuing capacity and excess bandwidth within the network. This drastically reduces the need for high frequency explicit rate feedback thus significantly reducing signaling and computational overhead. Our simulations show good convergence with a signaling rate of 10 feedback iterations per second per LSP [9].

Additionally, we present a framework for integrating the BBE/LBE classes of best effort service with real-time, guaranteed bandwidth service classes into a comprehensive framework for a QoS based network. This is accomplished through the use of Class Based Queuing [11] with a modified approach to queuing to handle the priority drop ordering of BBE/LBE traffic.

Low frequency signaling coupled with efficient calculation of WPMM fair rates makes the TERM system a feasible solution for explicit rate congestion control in large scale networks. To our knowledge this is the first implementation of such a system.

Chapter 2 - Background Information

2.1 Internet Traffic Management

Internet Protocol, and more generally, the Internet Protocol stack was developed with funding from DARPA [1]. DARPA's goal was to build a network that was powerful and flexible, while being resilient to attacks which could rapidly render large parts of the network non-functional. With this goal in mind, engineers set out to build a decentralized network. This network design was truly decentralized, as the vulnerability of any single point of failure was not an option.

The development of the Internet Protocol stack was done in the early 1970s. While very little networking technology from the 1970s survives today, the Internet Protocol stack has not only survived, but has thrived, becoming the basis on which the world's largest and most complex network is built. While over the past 30 years, the networking community has proposed numerous replacements for the Internet Protocol stack, the technology survives and is likely to be with us for many years to come.

2.1.1 The Internet Protocol Stack

Network protocols are typically described as being a stack. This moniker is appropriate; as well designed protocols have distinct layers with well-defined interfaces. These layers are built (or stacked) on top of each other such that implementations of a particular layer in the stack are not affected by the implementation of other layers. The Internet Protocol stack as it's commonly described is pictured in Table 1.

OSI Layer Number	Layer Name	Implementations
5-7	Application	HTTP, FTP
4	Transport	TCP, UDP
3	Network	IP, ICMP
1-2	Link	Ethernet, Wi-Fi

Table 1 - The Internet Protocol Stack

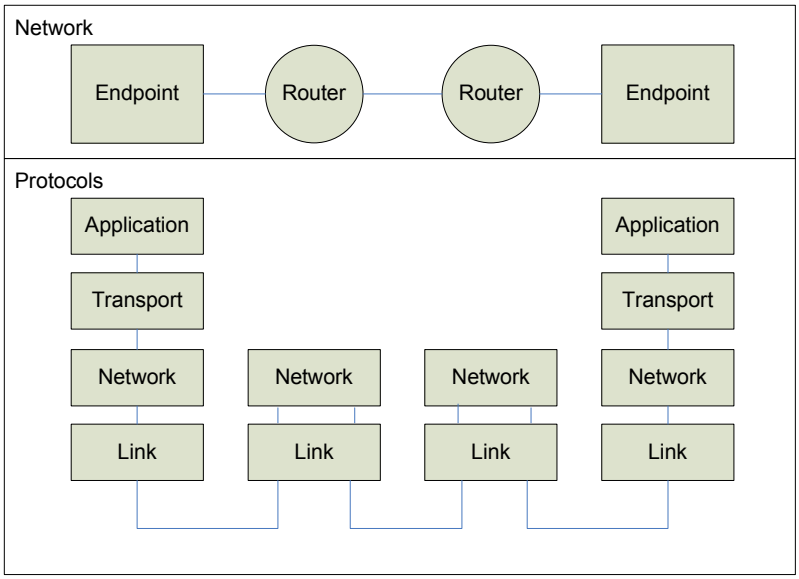

Figure 1 - Internet Protocol Stack Flow

Figure 1 shows the logical flow of data through the protocol stacks of endpoints and routers as the data traverses the network. From this picture it is evident that the network routers are only concerned with the Link and Network layer protocols. They treat higher layer protocols as an opaque payload. The endpoints, however, are concerned with the transport layer protocol, as it is the medium by which the connectivity is made from application to application.

The four layers of the Internet Protocol stack can be described as follows:

- **Link Layer** – The link layer is responsible for communications between adjacent entities. The layer is responsible for coding and integrity of data across the single hop from network device to network device. Perhaps the most ubiquitous Link layer used in Local Area Networks is Ethernet [2], though wireless standards such as Wi-Fi [3], and Wi-Max [4] are quickly moving into this space. In wide area networks, technologies such as Sonet/SDH [5], ATM [6], and Frame Relay [7] make up the Link Layer.
- **Network Layer** – The network layer is responsible for directing traffic across multiple network hops from source to destination. While Internet Protocol (IP) is the most widely used implementation of a network layer protocol, the Internet Protocol stack does contain other Network layer protocols who's purpose are typically administrative. Some of the other Network protocols are management protocols such as ARP [8] and RARP [9].
- **Transport Layer** – The Transport Layer is responsible for communications between applications. The two common Transport layer protocols are Transmission Control Protocol (TCP) [10] and User Datagram Protocol (UDP) [11]. Each has the common

feature that identifies endpoints on a node as ports. UDP is stateless thus no session is established between endpoints, and the packets it sends are independent. TCP creates sessions between endpoints thus ensuring data integrity and ordering of packets.

- **Application Layer** – Perhaps the most ubiquitous Application Layer protocol today is Hyper Text Transmission Protocol (HTTP) [12] which is the language of World Wide Web browsers, thought he number of unique application layer protocols are likely number many thousands. Application Protocols, as the name implies, are a structured way for application endpoints to communicate in a coordinated way. Application layer protocols are typically not validated or enforced by the network, though in some cases, network appliances exist to provide value added services for well known application protocols such ash HTTP and SMTP [13].

2.1.2 IP Network Architecture

At the coarsest grained level, IP networks consist of two kinds of hosts: endpoints and routers. Endpoints produce and consume network traffic while routers direct traffic between endpoints.

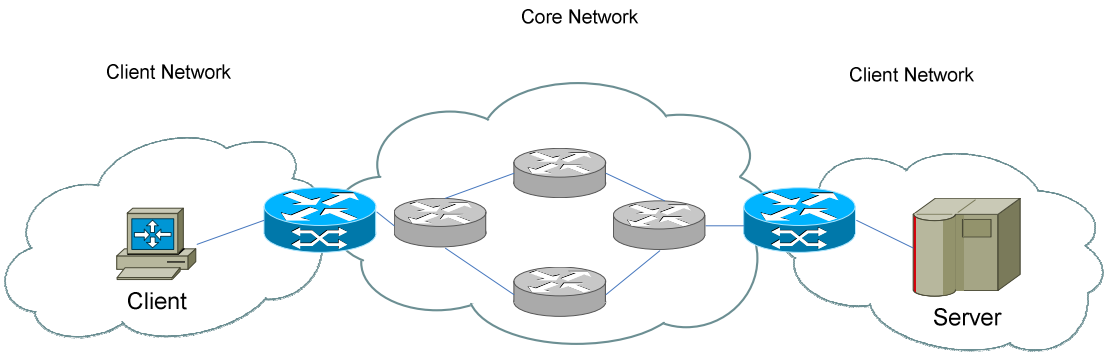

Figure 2 - Simple IP Network

Figure 2 shows a sample IP network. In the figure, the client is communicating with the server through the core network. The figure depicts two distinct types of routers: edge routers that span the client and core networks, and core routers that live entirely in the core network. Edge routers live at the edge of the network and define where the core network ends and the local stub network begins. Core routers are typically much larger than edge routers, handling multiple physical connections and large volumes of traffic. Core routers are typically not the source or the endpoint of network connections.

2.1.2.1 IP Routing

IP is a layer three (network layer) protocol. This means that its purpose is to guide packets through a series of network routers from source node to destination node. The job of the router is to direct and forward packets towards their destination through a network with arbitrary, changing topology. To accomplish this task, the router contains a "routing table" which is a map of where to send each packet.

Destination	Next Hop	Interface
123.194.21.0/24	123.154.32.1	Eth0
192.168.23.0/24	154.12.54.1	Eth1
192.168.0.0/16	154.12.23.1	Eth2
Default	132.24.152.3	Eth3

Figure 3 - Sample Routing Table

Figure 3 shows a typical small routing table for a router with at least 4 Ethernet interfaces. When an IP packet is received, the router uses the destination address in the packet along with the routing table to determine how to best forward the packet towards

its destination. The router examines the destination field in the packet, and performs a "longest match" against the destination entries in the routing table.

To understand the route-matching algorithm it's important to understand the notation used in the Destination field of Figure 3. An address represented as X.X.X.X/Y is an indication of a grouping of addresses with the first Y bits of X.X.X.X indicating the first Y bits of the destination address for each host on the corresponding network. For example, `123.194.21.0/24` identifies a network in which all hosts have the prefix `123.194.21.Z` with Z assuming any value from 1 to 254 (`123.194.21.0` is the network number, and `123.194.21.255` is the broadcast address). Thus a host with an IP address of `123.194.21.78` is in network `123.194.21.0/24`, while a host with the IP address `123.194.31.78` is not. Thus using the above routing table, a router receiving a packet with the destination IP address `123.194.31.78` would forward that packet to the next hop `123.154.32.1` through interface Eth0.

We say that the algorithm used by the router is a longest match because a packet destined for host `192.168.23.32` matches both route table entries corresponding to destinations `192.168.23.0/24` and `192.168.0.0/16,` but since `192.168.23.0/24` has a longer string of matching address bits (and is more specific), it is the match for the next hop and as per the routing table, the packet will be forwarded to next hop `154.12.54.1` through interface Eth1. A packet destined for `192.168.129.5` matches only `192.168.0.0/16` and would be forwarded to next hop `154.12.23.1` through interface Eth3. Typically a route table is configured with a default route to which packets that have no match are forwarded.

2.1.2.2 IP Routing Protocols

While there are some details involved in packet forwarding based on a routing table, the process is fairly straightforward. Where the real complexity lies is in calculating routing table entries given a large complex network with constantly changing topology. A number of effective protocols have been designed for this purpose; we'll review the two most prevalent here.

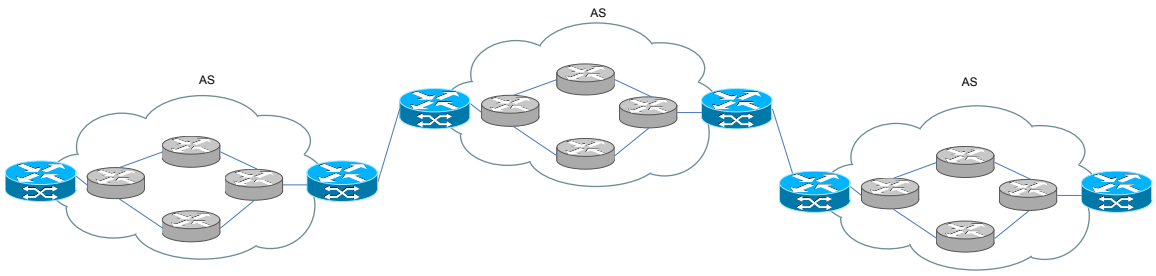

Figure 4 - Three AS Network

There are two distinct levels of granularity on which we calculate routes. An Autonomous System (AS) is a grouping of routers that are closely held such as those belonging to a single ISP, or a corporate network. Routing within an AS is performed using an Intra-AS, or internal, routing protocol such as Open Shortest Path First (OSPF) [14]. Routing between ASs is performed using an Inter-AS, or edge routing protocol such as Border Gateway Protocol (BGP) [15]. Figure 4 shows a network consisting of three Autonomous Systems. In this network, an external routing protocol would calculate routes between ASs, and an internal protocol would calculate routes within an AS.

2.1.3 TCP Congestion Control

The TCP protocol uses a sliding congestion window to manage congestion and reliability. The congestion window is the maximum number of unacknowledged packets a TCP connection can have in flight at a particular time. The TCP protocol uses the Seq and Ack fields in the TCP header to determine the number of outstanding packets with the Seq indicating the highest sequence number of packets sent, and the Ack field indicating the highest sequence number acknowledged packet. The difference between these two values is used to indicate the number of currently outstanding packets.

Because the congestion control mechanism used by TCP is to manipulate window size, and the sender manages the size of the congestion window, it's possible for many implementations of TCP congestion control to exist simultaneously on the same network. This is indeed the case in the public Internet.

The first implementation of TCP congestion control was codenamed "Tahoe" [16]. While Tahoe is not widely deployed today, it was revolutionary as it introduced the concepts of Slow Start and Congestion Avoidance used in most of today's TCP variants. While many variants of TCP congestion control exist, NewReno [17] is the most popular version of TCP used on the public Internet [18].

2.2 Asynchronous Transfer Mode (ATM) Networks

2.2.1 ATM Architecture

Asynchronous Transfer Mode (ATM) [6] networks were developed in the 1990's as the next generation infrastructure for broadband connectivity. The creators of ATM set out to build a network based on Quality of Service constraints. ATM networking technology was designed to be the one standard for data as well as voice, video, and other time and bandwidth sensitive real-time applications. The standards body called the ATM Forum [19] was created, and detailed specifications were developed for the design and implementation of ATM networking hardware and protocols.

2.2.2 Cell Switching

The ATM network paradigm was to support advanced, real-time services. The data-link protocol chosen to accomplish this was based on cells rather than packets. ATM Cells are a fixed length of 53 bytes. Each 53-byte cell is separated into a 5-byte header and a 48-byte payload. The small fixed width cells lent to the ability to handle high data rates efficiently in hardware. The fixed length was also critical to the ability to handle real-time data with throughput and jitter bounds in an effective manner.

2.2.3 Virtual Paths/Virtual Circuits

In ATM, traffic paths are built in a hierarchical manner as shown in Figure 5. The highest level for ATM (and any wired network in general), is the Physical Connection. It is the wire or fiber that connects ATM switches. The next level in the hierarchy is the Virtual Path. A Virtual Path traverses one or more Physical connections in the network. At the lowest level is the Virtual Channel. The Virtual channels carrying user traffic run

the entire length of the network from source to destination. They may traverse one or more Virtual Paths. The Virtual Channel (VC), or more specifically the Virtual Channel Connection (VCC), is analogous to a TCP connection in an IP network.

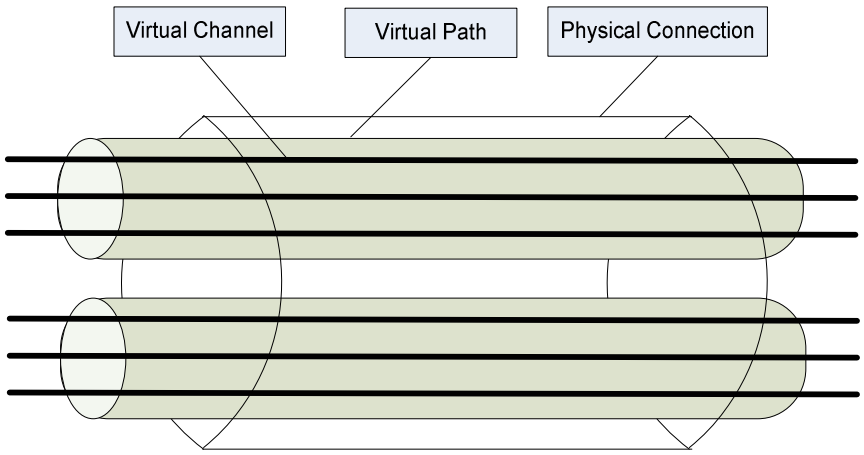

Figure 5 - ATM Hierarchical Flows

Figure 6 shows the format of the 53-byte ATM Cell [6]. The fixed length cell is divided into a 5-byte header and a 48-byte data segment. The 12 bit Virtual Path Identifier and the 16-bit Virtual Channel Identifier respectively indicates which VP and VC the cell should traverse. The 2-bit PT (Payload Type) field indicates to the endpoint the type of data contained in the Data field. The 1 bit CLP (Cell Loss Priority) field indicates to hardware cells that should be dropped first in the event of congestion. The HEC (Header Error Check) field contains a checksum of the header fields.

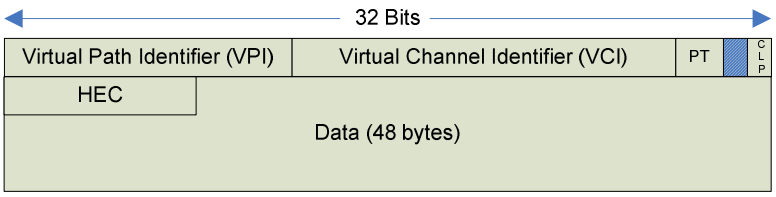

Figure 6 - ATM Cell Layout

2.2.4 Service Levels

ATM networks were designed with 4 distinct service levels. It was expected that these 4 levels of service would be used to handle appropriate traffic types; yielding appropriate service to each level while enforcing QoS constraints.

The Constant Bit Rate (CBR) service is the most straightforward of the services. In this service, dedicated bandwidth reservations are made for VCs. Variable Bit Rate (VBR) service is similar to CBR service in that a bandwidth reservation is made, but differs in that bandwidth that is not being used by a VC can be used by lower priority traffic. The Available Bit Rate Service (ABR) offers best effort service. While no explicit bandwidth reservations are made for an ABR VC, the ABR mechanism includes a comprehensive system for Explicit Rate congestion control. The Unspecified Bit Rate (UBR) class of service is a strictly best effort service whereby traffic is forwarded along a fixed path, but no guarantees on service are made.

2.3 Multiprotocol Label Switching (MPLS)

Traditional IP networks make independent routing decisions at each hop. As an IP packet traverses the network, each router forwards it based on destination address. This stateless, connectionless routing paradigm limits the ability to control traffic flows.

Though it was originally created as a fast-forwarding mechanism, MPLS [20] is often used for its ability to support Quality of Service (QoS) and Traffic Engineering (TE) functionality in IP networks. MPLS introduces a connection-oriented forwarding plane based on short fixed-length labels shimmed between the layer-2 and layer-3 headers. Label Switched Paths (LSPs) are explicit paths through the network defined by a series of labels between adjacent hops.

MPLS simplifies forwarding because routing decisions are made with an exact match on a short, fixed-length label rather than a longest match on an IP Address/Subnet routing table entry. More importantly, MPLS supports explicit routing, as paths are predetermined and setup while establishing an LSP.

Traditional IP routing gives equal treatment to all packets destined for the same address. Once the traffic from two sources are merged, the IP routing mechanism has no provision to again separate them, as it makes routing decisions based completely on the destination address. MPLS solves this problem, as forwarding decisions are made based on the label. Labels can be created and matched with routes in the control plane and these routes are enforced in the data plane. In addition to the freedom and flexibility in routing, MPLS allows for setting resource constraints at setup time in the control plane thus supporting QoS constraints for granular levels of traffic.

LSPs are typically provisioned using Resource Reservation Protocol (RSVP) [21]. Traffic Engineering extensions to RSVP (RSVP-TE) [22] allow LSPs to be specified with associated resource constraints. Label Distribution Protocol (LDP) [23] and the traffic engineering extensions to LDP, Constraint Routing Label Distribution Protocol (CR-LDP) [24] are also supported in many implementations.

2.3.1 MPLS Network Concepts

The MPLS Architecture is formally defined in [20]. This definition contains a high level description of components necessary to understand an operational MPLS network.

2.3.1.1 Label Switched Path (LSP)

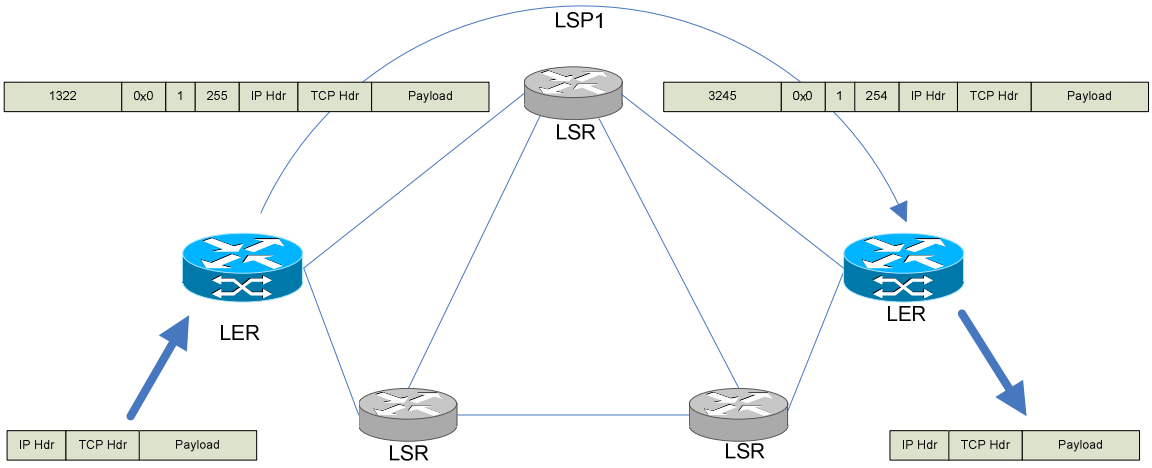

Figure 7 - Label Switched Path

An MPLS LSP is a path from the ingress to the egress of an MPLS network defined by a series of fixed length labels. The labels have significance only between adjacent routers. Upon the provisioning of an LSP, the upstream router creates a label unique for

the incoming interface and advertises that label to the downstream router via RSVP. Figure 7 shows a typical network LSP.

2.3.1.2 Label Switched Router (LSR)

An LSR is a router capable of forwarding MPLS packets based on labels. The LSR is neither the source nor the destination of an LSP; rather it is an intermediary in a core network. It participates in the LSP setup process by assigning unique label numbers to incoming LSPs, and swaps labels on the packets in and LSP by means of a Next Hop Label Forwarding Entry (NHFLE) table.

2.3.1.3 Label Edge Router (LER)

In addition to all of the functionality performed by an LSR, an LER can serve as a source or a destination of an LSP. As such, it must be able to associate IP traffic with the appropriate LSPs. It does this through the use of a FEC-to-NHLFE [25] table. Additionally the LER communicates with interior LSRs to exchanges packets on LSPs and to exchange labels during LSP setup. An LER will sometimes be classified as ingress or egress with respect to a particular LSP.

2.3.1.4 Forward Equivalence Class (FEC)

An FEC is a description of a class of traffic that requires the same handling and should thus be treated equally by the network. The simplest FEC is a FEC defined by all packets destined for the same IP subnet. There are many other factors that may be considered while determining the factors used to identify packets that belong to a FEC. These may include bandwidth requirements, or delay/jitter constraints. The FEC is used to map

packets to a particular LSP. The LSP is typically created to service the requirements of the packets in the FEC.

2.3.1.5 Label Stack

While in the simplest case, an LSP will traverse the network from ingress to egress, there arise cases where LSPs will be tunneled inside of other LSPs. MPLS supports this mode of operation by creating a stack of labels. This stack of labels is managed in the packet by creating a stack of Shim Headers. The Shim Header at the top of the stack identifies the current LSP that the packet is traversing, but at the egress of the LSP, the current Shim Header is popped off the stack and the packet is forwarded appropriately. There are provisions in the Shim Header to identify when a label is the only one on the stack. When the last label is popped off the stack the packet is routed based on it's level-3 protocol header.

2.3.2 MPLS Network Data Structures

MPLS entities maintain a number of important data structures. These data structures are used to determine the functioning of the MPLS routing mechanism. We will examine a few of them here as it will clarify the method by which MPLS routes packets.

2.3.2.1 Shim Header

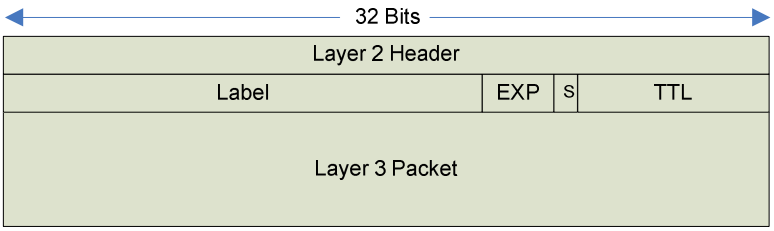

Figure 8 - MPLS Shim Header

MPLS inserts a header between the layer-2 and layer-3 protocols. This header is referred to as the Shim Header. The MPLS Shim Header shown in Figure 8, it contains 4 fields:

- **Label** – a 20-bit field indicating the LSP to which this packet belongs. A particular label value only has significance between adjacent routers.
- **Experimental** – a 3-bit field that's function is yet undefined in the standards. Typically used to propagate DiffServ [16] code point information between IP and MPLS networks.
- **Stack Pointer** – a 1-bit field used to indicate that this label is the lowest label in label stack.
- **Time To Live** – An 8-bit field that carries the number of hops this packet can traverse before it is dropped. Hops in the path typically decrement this field. It is used to drop packets stuck in network loops.

2.3.2.2 Next Hop Label Forwarding Entry (NHLFE) Table

The NHLFE [27] table is used to map incoming labels to outgoing labels. When an MPLS encoded packet reaches an MPLS enabled router (LSR, LER), it contains a Shim header with a Label. The label contained in the header is significant only in that it identifies the LSP the packet belongs to between the previous hop and the current hop. The label corresponding to the packets LSP between the current hop and the next hop will likely be different. The NHLFE table therefore maps the incoming label/interface pair to the outgoing label/interface pair. The act of changing the incoming label to the outgoing label is referred to as label swapping.

2.3.2.3 FEC-to-NHLFE Table

In addition to forwarding MPLS packets based on the NHLFE table, a LER must have the ability to map IP packets to LSPs. The LER classifies IP packets based on the FEC to which they belong. Once a packet is identified as belonging to a particular FEC, the LER uses the FEC-to-NHLFE table to route packets to the appropriate LSP, and ultimately the proper label and interface. The FEC-to-NHLFE table is therefore a mapping of FEC to outgoing label/interface.

When an IP packet reaches an LER, the FEC for the packet is identified. From the FEC, its outgoing label and interface are determined by a lookup in the FEC-to-NHLFE table. The LER then inserts the appropriate Shim Header, and forwards the packet on the appropriate interface.

Chapter 3 - Initial Approach

3.1 Design

3.1.1 Goals

Our goal is to introduce mechanisms necessary to enable effective congestion control for best-effort service in MPLS networks. We used the following criteria to guide the design.

- *Handling of Internet Protocol traffic must be Optimal.* IP networks dominate the current network landscape, with the majority of the world's traffic being sent using TCP or UDP. Therefore the system must work seamlessly with IP protocols.
- *Differentiated service levels must be supported.* Max-min explicit rate algorithms used in ABR service converge to fair but equal rates for all users. A network operator should have the flexibility to specify differentiated service levels (or weights) for allocating best effort bandwidth across LSPs.

- *Computational and signaling overhead must be minimal.* A primary design goal of MPLS is to reduce load on routers. As such, the computational load for feedback iterations should be minimized. Algorithms designed for MPLS systems should calculate rates in constant time. Specifically, the time it takes to calculate the fair rate for an LSP must be minimal, and cannot be a function of the number of LSPs that traverse a link.
- *All per-LSP traffic control must be done at the network edge.* While each of the LSRs in the LSP path are involved in calculating the explicit rate, the ingress LSR should be responsible for any per-LSP traffic management before forwarding traffic to the core LSRs. This will alleviate complexity and performance implications of doing per LSP queue management in the core LSRs.

3.1.2 Congestion Control Overview

Congestion control schemes can be roughly categorized into two classifications: binary congestion notification, and explicit rate congestion control. Binary schemes work by notifying the source of traffic that congestion is occurring in the network. In the case of TCP, this congestion notification comes in the form of dropped packets. Other networking protocols support congestion notification through a Congestion Indication (CI) bit. These schemes share the property that, though there is an indication that congestion has occurred in the network, no precise instructions are given on how to handle it.

Binary congestion control schemes are simple to implement as the network need only detect congestion, and appropriately mark packets as such. Typically, indications of congestion are a message to decrease the rate of transmission for a source. It has been

shown that this simple congestion implication leads to oscillations in network throughput severely limiting network efficiency [1].

Explicit rate congestion control aims to remedy the instability of binary congestion control schemes. In an explicit rate congestion control model, the network itself indicates to the source the exact rate at which the source can transmit. Well-implemented explicit rate congestion control schemes have been shown to manage network congestion in a fair and efficient manner [2], though the implementation and operation of explicit rate schemes are more complex.

Fair rates can be calculated in either of two ways. Centralized schemes collect all flow information in a central place, calculate explicit rates for all flows, and distribute explicit rates out to all network flows. While this scheme is simple to understand and implement, it suffers from the same scalability and robustness concerns as all centrally located network arbitrators.

In a more typical large-scale network scenario, a distributed approach is taken. Tracer packets are used that traverse the network on the same path taken by the packets of the corresponding flow. As the tracer packets cross network routers/switches, appropriate calculations are performed, and the tracer packets are updated with explicit rate information.

Tracer packets may be inserted in the network flow at deterministic intervals. This scheme makes the number of tracer packets proportional to the rate of the connection. This can have positive effects, as the number of updates for fast flows is proportionally greater than the rate for slower connections. The overhead for fast connections, though, can be substantial.

An alternate scheme is to insert tracer packets at fixed time intervals. While this does not have the property that update interval time decreases as network rates increase, it does scale well in large-scale networks with varied high and low throughput connections.

In a typical deployment, a distributed max-min fair rate algorithm is used to calculate fair rates for network connections. As described previously, a signaling mechanism using tracer cells is used to drive the algorithm to fair rates. There are many examples of distributed max-min fair rate algorithms.

Due to the extensive work done on the Available Bit Rate (ABR) class of service in Asynchronous Transfer Mode (ATM) [3] networks, we used ATM ABR as a starting point four our work.

3.1.3 ATM ABR Congestion Control

3.1.3.1 ABR In-Band Signaling

The ABR explicit rate congestion control mechanism inserts Resource Management (RM) cells in band with the network flow. An RM cells is inserted in-band typically after every 32 data cells. These RM cells follow a path identical to the data cells. Upon reaching the destination, they are returned to the source along the same path, thus making a complete round trip from source to destination and back to source using the same path in each direction. This is shown in Figure 9. Distributed explicit rate congestion control algorithms running on each switch in the path modify the Explicit Cell Rate (ECR) field in the RM cells in such a way that when the RM cell completes its round trip, the ECR field contains a fair rate at which the source should transmit. When the originating source receives a reverse RM cell, it sets its current rate to the value of the ECR field. Figure 10 below shows the layout for an ABR RM cell.

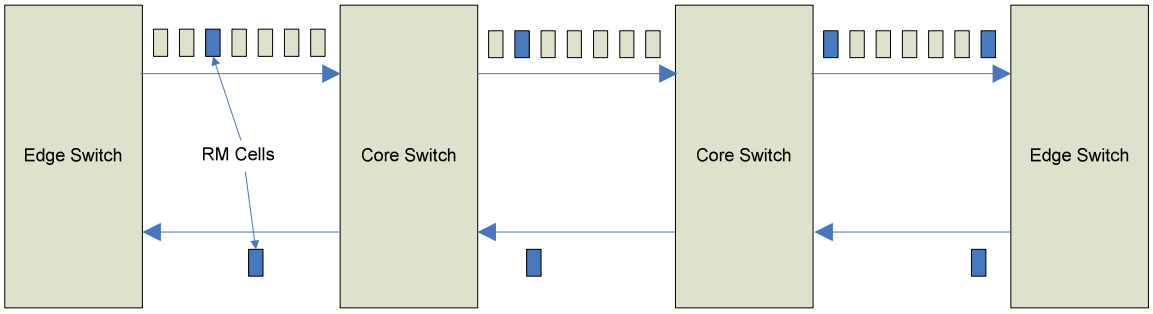

Figure 9 - ATM RM Cell Flow

ATM ABR service relies on a tight feedback loop where a typical VC of 100 Mbps would send and receive 7370 RM cells per second (one RM cell per 32 cells in-band). This tight feedback loop imposes a 6% bandwidth overhead and creates significant computational load on switches in the path.

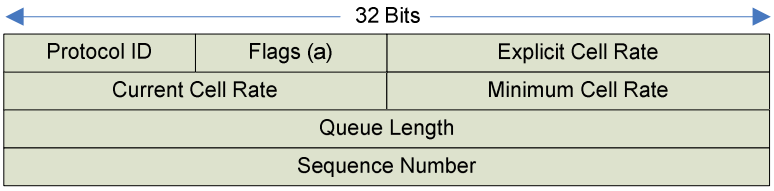

Figure 10 - ABR RM Cell

The Flags field (a) is segmented as follows:

Bits	Field
1-3	Reserved
4	Request/Acknowledge
5	Non-Increase
6	Congestion Indication
7	BECN Cell
8	Direction

3.1.3.2 ABR Distributed Algorithms

3.1.3.2.1 Binary Feedback Schemes

Binary schemes use a single bit to instruct a source to either increase or decrease rate. Early binary feedback schemes were built on “negative polarity” meaning that the source would continue to increase rate until there is a network indication to decrease rate. The problem with this scheme is that as network congestion increases, the probability of loosing a cell instructing a source to slow down increases. Thus the source continues to increase, exacerbating the problem. Subsequent implementations used a “positive polarity” scheme. Bits in the header were set to indicate congestion; switches in the path indicate excess capacity by clearing the bit.

Binary schemes have a number of attractive properties. They are simple to implement and deploy in a production network. Because of the limited feedback they provide, they are generally computationally simple, and thus impose a minimal performance overhead. Additionally, they are simple to understand for network designers and operators.

Binary schemes also have a number of drawbacks. Because the information provided in any feedback iteration is minimal, the convergence time for a network of connection can be long [4]. Additionally, and most significantly, binary schemes are prone to oscillations in the steady state as they use queue lengths for rate calculations [1]. Queue length is a highly variable metric in optimal conditions, but when queues are full, the queue length does not give complete information on congestion conditions thus the algorithms essentially guess at the congestion conditions in the network. These drawbacks of binary schemes were greatly exacerbated by the introduction of high-speed networks.

3.1.3.2.2 Explicit Rate Schemes

It was found that the drawbacks of the binary congestion schemes made them inappropriate for modern high-speed networks. The explicit rate approach was adopted as the standard for rate control in ABR.

One of the earliest explicit rate algorithms for ATM networks was the MIT algorithm [5]. To calculate a fair rate, this algorithm used the following formula is run iteratively across all active connections until a fixed point is reached:

$$\text{Fair Rate} = \frac{\text{Capacity} - \sum \text{Bandwidth of satatfied VCs}}{\text{Number of VCs} - \text{Number of satisfied VCs}}$$

It was shown that the algorithm converges to fair rates in 4k round trips where k is the number of unique fair rates in the network.

This implementation is significant because it served as a baseline for later developments in ATM congestion control algorithms. It did, however, suffer from a number of significant drawbacks. The time complexity for calculating the fair rate for a single VC is O(N). This means that the time to calculate the fair rate for a single VC is a function of the number of VCs in the network. This severely limits the scalability of deployments. Also, the scheme does not include any provisions for measuring actual traffic or congestion, thus a connection requesting a rate, but not using the fair rate, essentially wastes network bandwidth.

Since the introduction of the MIT algorithm there have been significant advances in the implementations of ABR explicit rate algorithms. There are a number of very good

sources that review the state of the art in ABR congestion control schemes [7]. Essentially, the current set of algorithms not only use the link capacity and requested rates for calculating fair rates, they also use network queue depths and current rate of connections. The need to do this is based on the fact that the explicit rate indicated in the RM cell is a hard upper limit on the rate at which a connection can transmit. The stochastic nature of networks leads to traffic that can, at some level, be probabilistically multiplexed into an equivalent bandwidth less than the sum of the peak rates for all connections. This limitation in the ABR congestion control standard has increased the complexity of implementations to the point where viable commercial implementations of algorithms have not been widely adopted.

3.1.4 Initial Design

In a manner similar to the implementation of ATM ABR service, the initial implementation of the proposed system is composed of three components []. First, a method for using RSVP to signal explicit rate information is presented. This signaling mechanism uses existing fields within currently defined RSVP objects to facilitate integration with legacy hardware.

Additionally, an elegant and powerful Weight Proportional Max Min (WPMM) algorithm is presented that calculates weighted fair rates in constant time. A proof is provided that this algorithm converges to network wide fair rates in O(M) time where M is the number of fair rates in the network. Additionally, it is shown that the algorithm, on average, converges to network wide fair rates in O(log M) time.

In this initial model, all sources are constrained at the edge router to offering traffic into the network at the assigned explicit rate. This is similar to the edge queuing used in ATM ABR service.

3.2 Signaling Mechanism

3.2.1 Resource Reservation Protocol (RSVP)

RSVP [6] is a network-control protocol that allows for the provisioning of specific characteristics for network flows. In the context of MPLS, the network flow is defined as an LSP. RSVP was designed for use in large-scale networks as such it is lightweight and flexible, and gracefully handles network interruptions or failures.

RSVP is not a routing protocol, but can work with existing routing protocols to enforce explicit routes for network flows. RSVP is a transport layer protocol in the IP stack, though it does not carry any application specific data, rather it caries control plane date used to provision characteristics of flows.

RSVP uses a round trip mechanism to provision LSPs. To create a new LSP, an RSVP PATH message is sent on the forward path from source to destination, traversing each LSR in the path. Upon receiving an RSVP PATH message, the destination LSR replies with an RSVP RESV message along the identical path in the reverse direction. Each LSR in the path uses the RESV message to make necessary bandwidth reservations and install labels, thus creating an LSP. RSVP is a soft state protocol meaning that without being refreshed periodically, an LSP will time out and will be removed from the network. To prevent timeout, PATH and RESV messages are send periodically to maintain state of the LSP.

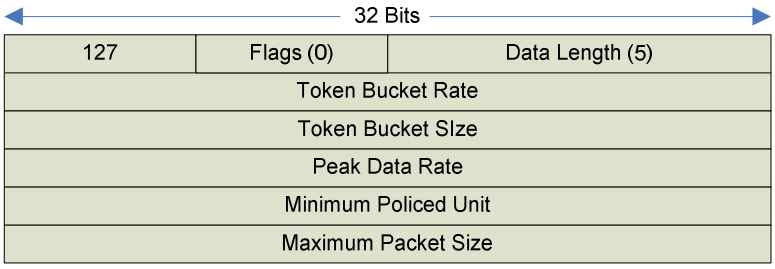

Figure 11 - RSVP TSpec Object

Figure 11 shows the layout of the RSVP Traffic Specifier (TSpec) [7]. The TSpec is included in RSVP PATH and RESV messages used to provision LSPs. The TSpec conveys the traffic characteristics of the proposed LSP to the LSRs along the path. The sender sets the *Token Bucket Rate* and *Token Bucket Size* to indicate the average rate and buffer requirements of the LSP. Additionally, the sender can indicate a proposed maximum data rate by setting the *Peak Data Rate* field. An LSR can reject the request, but granting the request does not guarantee that the requested bandwidth will be available at all times. In the PATH message (forward direction), the TSpec is contained in the SENDER_TSPEC portion of the RSVP message. In the RESV message (reverse direction), the TSpec is carried in the FLOWSPEC portion of the RSVP message.

Upon receiving a PATH message, the destination copies the TSpec out of the SENDER_TSPEC into the FLOWSPEC of a new RESV message. In addition to a TSpec, the RESV FLOWSPEC may contain a Resource Specifier (RSpec). The RSpec component contains a field called *Rate* that instructs each LSR along the path to reserve bandwidth specifically for an LSP. A sender can set the *Peak Data Rate* field to a value greater than the *Rate* field. This is an indication to the LSR that the LSP may use more

than the reserved bandwidth. Any bandwidth above the reserved rate is provided at best effort by the LSR.

Note that while we don't explicitly make use of the Traffic Engineering extensions to RSVP (RSVP-TE) [8] the work presented is compatible with best effort LSP tunnels created using RSVP-TE.

3.2.2 Fair Rate Signaling

This section describes our initial approach for signaling explicit rate congestion control information using RSVP PATH and RESV messages to propagate congestion control information in a manner similar to ABR Forward and Reverse RM Cells. We show that the TSpec contained in both the PATH and RESV messages, along with the RSpec contained in RESV messages, can be used unmodified to transmit necessary information [9]. This can be done without altering the meaning of these fields, and in a manner that enables legacy integration.

The minimum set of fields necessary to enable explicit rate congestion control is *Explicit Rate*, *Minimum Rate*, and *Weight*. Mapping of these fields into RSVP is accomplished as follows:

1) *Explicit Rate*: In a method consistent with its intended use, the *Peak Data Rate* field in the TSpec is used to carry the explicit rate. This value is set by the source to its peak rate, and is modified by LSRs in the path such that at the end of the forward path (PATH message) it contains a fair explicit rate for the LSP. This explicit rate is carried unchanged back to the source in the RESV message. The source uses the final value of *Peak Data Rate* as its explicit rate.

2) *Minimum Rate*: In a method identical to its current use, the *Rate* field in the RESV RSpec is set to the minimum rate required for an LSP. Note that this is set by the destination, and not the source. Readers are referred to [10] for an overview of this mechanism.

3) *Weight*: The *Token Bucket Rate* field in the TSpec object is designed to indicate mean bandwidth requirements for an LSP. This value indicates relative bandwidth requirements across LSPs in the network. We use the bandwidth value in this field to indicate the weight of the LSP. In this manner, network operators can differentiate between LSPs.

3.2.3 Legacy Integration

The proposed signaling mechanism extends RSVP in a standard way to enable signaling of explicit rate congestion control information. As such, it does not introduce breaking changes to non-conforming LSRs.

One factor that requires attention when dealing with legacy LSRs is that when an LSR receives a PATH or RESV message for an existing LSP that does not introduce a state change, the message is not forwarded immediately. Each LSR keeps a refresh timer and periodically propagates PATH and RESV messages as keep-alive indicators to adjacent neighbors. One way to force the propagation of PATH and RESV messages is to include an ERROR_SPEC in each message. The ERROR_SPEC object contains a flag that can be set to indicate that the error message is informational. This informs the LSR that the message should be forwarded immediately but no action is required.

Additionally, since explicit rates are managed on a link-by-link basis. It would be trivial to modify the proposed system to manage fair rates of LSPs on both incoming and

outgoing links. In such a manner, explicit rates could be enforced in a network path with legacy LSR as long as the LSP does not traverse two consecutive legacy LSRs.

3.2.4 Signaling Overhead

The minimal size for an RSVP PATH message is 84 bytes, and the minimal size for an RSVP RESV message is 92 bytes. The RSVP transport mechanism uses raw IP (eliminating TCP or UDP headers). If we consider a network running over Ethernet, the overhead of an RSVP message is the size of the message, plus the size of an IP header, plus the overhead of an Ethernet frame. An IP header without options has a size of 20 bytes. The overhead per packet of Ethernet is 26 bytes.

On an Ethernet network, per round trip, both a PATH and a RESV message are sent. The effective size of the PATH message is 84 (message size) + 20 (IP header) + 26 (Ethernet frame) which is 130 bytes, or 1040 bits. In a similar manner, the size of a RESV message is 138 bytes (92 + 20 + 26) or 1104 bits. Thus for each iteration of the feedback loop, 2144 bits are sent. While the required signaling rate may vary, in our simulations we used a rate of 10 iterations per second. This imposes a signaling overhead of 21.4 Kbps, or 0.21% of the bandwidth on a 10 Mbps LSR.

3.3 WPMM Fair Rate Algorithm

This section begins by reviewing max-min fairness, and max-min extensions to support WPMM fair rates. We then examine design considerations for MPLS fair rate algorithms, and present a WPMM fair rate algorithm that is consistent with our design considerations and calculates weighted fair rates in O(1) time.

3.3.1 Max-Min Fair Explicit Rates

Max-min rate allocation algorithms (and variations on max-min) have been studied extensively and are widely accepted as offering optimal sharing criteria for network flows [11]. Simply put, a max-min algorithm attempts to maximize resources available to the flows with the minimum rates.

3.3.1.1 Example

The "water filling" max-min algorithm gives us a good intuitive feel for how a max-min allocation is calculated. Consider a series of vessels with the capacities listed below in Table 2.

Vessel	Capacity	Fair Allocation
1	0.25	0.25
2	0.50	0.50
3	0.50	0.50
4	1.0	0.75

Table 2 – Max-Min Fair Rates

Assume we have 2 gallons of water and are asked to fill the vessels to a max-min fair volume. We begin by filling each vessel at an equal rate. At the point when we've dispensed 1 gallon, each of the vessels has 0.25 gallons and vessel 1 is full. We continue filling the other three vessels at an equal rate. At the point when we've dispensed 1.75 gallons, vessels 2, 3, and 4 each contain 0.5 gallons, and vessels 2 and 3 are full. We dispense the remaining water into vessel 4 leading to the fair allocations in Table 2. This is considered fair because all sources get an equal share, provided they can use it. Referring back to the above definition of max-min fairness, it is only possible to increase the allocation to the vessel with the maximum allocation. If it were possible to increase the allocation to a vessel that was did not have a maximum allocation, the distribution would not be max-min fair.

If we equate the vessel volume to requested bandwidth and the total volume of water to link capacity, we can easily see how this concept can be used to allocate network fair rates.

3.3.1.2 Max-Min Definition

A network N is typically defined as follows. L is a set of links in network N and l is a link in L. S is a set of sessions in network N, and s is a session in S that traverses one or more links in $l \in L$. r_s is the rate of session $s \in S$. $F_l = \sum_{s \in S_l} r_s$ is the flow on link l. C_l is the capacity of link l. $\boldsymbol{r} = \{r_s | s \in S\}$ is a rate allocation vector for the sessions in the network.

Two conditions must be satisfied for a rate allocation vector to be feasible: $r_s \geq 0$ for all $s \in S$, and $F_l \leq C_l$ for all $l \in L$. An allocation vector $\boldsymbol{r}$ is max-min fair when it is not possible

to increase the rate of a user $r_{s\in S}$ without loosing feasibility or decreasing the rate of another user $r_{s'\in S}$ with rate $r_{s'} \leq r_s$ [11].

Max-min is limited in the sense that the concept of fair is identical for all flows. The term Weight Proportional Max-Min (WPMM) describes a system for allocating fair rates where users are assigned weights (priorities), and are allocated a share of the available bandwidth proportional to their weight. Additionally, WPMM introduces the notion of Minimum Rate (MR) and Peak Rate (PR), which must be satisfied. For an allocation vector to be WPMM feasible, the following two conditions must be satisfied [12], $PR_s \geq r_s \geq MR_s$ for all $s \in S$, and $F_l \leq C_l$ for all $l \in L$. Similarly, an allocation vector $\boldsymbol{r}$ is WPMM fair if it is not possible to increase the rate of a user $r_{s\in S}$ without loosing feasibility or decreasing the rate of another user $r_{s'\in S}$ with rates,

$$\frac{(r_{s'} - MR_{s'})}{W_{s'}} \leq \frac{(r_s - MR_s)}{W_s}$$

where users r_s and $r_{s'}$ have minimum rates and weights of (MR_s, W_s) and ($MR_{s'}$, $W_{s'}$) respectively.

3.3.2 Algorithm Selection Criteria

The water filling algorithm described in the above example is centralized, meaning that to calculate fair rates it must know the state of all LSPs at every LSR. While convenient and effective for some applications, maintaining global state information makes centralized explicit rate algorithms impractical in large mission critical network

infrastructures. Distributed congestion control algorithms more closely fit the requirements for congestion control in MPLS networks.

A discussion of design and testing of fair rate algorithms been covered extensively in existing work [1,4]. The issues with MPLS are similar to those of ATM, and work done on ATM congestion control is of tremendous value in designing fair rate algorithms for MPLS. Implementation details and design goals of MPLS and RSVP, along with the mechanics of the signaling mechanism may, however, influence the design of an appropriate fair rate congestion control algorithm. The algorithm should use only *Explicit Rate*, *Minimum Rate*, and *Weight* fields. The algorithm should also set the fair value for *Explicit Rate* in the forward direction.

3.3.3 Distributed WPMM Fair Rate Algorithm

To our knowledge, there is no WPMM distributed fair rate algorithm that meets our design goals and works with the proposed signaling mechanism. Algorithms have been proposed that calculate WPMM fair rates in O(N) time thus inducing excessive computational complexity [12,13], or that rely on the high frequency of ATM ABR RM cells to monitor queue depths in real time [14] which induces significant signaling overhead. In this section we present an appropriate distributed WPMM fair rate congestion control algorithm. Using the proposed signaling mechanism, the algorithm works in a distributed fashion to calculate WPMM fair rates in O(1) time.

3.3.3.1 WPMM Algorithm

The proposed algorithm categorizes LSPs on a link as either satisfied or bottlenecked. An LSP is bottlenecked at link *l* if an increase in the rate allocated to the LSP at link *l*

could increase its end-to-end throughput. Otherwise it is satisfied, as an increase in the rate allocated on link *l* would not increase its end-to-end throughput. An LSP satisfied on link *l* is either bottlenecked at another link, or is receiving the requested peak rate specified by the source.

The algorithm requires that the LSR maintain values for both link and LSP state. For each LSP j, the LSR maintains values for:

MR_j	Minimum bandwidth for LSP_j
CR_j	The current rate at which LSP_j is transmitting
$A_{j,l}$	Current allocation to LSP_j on link *l*
W_j	The weight of LSP_j
$b_{j,l}$	Binary indication that LSP_j is bottlenecked at link *l*

Per outgoing link *l*, the LSR maintains values for:

B_l	Bandwidth available to best effort traffic on link *l*
BS_l	Bandwidth currently allocated to satisfied LSPs on link *l*
TMR_l	The total of the minimum rates MR_j for all *j* traversing link *l*
TW_l	The total weight of all LSPs that traverse link *l*
TBW_l	The total weight of all bottlenecked LSPs that traverse link *l*

For clarity, we first show how an algorithm could be designed to calculate max-min fair rates. We then introduce our method for calculating WPMM fair rates.

The bandwidth available to bottlenecked LSPs on a link is the total bandwidth available to best effort traffic minus the bandwidth allocated to satisfied LSPs, or $B_l - BS_l$. Therefore, the max-min fair rate for a bottlenecked LSP is $(B_l - BS_l)/NB_l$, or the bandwidth available for bottlenecked LSPs divided by the number of bottlenecked LSPs. The requested Explicit Rate for an LSP at an LSR is the minimum value of the peak requested at the source and the fair rates calculated by upstream LSRs. This value is

carried in the ***Peak Data Rate*** field of the TSpec. If the fair rate for an LSP is greater than the requested ER, the LSP is marked satisfied, the requested ER is added to BS_l, and NB_l is decremented.

Fair rates on a link change as the number and weights of LSPs on the link change. Any iteration of the algorithm could cause a satisfied LSP to become bottlenecked. We therefore need a method for calculating a fair rate for a satisfied LSP to see if it remains satisfied. We cannot use the same formula we use for a bottlenecked connection, $(B_l - BS_l)/NB_l$, because the rate allocated to the satisfied LSP is already included in BS_l. To remedy this, we borrow a technique from [15] to calculate the fair rate for a satisfied LSP.

We calculate the max-min fair rate for a satisfied LSP as if it were bottlenecked by temporarily subtracting the current allocation of the LSP from BS_l and incrementing the bottlenecked LSP count NB_l. Thus the max-min fair rate for a satisfied LSP *j* is $(B_l - (BS_l - A_{j,l}))/(NB_l + 1)$. If the fair rate for an LSP is greater than the requested ER, the LSP is satisfied otherwise it is bottlenecked; appropriate updates are made to BS_l and NB_l.

We extend the above method to calculating WPMM fair rates as follows. The fair allocation of bandwidth for a bottlenecked LSP *j* is

$$FR_{j,l} = \left(B_l - TMR_l - BS_l\right)\frac{W_j}{TBW_l} \tag{1}$$

The allocation to a satisfied LSP *j* is

$$FR_{j,l} = \left(B_l - TMR_l - \left(BS_l - A_{j,l}\right)\right)\frac{W_j}{TBW_l + W_j} \tag{2}$$

In addition to calculating the fair rate as above, we calculate the allocation to this LSP if all LSPs were bottlenecked at this link. This value,

$$FR_{\min,j,l} = (B_l - TMR_l)\frac{W_j}{TW_l} \quad (3)$$

is the lower bound of the fair rate that this LSP will be allocated in the steady state. In our analysis of the convergence time for the algorithm, we will show that this step significantly decreases the time complexity for network wide convergence of the algorithm.

We take the maximum value of the fair rate and the minimum fair rate as the current fair rate for LSP *j* on link *l*.

$$FR_{\max,j,l} = \max(FR_{j,l}, FR_{\min,j,l}) \quad (4)$$

Because the running time of the algorithm is not a function of the number of LSPs in the network, the computational complexity of the algorithm is O(1).

Upon beginning a feedback loop, the source LSR forwards the TSpec towards destination with the TSpec ***Peak Data Rate*** field equal to the requested maximum rate for the LSP, and the TSpec ***Token Bucket Rate*** equal to the average rate, or weight, for the LSP. The destination LSR uses the values in the TSpec of the PATH message and populates a RESV message containing a FLOWSPEC with identical values. The receiver may also insert an RSpec into the FLOWSPEC with RSpec.Rate equal to the minimum rate for the LSP. Note that when a new LSP *j* is provisioned on link *l*, it is initially bottlenecked ($b_{j,l} = 1$) with $CR_j = 0$.

At each LSR in the path, upon receiving an RSVP PATH or RESV message for LSP j:

```
1   If Message has RSpec TMR_l += RSpec.Rate – MR_i; MR_j = RSpec.Rate
2
3   Set ER = TSpec.PeakDataRate – MR_j
4
5   If LSP is Satisfied (b_j,l==0)
6           FR_max,j,l = (B_l - TMR_l - BS_l + A_j,l) * (W_j / (TBW_l + W_l))
7   Else
8           FR_max,j,l = (B_l - TMR_l - BS_l) * (W_j / TBW_l)
9
10  If FR_max,j,l < (B_l - TMR_l) * (W_j / TW_l)
11          FR_max,j,l = (B_l - TMR_l) * (W_j / TW_l)
12
13  If Message Direction is Forward (PATH Message)
14          Set BottleneckRate = min(ER, CR_j)
15          If FR_max,j,l < ER
16                  Set TSpec.PeakDataRate = FR_max,j,l + MR_j
17  Else Message Direction is Reverse (RESV Message)
18          Set BottleneckRate = ER
19          Set CR_j = ER
20
21  Forward RSVP Message
22
23  If FR_max,j,l <= BottleneckRate
24          If LSP is Satisfied (b_j,l==0)
25                  Mark LSP Bottlenecked (b_j,l=1)
26                  Subtract Allocation from Satisfied Rate (BS_l -= A_j,l)
27                  Add LSP Weight to Total BN Weight (TBW_l += W_j)
28          Set A_j,l = FR_max,j,l
29  Else
30          If LSP is Bottlenecked (b_j,l == 1)
31                  Mark LSP Satisfied (b_j,l = 0)
32                  Add LSP Allocation to Satisfied Rate (BS_l += A_j,l)
33                  Subtract LSP Weight from BN Weight (TBW_l -= W_j)
34          Update Satisfied Rate (BS_l += BottleneckRate - A_j,l)
35          Set A_j,l = BottleneckRate
```

In the above algorithm, lines 1 and 3 perform accounting for minimum rates. Note that an appropriate admission control system for managing the granting of minimum rates should be a part of a production implementation, though the design of such a system is outside the scope of this work. Lines 5 through 11 calculate $FR_{max,j,l}$ using the method described above.

Lines 13 through 19 perform per-direction (PATH/RESV) accounting. Lines 14 and 18 are responsible for determining the rate at which this LSP is bottlenecked. In the forward direction, the algorithm works to calculate a lower bound of the bottleneck rate by taking the minimum of the *ER* and the CR_j fields (Line 14). In this manner, we assume that, in the best case, the new bottleneck rate will be the same as the previous bottleneck rate (CR_j). If the *ER* field is less than CR_j, we know that the new bottleneck rate will be less than current bottleneck rate and will be at most *ER*. Note that we don't consider fair rate calculated in the current iteration of the algorithm in this step. On line 23, a comparison of the current value for the fair rate ($FR_{max,j,l}$) and the bottleneck rate is performed. If the fair rate is less than or equal to the bottleneck rate, the LSP is bottlenecked at the current node. In the reverse direction, the explicit rate calculation has completed, and the bottleneck rate is set to the *ER* (Line 18).

In the forward direction, lines 15 and 16 compare the fair rate for this LSP at this node ($FR_{max,j,l}$) against the *ER* that has been calculated by previous nodes. If the $FR_{max,j,l}$ is less than the current value of *ER*, appropriate changes are made to the TSpec ***Peak Data Rate*** to reduce the explicit rate. In line 19, CR_j is set to *ER* for use in the bottleneck rate calculation in the next iteration.

When line 21 is reached, any necessary change has been made to the RSVP TSpec, so the message is released in the appropriate direction. Line 23 determines if this LSP is bottlenecked at this link. Based on the decision, lines 24 through 35 perform appropriate marking and accounting.

3.3.3.2 Example

As an example of the algorithm in action, we consider the network in the Figure 12. As before, links A and B are each 150 Mbps, and each of the LSPs transmits data at an unconstrained rate. Let's assume that at t_0, LSPs 1 through 3 are transmitting data, and the system has converged to fair rates. LSP 1 traverses both links A and B, while LSPs 2 traverses only link A, and LSP 3 only link B. Since each of the links has two active LSPs, it's trivial to show that the fair rates for each of the LSPs is 75 Mbps. We assume that all LSPs are sending PATH/RESV messages at the same frequency. Each LSP has an equal weight of 1 and a zero minimum rate.

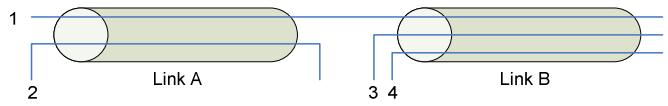

Figure 12 - Simple Network

Let's now assume that at $t_1 > t_0$ LSP 4 is provisioned on link B. Since LSP 4 starts bottlenecked, the Total Bottleneck Weight for link B (TBW_B) is incremented to 3. Since both LSP 1 and LSP 3 are bottlenecked, the Satisfied Bandwidth of link B (BS_B) is zero. A fair rate is calculated for LSP 4 using equation (1) and equation (3) and is found to be 50 Mbps. Therefore, the explicit rate for LSP 4 is set to 50 Mbps.

Next, in the forward direction, LSP 1 is given a fair rate of 50 Mbps at link B in a manner similar to LSP 4. In the reverse direction, when the RESV message reaches link A, it is found that LSP 1 has been bottlenecked elsewhere at a rate of 50 Mbps and thus it is marked satisfied on link A. Its weight is subtracted from TBW_A, and BS_A is set to 50

Mbps. Next LSP 2, which traverses only link A, finds that TBW_A and BS_A are 1 and 50 Mbps respectively. Using equation (1), a fair rate of 100 Mbps is calculated for LSP 2. LSP 3, which traverses only link B, is assigned a fair rate of 50 Mbps in a manner identical to that of LSP 4. At this point, the system has converged to a fair rate vector of (50, 100, 50, 50) for LSPs 1 through 4. This was accomplished with a single feedback iteration for each LSP.

3.3.3.3 Convergence Time

We use a technique from [16] extended to handle weighted fair rates to analyze the time complexity of network wide convergence for the algorithm. Let *N(L,S)* be a network where L is a set of links, and S is a set of sessions. For each link $l \in L$ there is a capacity C_l, and each session $s \in S$ there is a fixed path. S_l is the set of sessions that traverse link l. w_s is the weight of session $s \in S$. $W_l = \sum_{s \in S_l} w_s$ is the sum of the weights of the sessions traversing link l.

The weighted strength J_l of link $l \in L$ is defined as $J_l = C_l/W_l$. Link $k \in L$ is defined to be a homo-source link of $l \in L$ if there is a session $s \in S$ that crosses both k and l. L_l^C is a set including l and all homo-source links of l. l is a min-link in network *N(L,S)* if $J_l = \min_{i \in L_l^C} J_i$. Let L^m be the set of min-links in *N(L,S)*, and S^m be the set of sessions that traverse links in L^m.

N(1)[L(1), S(1)] is the level-1 decomposition of *N(L, S)* where $S(1) = S - S^m$, and $C_l(1) = C_l - \sum_{i \in S_l \cap S^m} r_i$. The decomposition of *N(1)* is the level-2 decomposition of *N*, or *N(2)*. The k-level decomposition of *N* is *N(k)[L(k), S(k)]*. Since S is a finite nonempty set, and

$||S(k)|| < ||S(k-1)||$, N is defined to be an H level network where H is the smallest value of k such that $S(k) = \varnothing$. $L^m(k)$ is the set of min links in $N(k)$, and $S^m(k)$ is the set of sessions that traverse links in $L^m(k)$. In this hierarchy, bottleneck rates of level $k < H$ cannot be determined until the bottleneck rates for levels (1, …, k-1) have been determined. For each $l \in L^m(k)$, l is a bottleneck for all sessions in $S^m(k)$ that traverse l, and the fair rate $r_s = I_l(k) * w_s$ for all $s \in S_l$.

Without equation (3), at each level k of the hierarchy the potential convergence time could be M_k where M_k is the number of unique fair rates for all sessions in $S^m(k)$. This would increase the worst-case time complexity of the algorithm to O(M), where M is the number of bottleneck rates in the system. Equation (3) forces the algorithm to converge to fair rates in two refresh cycles for all sessions in $S^m(k)$. In this way, we show that the algorithm converges in O(H) time, and is a function of the number of levels in the hierarchy rather than a function of the number of unique rates. In the worst case, H could be equal to M making the worst-case convergence time O(M). Since each level in H can contain multiple fair rates, the average time complexity will likely be much less than o(M).

In [17], a similar method of bottleneck ordering was used to analyze convergence time complexity of explicit rate protocols. Simulations were done for linear networks of size of 5 to 400 links, and 25 to 2000 flows. Link capacities and flow paths were randomly generated. 10 simulations were run for each network size. Results showed that the average time complexity for convergence was o(log M).

This analysis shows that, in an arbitrary network topology, the algorithm drives the entire network to fair rates in finite time, and at fair rates the system converges to a fixed

point. Additionally it shows that while the worst case time complexity is still on the order of the number of fair rates in the network (O(M)), decomposing bottlenecks into a hierarchy, and forcing convergence in a single signaling iteration at each level, has been shown to reduce the average running time to a log function of the number of fair rates (o(log M)) thus supporting the claim that the algorithm is scalable.

3.3.3.4 Proof of Convergence

This proof begins by proving the hierarchical bottleneck ordering method for weighted fair rates. It then proceeds to show that the minimum fair rate constraint enforced by equation (3) forces the highest level of the hierarchy to converge in 1 feedback iteration. We show that if the previous level in the hierarchy has converged, the current level will converge in at most 2 feedback iterations. Thus by induction we show that the algorithm converges in a maximum of 2H - 1 feedback iterations where H is the number of levels in the hierarchy.

Definition 1: Let L be a set of links in network N, with l denoting a link in L, and $C_{l \in L}$ denoting the capacity of link l. A session is defined to be a network tunnel traversing a fixed path for which explicit rates are calculated. Let S be a set of sessions in network N, and let s denote a session in S that traverses one or more links $l \in L$. Let $S_{l \in L}$ be the set of sessions that traverse link l. Let $w_{s \in S}$ be the weight of session s, and $W_l = \sum_{s \in S_l} w_s$ be the sum of the weights of the sessions traversing link l. Let r_s be the rate for session s. Let $\boldsymbol{r}: S \rightarrow R^N$ be a rate allocation vector for sessions in the network.

Definition 2: If $r_s \geq 0$ for all $s \in S$, and $C_l \geq \sum_{s \in S_l} r_s$ for all $l \in L$, the rate allocation vector $\boldsymbol{r}$ is said to be *feasible*. An allocation vector $\boldsymbol{r}$ is weighted max-min fair if it is feasible, and it is not possible to increase the rate of a user $r_{s \in S}$ without losing feasibility or decreasing the rate of another user $r_{s' \in S}$ with rates $(r_{s'} / w_{s'}) \leq (r_s / w_s)$.

Definition 3: The weighted strength J_l of link $l \in L$ is $J_l = C_l/W_l$. Link $m \in L$ is a homo-source link of $l \in L$ if there is a session $s \in S$ that crosses both m and l. Let L_l^C be a set including l and all homo-source links of l. Link l is a min-link in network N if $J_l = \min_{i \in L_l^C} J_i$. Let L^m denote the set of min-links in N, and S^m be the set of sessions that traverse links in L^m.

Definition 4: Let $N(0) = N$. We define $N(1)$ to be the level-1 decomposition of $N(0)$ where $S(1) = \left\{ s \mid s \in S, s \notin S^m \right\}$. The decomposition of $N(1)$ is the level-2 decomposition of $N(0)$, or $N(2)$. The k-level decomposition of N is $N(k)$. We define N to be an H level network where H is the smallest value of k such that $S(k) = \varnothing$. Let $L^m(k)$ be the set of min-links in $N(k)$, and $S^m(k)$ is the set of sessions that traverse links in $L^m(k)$.

Let $S(k) = \left\{ s \mid s \in S, s \notin \bigcup_{j=0}^{k-1} S^m(j) \right\}$ be the set of sessions in level k. Let $S_l^B(k) = S_l \cap \bigcup_{j=0}^{k-1} S^m(j)$ be the set of sessions that traverse link $l \notin \bigcup_{j=0}^{k-1} L^m(j)$ and are bottlenecked at a link other than l. Let $S_l(k) = S_l \cap S^m(k)$ where $l \in L^m(k)$ be the set of

sessions on link l not yet bottlenecked at level k. Let $J_l(k) = C_l(k)/W_l(k)$ be the weighted strength of link l in level k where $C_l(k) = C_l - \sum_{i \in S_l^B(k-1)} r_i$ and $W_l(k) = W_l - \sum_{i \in S_l^B(k-1)} w_i$.

Lemma 1: For k, i such that $\| S_l^B(k)\| > \|S_l^B(i)\|$, $J_l(k) > J_l(i)$.

Consider a session $s \in S_l$ that is bottlenecked at a link $m \neq l$ at a level $i < k$. By Definition 4,

$$J_l(i) = \frac{C_l(i)}{W_l(i)}, \text{ and } J_l(k) = \frac{C_l(i) - r_s}{W_l(i) - w_s}.$$

We claim that

$$J_l(k) > J_l(i) \text{ or } \frac{C_l(i) - r_s}{W_l(i) - w_s} > \frac{C_l(i)}{W_l(i)}.$$

This inequality holds if $\frac{C_l(i)}{W_l(i)} > \frac{r_s}{w_s}$. Since s is bottlenecked at link m at level i, $\frac{r_s}{w_s} = J_m(i)$, the inequality holds if $J_m(i) < J_l(i)$, which is true by Definition 3 since m and l share session s, and s is bottlenecked at m at level i. □

Proposition 1: For $0 \leq k \leq H$, each link $l \in L^m(k)$ is a bottleneck for all sessions in $S_l(k)$, and the fair rate $r_s = J_l(k) * w_s$ for all $s \in S_l(k)$.

Assume a session $s \in S_l(k)$ is granted a fair rate $r_s > J_l(k) * w_s$ then there would exist another session $s' \in S$ with rate $r_{s'}/w_{s'} < J_l(k) < r_s/w_s$ and $r_{s'}$ could be increased by decreasing the rate of r_s where $r_s/w_s > r_{s'}/w_{s'}$ thus the rate vector $\boldsymbol{r}$ would not be weighted fair.

Conversely, if a session $s \in S_l(k)$ is granted a fair rate $r_s < J_l(k) * w_s$ then there would exist another session $s' \in S$ with rate $r_{s'}/w_{s'} > J_l(k) > r_s/w_s$ thus you could increase the rate of r_s by decreasing the rate of $r_{s'}$ where $r_{s'}/w_{s'} > r_s/w_s$ and the rate vector $\boldsymbol{r}$ would not be weighted fair.

If all sessions $s \in S_l(k)$ are granted fair rates $r_s = J_l(k) * w_s$, then all sessions bottlenecked at l are bottlenecked at the same weighted rate, and any sessions $s' \in S_l^B(k)$ are bottlenecked elsewhere at a weighted rate $r_{s'}/w_{s'} < r_s/w_s$ (per Lemma 1) thus they cannot be increased by decreasing r_s. Since when link $l \in L^m(k)$, each session $s \in S_l$ is bottlenecked at l or bottlenecked elsewhere, there does not exist a session $s' \in S$ with rate $r_{s'}/w_{s'} < r_s/w_s$ such that $r_{s'}$ can be increased by decreasing r_s thus the rate vector is fair at l.

□

Proposition 2: For $1 \leq k \leq H - 1$, the fair rate r_s for sessions $s \in S^m(k)$ cannot be determined until fair rates $r_{s'}$ for all $s' \in \bigcup_{j=0}^{k-1} S^m(j)$ have been determined.

To determine r_s for a session $s \in S_l$ where $l \in L^m(k)$, we must first determine $J_l(k)$. To determine $J_l(k)$, we must determine $C_l(k) = C_l - \sum_{i \in S_l^B(k)} r_i$ which is the sum of the rates of sessions that traverse l that are bottlenecked at a level less than k. To determine $C_l(k)$, we must first find the rate for sessions in $S_l^B(k)$, which are the sessions bottlenecked at a level less than k that traverse l; $S_l^B(k)$ is not determined until the conclusion of level $k-1$.

□

Lemma 2: Sessions in $S^m(0)$ will converge to fair rates in one feedback iteration.

The fair rates for sessions $s \in S^m(0)$ is $r_s = J_l(0) * w_s$ where $s \in S_l$ and $l \in L^m(0)$. $J_l(0) = C_l/W_l$ is the minimum fair rate that can be allocated to a session on link l. Lines 11 and 12 of the algorithm enforce that, in any iteration, the minimum explicit rate that will be granted to a session on a link is C_l/W_l. By Definition 4, if $l \in L^m(0)$, $J_l(0)$ is the minimum weighted strength of all links s traverses thus $r_s = J_l(0) * w_s$ is the minimum fair rate for s and will be established in any feedback iteration. Links $l \in L^m(0)$ have the property that all sessions $s \in S_l$ are bottlenecked with the same weighted strength $J_l(0)$ which is established once W_l has been established. If we define a feedback iteration such that it starts when W_l is modified and ends when each session $s \in S$ has processed a round trip signaling message, the algorithm will converge to fair rates for $s \in S^m(0)$ in one iteration.

□

Lemma 3: If sessions in $S^m(k-1)$ have converged to fair rates, sessions in $S^m(k)$ will converge to fair rates in two feedback iterations.

At the conclusion of level k, for $l \in L^m(k)$, sessions $s \in S_l(k)$ will be bottlenecked at link l at the fair rate $r_s = (B_l - BS_l) * \frac{w_s}{TBW_l}$. Sessions $s \in S_l(k)$ will be assigned this rate once BS_l and TBW_l are consistent which means that only sessions $s \in S_l^B(k)$ are satisfied at l, and all other sessions are bottlenecked at l. At the beginning of level k, by Definition 4, all sessions $s \in S_l^B(k)$ are bottlenecked at their fair rate at a level $i < j$ at links $m \neq l$. By Lemma 2, $J_l(k) > J_l(i) > J_m(i)$ thus the sessions $s \in S_l^B(k)$ will remain bottlenecked at m.

Sessions $s \in S_l(k)$ are either bottlenecked at l, or bottlenecked elsewhere. If, at the beginning of level k, session s is marked bottlenecked at link l, the resulting change in bandwidth from level k-1 to k will, by Definition 4, make $r_s = J_l(k) * w_s$ minimum along the path and the session will remain bottlenecked at link l.

If, at the beginning of level k, a session $s \in S_l(k)$ marked bottlenecked at $m \neq l$, it will be bottlenecked at l after a single feedback iteration. This is true if, at the beginning of level k, per the algorithm,

$$(B_l - BS_l + r_s) * \frac{w_s}{TBW_l + w_s} < (B_m - BS_m) * \frac{w_s}{TBW_l}$$

for all $s \in S_l(k)$ that traverse m. Since s is bottlenecked at m, this is equivalent to

$$\left(B_l - BS_l + r_s\right) * \frac{w_s}{TBW_l + w_s} < r_s$$

which is equivalent to $\left(B_l - BS_l\right) * \frac{w_s}{TBW_l} < r_s$, or

$$\left(B_l - BS_l\right) * \frac{w_s}{TBW_l} < \left(B_m - BS_m\right) * \frac{w_s}{TBW_m}.$$

This inequality holds if $J_l(k) < J_m(k)$ which is true by Definition 4.

Thus, during the second feedback iteration, BS_l and TBW_l are consistent, and each session $s \in S_l(k)$ will receive a weighted fair rate of $\left(B_l - BS_l\right) * \frac{w_s}{TBW_l}$. Sessions $s' \in S_l^B(k)$ are bottlenecked elsewhere at a weighted rate $r_{s'}/w_{s'} < r_s/w_s$ (per Lemma 1) thus they cannot be increased by decreasing r_s. Since each session $s \in S_l$ is bottlenecked at l at an equal weighted rate, or bottlenecked elsewhere, there does not exist a session $s' \in S$ with rate $r_{s'}/w_{s'} < r_s/w_s$ such that $r_{s'}$ can be increased by decreasing r_s thus the rate vector is fair at l. □

Proposition 3: The algorithm will converge to WPMM fair rates in at most 2H - 1 feedback iterations.

Per Lemma 2, sessions $s \in S_m(0)$ will converge to WPMM fair rates in one iteration. Per Lemma 3, if sessions $s \in S_m(k-1)$ have converged, sessions $s' \in S_m(k)$ will converge

to WPMM fair rates in two feedback iterations. Therefore, by induction it is shown that the algorithm converges to WPMM fair rates in at most (2H – 1) feedback iterations. □

3.4 Simulation Results

Our goal in these simulations is to show that the proposed signaling mechanism and algorithm will converge to WPMM fair rates in non-trivial network configurations. Additionally, results highlight the benefits of using explicit rate congestion control on network utilization.

3.4.1 Simulation Infrastructure

Simulations were performed using NS2 version 2.28 [18] that has been extended to support MPLS-TE [19], and RSVP-TE [20.21]. Enhancements were made to the simulator to implement the best effort traffic management system described in this paper. The simulations used TCP New Reno, the most prevalent variant of TCP on the Internet.

Unless otherwise noted, the propagation delay on links between LSPs is 5ms, the propagation delay between sources and LSRs are 3us. All traffic flows from left to right. Signaling is done a rate of 10 RSVP PATH/RESV messages per second.

3.4.2 NS2 Simulator Background

NS2 is a discreet time network simulator used extensively for development of network and administrative protocols. The implementation is open source and is extended and maintained by a skilled and motivated group of programmers from academia and industry.

It is written in a mixture of C++ and an object oriented version of TCL called OTCL. This combination of a compiled language and a scripting language lend tremendous flexibility to the use of the simulator. The compiled C++ components run fast and are

typically used for tasks that require per-packet processing. The OTCL code is used primarily for configuration of network scenarios, and administrative tasks.

The NS2 implementation provides an object oriented framework for building network components. Using this object model, programmers can easily build extensions to the simulator.

Aside from the core simulator functionality, NS2 contains modules for most common networking technologies. There is a full implementation of the IP stack including most common variants of TCP, UDP, and administrative protocols. While many modules come with the NS2 build, there are many other modules that are created and published by other researchers around the Internet. Our simulations made extensive use of a package created for MPLS and RSVP [22].

The NS2 package also includes a graphical configuration and simulation visualization tool called the Network Animator or NAM, and a suite of tools for graphing results. There are also available tools for generating large topologies for network simulations.

3.4.3 WPMM Algorithm Implementation

Figure 13 shows the C++ source code for the forward direction implementation of the WPMM Fair rate algorithm [9] in the NS2 simulator. This module was built as part of the RSVP implementation.

```
void RSVPAgent::rate_calculation_forward(RESOURCE_MGMT *rm,
                                         psb* p,
                                         int newconn)
{
        double bmax, gmax, amax;

        double be = get_be_capacity(p);
        int w = rm->get_w();

        double est_rate = fmin(rm->get_cr(), rm->get_er());

        gmax = be * ((double)w/term_totalw[p->next_hop]);

        if (p->term_satisfied)
            bmax = (be - term_as[p->next_hop] + p->term_rate)*
                          ((double)w/(term_bnw[p->next_hop] + w));
        else
            bmax = (be - term_as[p->next_hop])*
                          ((double)w/term_bnw[p->next_hop]);

        amax = fmax(bmax, gmax);

        if (amax < rm->get_er()) rm->set_er(amax);

        if (amax <= (est_rate + 0.0000001)) {
            if (p->term_satisfied) {
                p->term_satisfied = 0;
                term_nb[p->next_hop]++;
                term_as[p->next_hop] -= p->term_rate;
                term_bnw[p->next_hop] += w;
            }
            p->term_rate = amax;
        } else {
            if (!p->term_satisfied) {
                p->term_satisfied = 1;
                term_nb[p->next_hop]--;
                term_as[p->next_hop] += p->term_rate;
                term_bnw[p->next_hop] -= w;
            }
}
```

Figure 13 - WPMM Algorithm Source Code

3.4.4 Convergence to Fair Rates

Figure 14 shows a 5-LSR "parking lot" WAN configuration called the "General Fairness Configuration 1." It was designed by the ATM Forum to test ABR congestion control schemes for max-min fairness [3]. The links between adjacent LSRs have a propagation delay of 5 ms. Links between sources and LSRs have a propagation delay of 2.78 us.

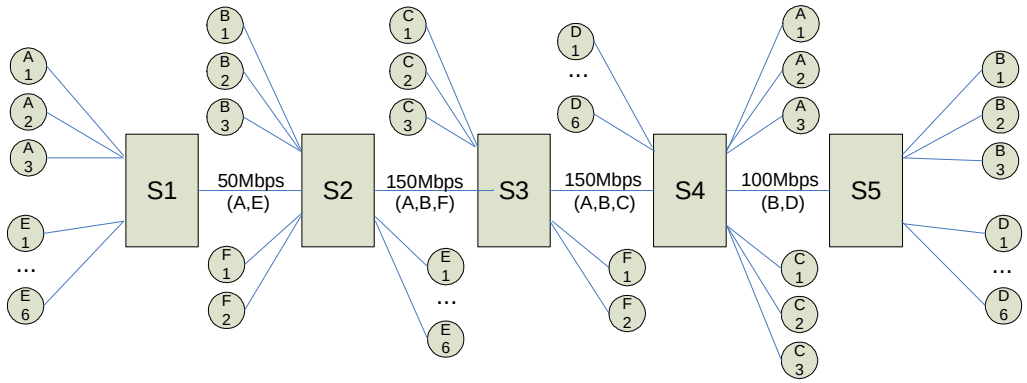

Figure 14 – General Fairness Configuration 1 (GFC1)

The GFC1 configuration consists of 23 LSPs: three A's, three B's, three C's, six D's, six E's, and two F's. The first link is 50 Mbps and carries the traffic of the three A sources and the six E sources leaving the three A LSPs and the six E LSPs bottlenecked at 5.56Mbps. The B and D LSPs share 100Mbps across the fourth link leaving each of them bottlenecked at 11.11Mbps. The LSPs A, B, and F share 150Mbps across the second link The A and B LSPs, bottlenecked at 5.56Mbps and 11.11Mbps respectively, take up 50Mbps of the second link. This leaves 50Mbps for each of the two F LSPs. The A, B, and C LSPs share the 150Mbps on the third link. Again the A and B LSPs take up 50Mbps leaving 33.33Mbps for each of the three C LSPs. In summary, the fair rate vector

for the LSPs of A, B, C, D, E, and F is {5.56, 11.11, 33.33, 11.11, 5.56, 50.00} Mbps. Table 3 summarizes the number of LSPs, ingress node, egress node, and Ideal fair rate for the LSPs in the GFC1 configuration.

We simulated two scenarios. In the first scenario, all sources are started simultaneously and transmit at a constant rate of 200Mbps, which is greater than the capacity of any link in the network. We call this "Unconstrained" as no congestion control mechanism is used. In the second scenario, we set a peak rate of 200Mbps, but employ the proposed RSVP signaling and WPMM fair rate algorithm to throttle admission into the network. We call this scenario "Explicit Rate"

In the Explicit Rate scenario, we transmitted RSVP PATH messages at a frequency of 10 per second for each LSP. This leads to a signaling overhead of approximately 21Kbps, or 0.21% of the bandwidth on a 10Mbps link. Our simulations show that this small signaling overhead, in addition to allocating fair rates for LSPs, leads to significant network throughput gains. In the algorithm, convergence time is a function of round trip congestion control messages. This frequency can be increased or decreased to ensure timely convergence.

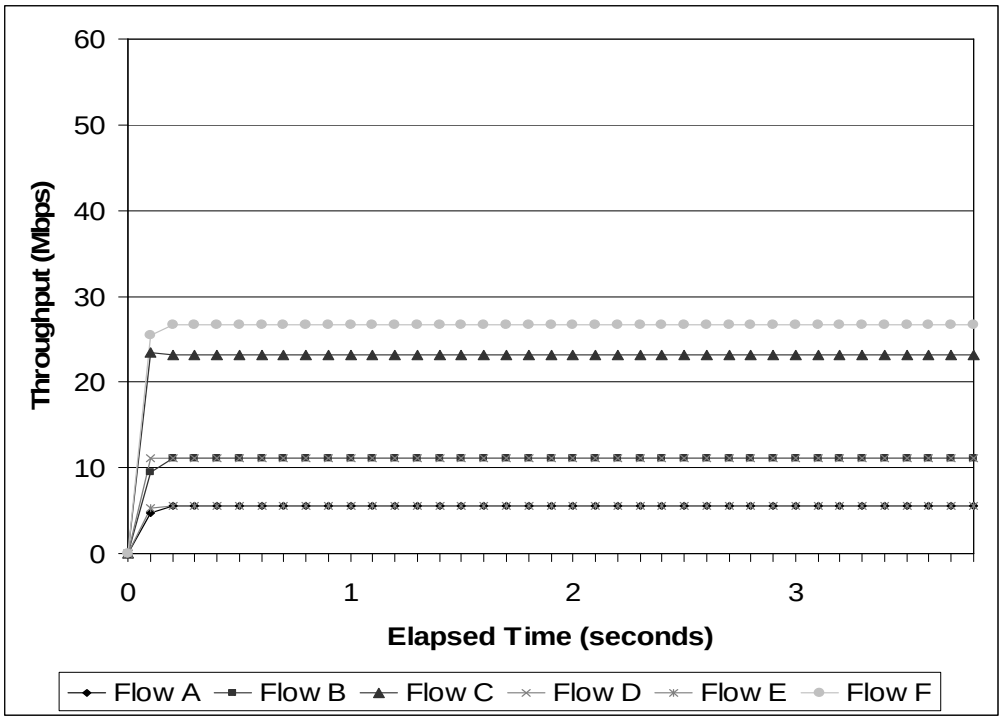

Figure 15 – Unconstrained Scenario

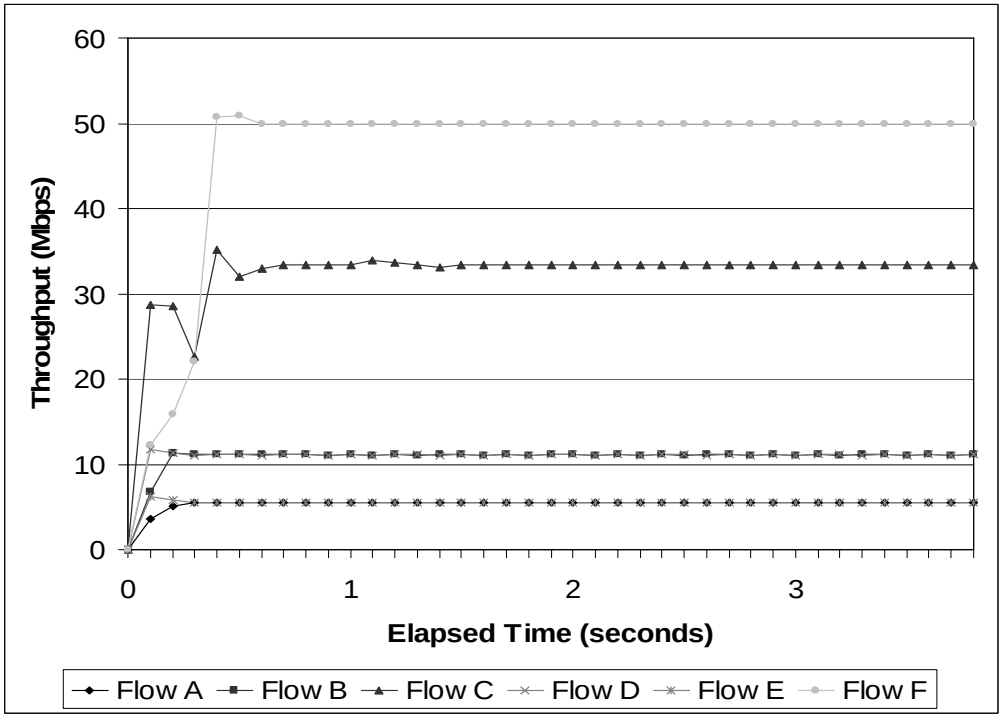

Figure 16 – Explicit Rate Scenario

Figure 15 and Figure 16 show end-to-end throughput for the Unconstrained and Explicit Rate scenarios. Figure 16 is intended to show the steady state convergence of the algorithm. Note that all 23 scenarios are started simultaneously leading to the witnessed convergence period. The next scenario addresses the dynamic convergence of individual LSPs for a network in a steady state. **Table 3** shows the per LSP steady state rate. It shows clearly that the RSVP signaling mechanism drives the algorithm to fair rates.

Group	# LSPs	Ingress	Egress	Ideal	FR	BE	Delta
A	3	S1	S4	5.56	5.56	5.56	0.00
B	3	S2	S5	11.11	11.11	11.11	0.00
C	3	S3	S4	33.33	**33.33**	**23.15**	**10.18**
D	6	S4	S5	11.11	11.11	11.11	0.00
E	6	S1	S2	5.56	5.56	5.56	0.00
F	2	S2	S3	50.00	**50.00**	**26.66**	**23.33**

Table 3 - Steady State Throughput

Additionally, Table 3 shows that imposing fair rates for LSPs leads to greater utilization of resources. We can see that LSPs of class C and F have greater throughput in the Fair Rate scenario. The Fair Rate scenario had an aggregate throughput (summing all LSPs multiplied by rate per LSP) of 350 Mbps, while the unconstrained scenario had an aggregate throughput of 273 Mbps. Thus 28% greater network utilization was achieved using explicit rate congestion control.

This efficiency is a result of an LSPs ingress node throttling traffic to the explicitly allocated rate. The algorithm ensures (without guarantees), that the bandwidth to carry the explicit rate to its destination is available. Transmitting packets part way through a network and dropping them before they reach the destination wastes network resources.

With explicit rate congestion control, most packets that are dropped are dropped at the ingress node leading to an efficient use of network resources.

3.4.5 Staggered Arrivals

The configuration shown in Figure 17 was proposed in [23] to assess the responsiveness of explicit rate algorithms. In this configuration, six flows share a single link. Sources are started one at a time at 1-second intervals. Figure 18 shows the results of this test scenario. Results clearly indicate that the algorithm converges gracefully to fair rates.

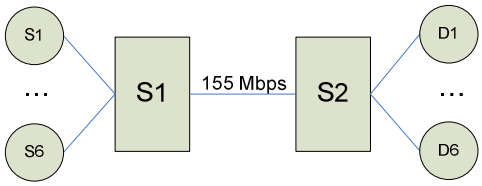

Figure 17 - Staggered Arrival Configuration

Figure 18 - Staggered Arrival Scenario

3.4.6 Weighted Fair Rates

The configuration shown in Figure 19 was proposed in [24]. It is used to assess the max-min fairness across LSPs. Flows 1 through 15 share the bandwidth of the first link and thus should have a fair rate of 10 Mbps. Flows 15, 16, and 17 share link 2. Since flow 15 is bottlenecked in the first link at 10 Mbps, flows 16 and 17 should converge to a fair rate of 70 Mbps each.

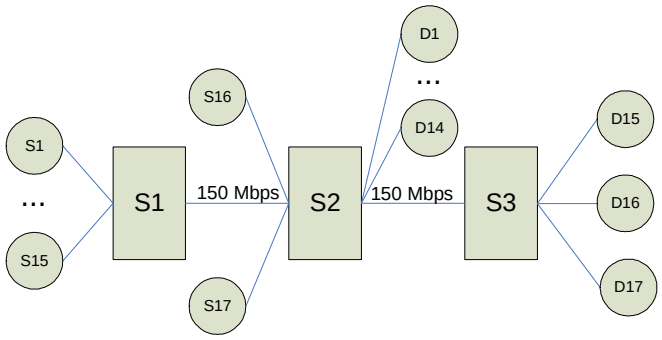

Figure 19 - Upstream Traffic Configuration

In this scenario, however, we impose WPMM fair rates by increasing the weight of LSP 15 to 5 while leaving the weight for the remaining LSPs at 1. Thus the fair rate for LSP 15 is 5 times the rate of each of the LSPs 1-14 on link 1, or 39.5 Mbps. The fair rate for LSP 15 on link 2 is 107.1 Mbps. Since LSP 15 is bottlenecked at 39.5 Mbps on link 1, LSPs 16 and 17 share the excess bandwidth. Figure 20 and Table 4 show the experimental results of this scenario and confirm that the algorithm converges to WPMM fair rates. Once again note that all LSPs were started simultaneously.

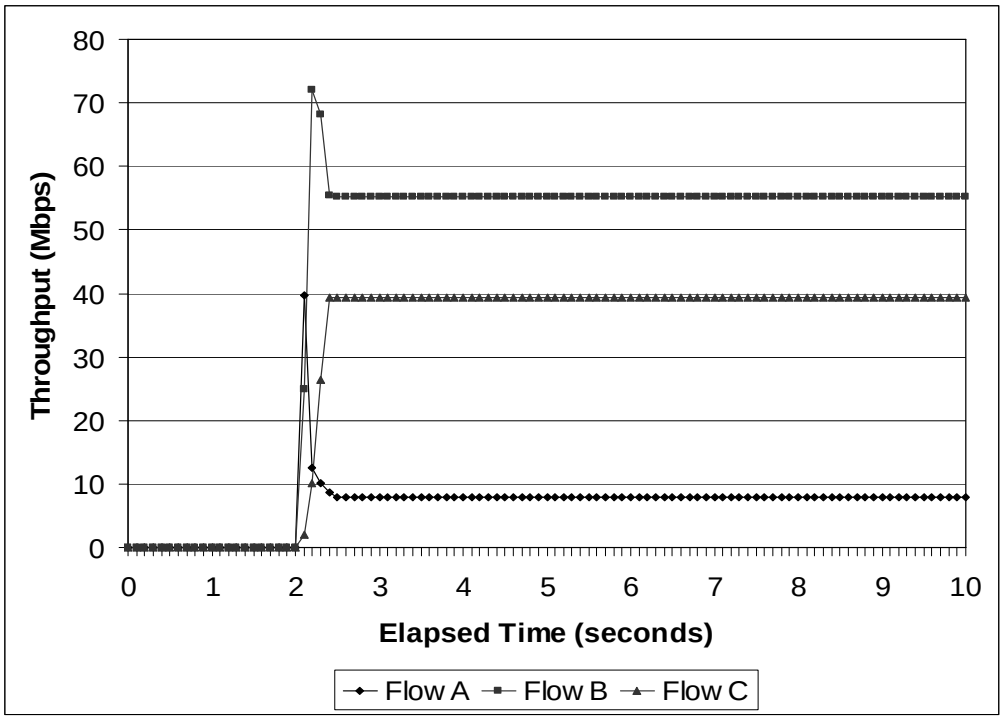

Figure 20 - Upstream Traffic Scenario

	Flows	Weight	Weighted Fair Rate	Actual Rate
A	1-14	1	7.9	7.9
B	16, 17	1	55.2	55.2
C	15	5	39.5	39.5

Table 4 - Upstream Traffic Scenario

3.4.7 TCP Traffic

The simulation of the GFC1 scenario in section 3.4.4 above was modified such that sources A, B, D, and F transmitted TCP traffic at a constant rate, while sources C and E transmitted UDP traffic at a constant rate. From this graph it's easy to see that the initial design of the system has problems working with TCP traffic. Average throughput for this scenario is just 230 Mbps, which is 34% less than the throughput for UDP sources, and 26% less than the throughput of the equivalent network transmitting unconstrained TCP traffic (Figure 22).

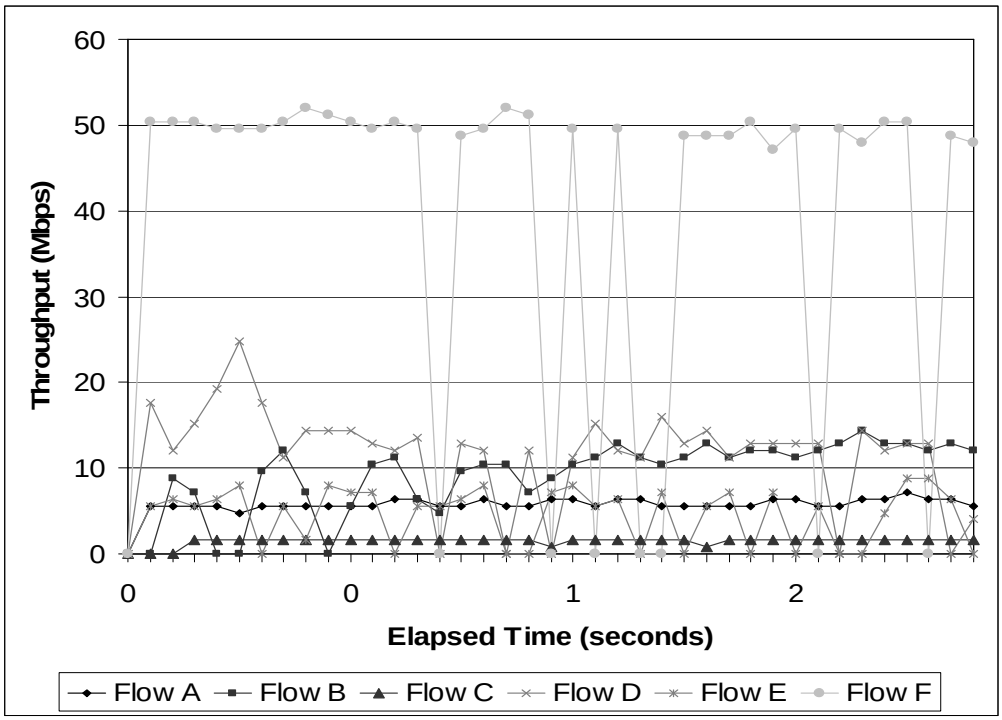

Figure 21 - TCP Traffic

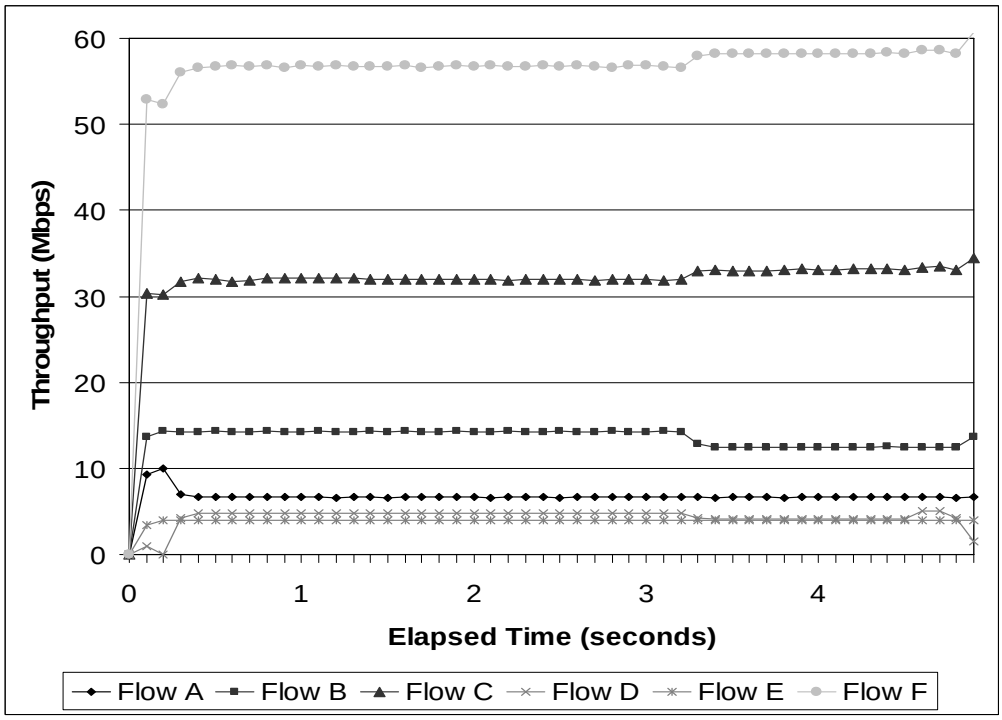

Figure 22 - Unconstrained TCP Traffic

3.4.8 ON/OFF Sources

Figure 23 shows a network with two LSPs traversing a single link of 50 Mbps. Source/Destination 1 are connected by LSP1, and Source/Destination 2 are connected by LSP2. Both sources use a Pareto On/Off source that switches the transmission rate between 0 Mbps and 50 Mbps at time intervals determined by a Pareto distribution. Figure 24 shows the results of the throughput for the scenario where no explicit rate congestion control is used.

In this figure it is clear that each of the sources cooperate. While both sources are transmitting, each is limited to 50% of the link bandwidth, but in the case when one is idle, the other can use the remaining bandwidth.

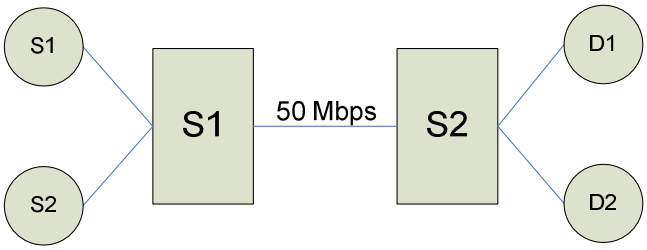

Figure 23 - Two LSP Configuration

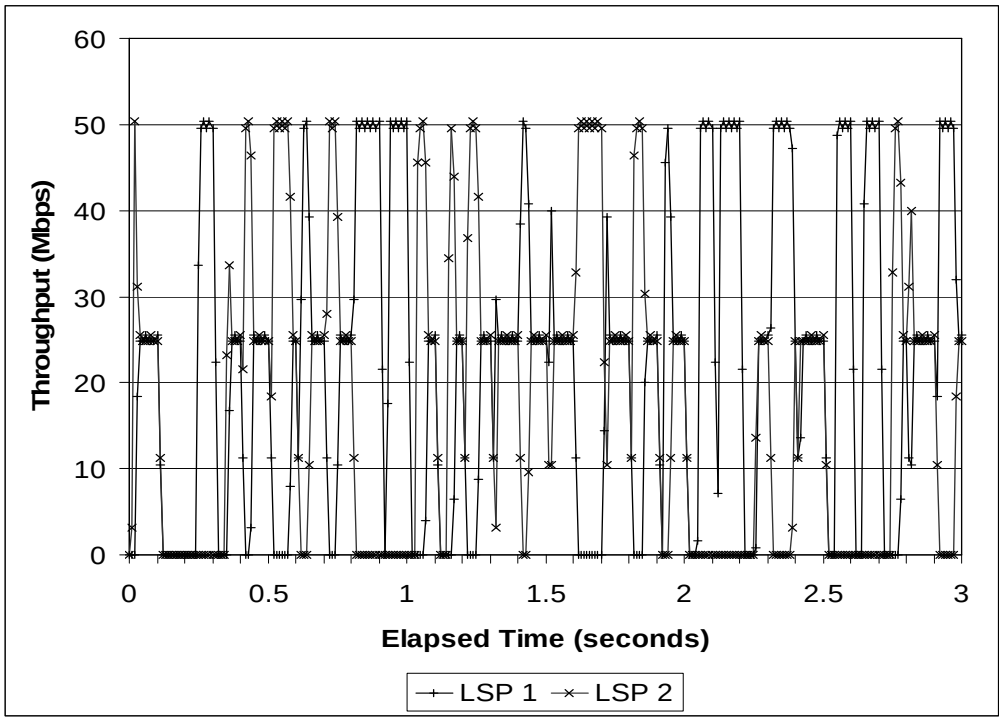

Figure 24 - ON/OFF Unconstrained Scenario

Figure 25 shows the throughput for the scenario where explicit rate congestion control is used to enforce fair rates. This picture depicts a scenario where each of the LSPs is constrained to a peak rate of 50% of the link bandwidth. In this scenario, 28% of the link bandwidth is wasted due to the fact that LSPs cannot share unused capacity.

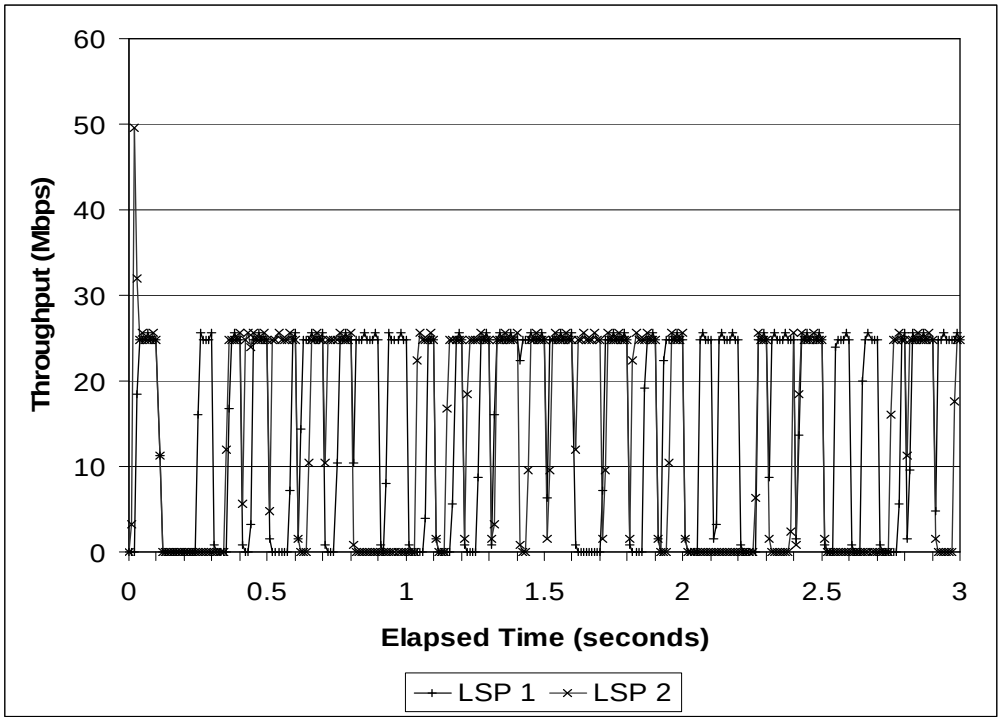

Figure 25 - ON/OFF Constrained Scenario

3.4.9 Limitations of Initial Approach

The initial approach worked well to calculate and enforce fair rates for Constant Bit Rate UDP traffic. The signaling mechanism worked well to calculate and communicate fair rates, and the edge queuing mechanism worked to enforce weighted fair rates for the service.

While a step in the right direction, this initial approach did suffer from serious limitations, and did not satisfy our design goals. First, the system performed poorly with TCP traffic. Introduction of TCP sources into a network proved to be unstable and were severely limited in throughput. Second, at reasonable signaling rates, the sources did not cooperate and significant bandwidth was wasted leading to an inefficient network.

Chapter 4 - TCP Friendly Congestion Control

4.1 Building Blocks

The Initial Approach calculated fair rates that were a hard upper bound on how rapidly a flow may transmit. The fair rates were, in fact, a hard lower bound on the rate a flow should transmit as well, as there is no straightforward way to manage the excess capacity caused by idle connections that have a current bandwidth allocation. This is the precise mechanism used by ATM ABR congestion control and thus it suffered from the same shortcomings.

The hard bounds in the ATM ABR system led to the design of a complex system of real time signaling, and complex congestion control algorithms designed to react quickly enough to assign meaningful fair rates to dynamically changing network flows. As the complexity increased, the cost of implementation and the management overhead of networks that implemented the ABR congestion control system increased.

Despite the cost and complexity of implementing the system, many researchers, and indeed, many hardware manufacturers embraced the idea and worked to make it a commercially viable technology.

Perhaps the biggest shortcoming of the technology and what likely led to its ultimate fall from favor is its limited ability to deal effectively with IP traffic. The ATM implementer's designs for next generation networks had little place for IP, and designs reflected this. Yet the venerable IP infrastructure has not only survived, but thrived to the extent that few other network level protocols exist.

To achieve our stated design goals, the Modified Approach takes a new approach to the management of fair rates for network flows. We call this Modified Approach TCP Friendly Explicit Rate Congestion Control for MPLS or TERM [1].

Rather than setting hard upper or lower bounds on network flow, the TERM system uses fair rates to mark packets with a drop priority of Better than Best Effort (BBE), or Less than Best Effort (LBE). The system makes the determination on how to mark packets in the following way. A flow is assigned an explicit fair rate. The flow can transmit above or below the fair rate, traffic up to the fair rate is marked BBE, and traffic over the fair rate is marked LBE. The network includes a queuing mechanism that properly handles the drop priority marking thus dropping LBE packets before BBE packets. In this manner, congestion is imposed on a per-flow basis, and in a fair or differentiated-fair manner.

Because we've alleviated the necessity to assign precise fair rates, the signaling and algorithmic complexity of the system is vastly reduced. Rather than assigning fair rates thousands of times per second to deal with dynamic network conditions, fair rates in the

TERM system are calculated a few times per second to reflect changes in network flow requirements.

The TERM system makes use of the WPMM Fair Rate Algorithm detailed in the previous section unmodified. Additionally, the system introduces the following significant building blocks to achieve Explicit Rate congestion control:

- A system for priority marking packets based on fair rates at the ingress of the network [2]
- A Queue Management System capable of enforcing the marked drop priorities of packets such that fair rates are enforced [1]
- Extensions to RSVP to handle Explicit Rate congestion control information [3]

These building blocks are explored in greater detail in the following chapters.

4.2 Fair Rate Marking

The bursty nature of IP traffic makes limiting LSP bandwidth to explicit rates inefficient. An LSP that is allocated an explicit rate may at some instant use the exact explicit rate, but will more likely require more or less than the explicit rate. In TERM, the explicit rate is the rate at which an LSP can transmit traffic with a priority of Better than Best Effort (BBE); traffic above the explicit rate is categorized as Less than Best Effort (LBE). The distinction between the classes is that when congestion occurs on a link, LBE traffic queued on the link will be dropped before BBE traffic. BBE and LBE traffic differ in drop priority only. The lowest order MPLS exp bit is used to indicate the drop priority of the packets to LSRs in the path.

The system includes a lightweight estimator to mark packets. Simply put, the estimator estimates if the LSP is transmitting at a rate over or under the explicit rate, and marks packets as BBE or LBE respectively. It is designed such that any packet causing the LSP to exceed the explicit rate is marked as LBE otherwise the packet is marked BBE.

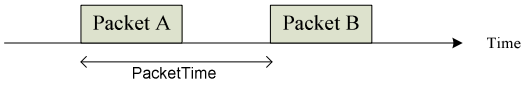

Figure 26 - TERM Estimator Time Budget

The TERM estimator is designed around a time budget (Figure 26). An LSP transmitting data at FairRate will have packet inter-arrival times of PacketSize/FairRate (or PacketTime). The TimeBudget is incremented at each packet arrival. If the TimeBudget is greater than PacketTime, the packet is marked as BBE and the

PacketTime is subtracted from the TimeBudget, otherwise the packet is marked LBE and TimeBudget is not changed. An upper bound on the size of TimeBudget is set so that an idle LSP cannot accumulate a large TimeBudget. This upper bound, MaxBudget is tunable. Our simulations show that a value of twice the maximum packet time is appropriate. The algorithm below is run at the ingress LSR for each packet.

```
TimeBudget  += CurrentTime – LastTime
LastTime = CurrentTime
PacketTime = PacketSize/FairRate
If PacketTime <= TimeBudget
   Mark Packet BBE
   TimeBudget = min(TimeBudget – PacketTime, MaxBudget)
else
   Mark Packet LBE
```

4.3 Queue Management

4.3.1 IP Network Congestion Control

Routers in an IP network are responsible for forwarding IP packets from incoming interfaces to outgoing interfaces. When lightly loaded, this is a straightforward task of mapping IP packet destination addresses to outgoing links. During times of heavy utilization there will often arise the situation where the demand to forward packets on a particular interface exceeds the interfaces ability to accept traffic. This is referred to as network congestion, and measures to ensure that routers act in a consistent manner during times of congestion are known as network congestion control.

4.3.1.1 FIFO Queuing

The most straightforward method of handling network congestion is to create a queue. A queue is a data structure whereby elements are ordered from front to back. A First-In, First-Out (FIFO) queue is implemented such that the most recently inserted element will be at the back of the queue, and the oldest remaining element will be at the front of the queue, other elements will be ordered between the front and the back in the order in which they were received. When there is available capacity for an element to be serviced, the element at the front of the queue is selected, and is removed from the queue.

In a router using FIFO queuing, a single queue is created per outgoing link. When a packet is destined for a particular link, it is inserted into the queue corresponding to that link, and is thus scheduled in order, for service when bandwidth becomes available for the link.

Figure 27 - FIFO Queue

This simple mechanism appears to be fair. From our common experience in supermarkets and banks it seems to be quite a workable solution. Problems arise because of the finite memory capacity of routers. When traffic is queued, it is stored in memory. If the rate at which packets are added to a queue exceed the rate at which they are being serviced, at some point the queue will become full and packets must be discarded.

One common implementation of the FIFO queue is called Drop Tail. In the Drop Tail queue, when congestion necessitates that a packet be dropped, the packet on the tail of the queue is dropped. In times of congestion, Drop Tail queuing has been shown to be less than effective for a number of reasons. Firstly, in the ideal case, connections offering high rates of traffic easily drown out lower rate services. This gives incentives to network sources to transit traffic at a higher rate in times of congestion. Additionally, the Drop Tail queue has been shown to induce a phenomenon called Global Synchronization [4] in IP networks. This is due to the interaction between the network throughput and the TCP native congestion control mechanism. The side effect of Global Synchronization is that the TCP congestion control mechanisms of many TCP sources synchronously oscillate between high rates and low rates of output. This thrashing is detrimental to network performance, and once started, is hard to contain.

4.3.1.2 Round Robin Queuing

Round robin [5] queuing has been shown to alleviate many of the problems plaguing the Drop Tail queuing mechanism. One method of implementing Round Robin queuing is to create a queue for each network flow. When bandwidth becomes available on the link packets are sent from each of the queues in order with the first queue being serviced, then the second, and so on. When the last queue is reached, the first queue is serviced again. In this manner, all flows are granted equal rates and are treated fairly. This also eliminates global synchronization. While, in principal, this solution is appealing, the practical implications in space, computing time, and implementation complexity of managing per-flow queuing make them impractical.

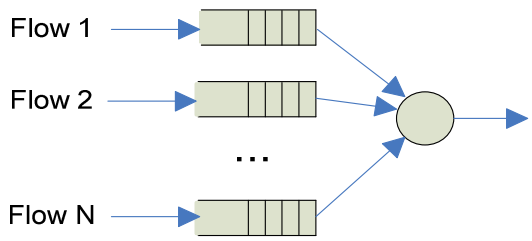

Figure 28 - Round Robin Queue

Another method used to implement Round Robin queuing is to provision N queues, each of the same size, per outgoing link. As packets are received, they are run through a special function called a “hash”. This hash function examines header fields and makes a decision on which of the N queues to deposit the packet on. A properly implemented hash function will deposit all packets of a particular flow on the same queue. Failure to do this may result in packet re-ordering and have detrimental effects on the TCP congestion

control mechanism. A good hash function will distribute packets evenly over the N queues. The problem with round robin queuing is that sources sending small packets will receive an unfairly small share of link bandwidth as the Round Robin scheduling mechanism takes a packet from each queue without regard for the size. Additionally, queues with large packets, or queues for whom the hash function deposits a disproportional large share of packets on, will overflow and drop packets while other queues may have remaining space.

4.3.1.3 Deficit Round Robin Queuing

Deficit Round Robin (DRR) [6] queuing was developed to alleviate the shortcomings of the Round Robin queuing scheme. DRR uses a hash function to separate network flows into a finite number of queues (Figure 29). All queues share the same queue buffer memory. When the buffer memory is full, the longest queue is identified, and the tail packet on that queue is dropped.

DRR uses a credit based system to manage flow on a link. Per queue, DRR manages a deficit counter. It iterates round robin through queues adding a quantum (a small integer value in bytes) to each deficit counter. After adding the quantum, the value of the deficit counter is compared to the size of the packet at the head of the queue. If the value of the deficit counter is larger than the size of the packet, the packet is released and the packet size is subtracted from the deficit counter. In this manner, queues are serviced at equal fair rates.

Since there are a finite number of queues, there is a possibility that multiple flows will share a single queue. In this scenario, the flows sharing a queue will behave in a manner similar to a drop tail queue. This condition is referred to as a queue collision.

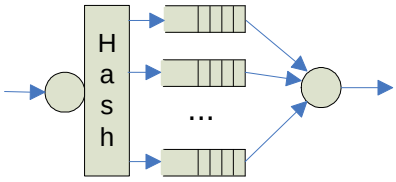

Figure 29 - DRR Queue

4.3.2 TERM Queue Management

4.3.2.1 Goals

Our goal is to design a queue management system that will enforce fair rates on LSPs in such a way that the throughput of TCP will not be detrimentally affected. Additionally, the system must enforce fair rates within the network while not incurring excessive signaling or computational overhead. As such, the only signaling information that is available to the queuing mechanism is the BBE/LBE priority marking performed by the TERM fair rate marking component, thus the system must not require per-LSP information. Additionally, the system must process packets in O(1) time. That is, the computational time to process a packet must not be a function of the number of LSPs traversing the router. The clock time required to process the packet will of course be a function of the destination link bandwidth and current queue depth on the link.

Based on these restrictions, the system must use the BBE/LBE marking to enforce fair rates on LSPs. The fair rates must be enforced in such a way that the detrimental effects of combining TCP and UDP traffic are minimized. The queuing system must not re-order packets as excessive re-ordering can have detrimental effects on the TCP native congestion control mechanism.

4.3.2.2 TERM Queue Management System (TQMS)

In the TERM system, all packets offered at the edge of the network are potentially injected into the core. It is the job of the queue management system in the LSRs to use the drop priority marked in the packet to drive the system to stable fair rates. LSRs typically queue packets per outgoing link. One way to accomplish fairness between BBE and LBE traffic is to create two queues per outgoing link; one for LBE traffic and one for BBE traffic. The BBE queue can be drained before the LBE queue. This solution, however, leads to re-ordering of packets, which has detrimental effects on the native TCP congestion control mechanism.

Another solution is to implement a single queue per outgoing link, and upon filling the queue, drop LBE packets that arrived most recently. This queuing mechanism is known as Drop Tail (with a modification to drop LBE packets first). Both practical experience and analytical studies have shown that the Drop Tail queuing mechanism leads to unfair allocations, and can induce global synchronization of TCP sources leading to network thrashing [4].

To remedy this, we introduce the TERM queue management system (TQMS). TQMS builds on the work done on DDR to fairly queue packets and minimize the interactions between competing flows. TQMS adds the dimension that allows it to enforce weighted fair rates base on BBE/LBE fair rate marking. This is achieved by changing the congestion profile seen by competing flows such that packets are dropped in a manner proportional to the BBE/LBE ratio.

Most significantly, TQMS differs from DRR in the way it manages the credits to queues during the round robin process. DRR segments traffic into a finite number of queues using a hash function. Each queue has a deficit counter. When there is capacity on a link, the DRR packet scheduler for the link visits each of the queues that service the link in order. When a queue is visited, the deficit counter is incremented by a fixed value called a quantum, and the packet at the head of the queue is compared to the size of the deficit counter. If the packet at the head of the queue is smaller than the value of the deficit counter, the packet next to be sent on the network link. In this way DRR elegantly enforces fair queuing in a way that minimizes the interaction between competing flows without introducing excessive computational overhead.

TQMS performs the packet selection mechanism in a similar manner with the following exception. In TQMS, the quantum added to the deficit counter in each queue is not a fixed value, rather a maximum quantum size is set. The actual quantum added to the deficit counter of a queue is the maximum quantum size multiplied by the fraction of the size of BBE traffic to the size of all packets in the queue. In this way, TQMS uses the packet marking to enforce fair rates by allocating greater bandwidth to queues that contain greater proportions of packets marked BBE.

In the TQMS system, a minimum quantum size is established as to ensure that no queue gets entirely starved, as starvation has negative effects as the starvation of TCP connections causes the connections to go into the slow start phase which floods the network with traffic and can lead to network oscillations.

Another way in which TQMS differs from DRR is the manner in which it manages the dropping of packets. In DRR, when limited memory causes the router to drop a packet,

the DRR mechanism identifies the longest queue and drops the most recently received packet. This packet dropping mechanism is appropriate when all packets are equal. In the TERM system, packets are marked with the drop priorities of BBE or LBE with the intention that the LBE traffic be dropped before BBE traffic. As such, in times of congestion, the TERM system identifies the queue with the most LBE traffic and drops the most recently received LBE packet. In this way the system enforces the drop priority thus establishing different congestion profiles for LSPs based on the proportion of BBE to LBE traffic that is a function of the fair rate allocated versus the volume of traffic offered into the network.

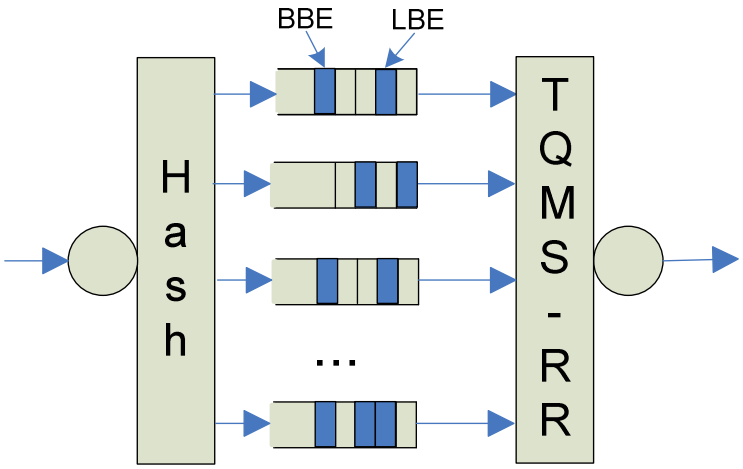

Figure 30 - TERM Round Robin Queue

4.4 TERM Modified RSVP Signaling Approach

Though the initial signaling approach of using RSVP unmodified showed promise, feedback was that overloading the parameters in the existing RSVP PATH and RESV messages proved to be confusing and inelegant. The modified signaling approach also uses RSVP PATH and RESV messages in a manner similar to the use of RM cells in ABR. Rather than re-using parameters in the existing TSpec message, the modified approach uses the standard method for extending RSVP to add explicit rate signaling capabilities to the protocol.

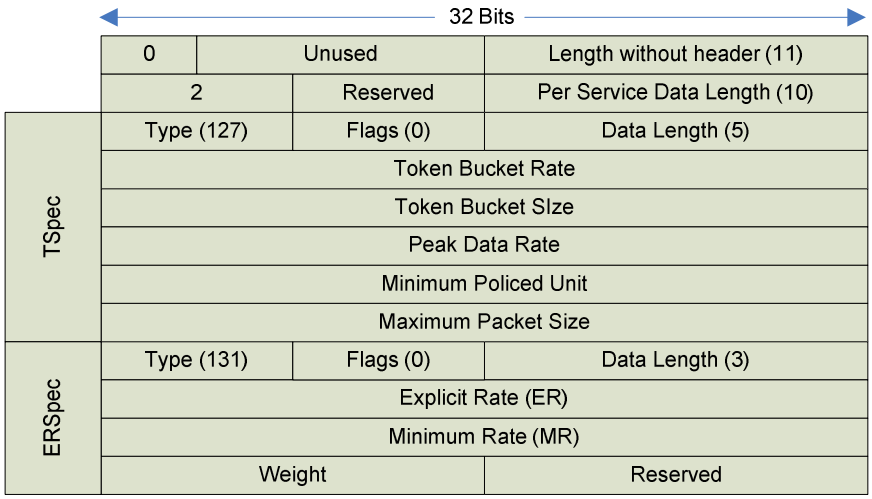

Figure 31 – RSVP FlowSpec Object

We propose creating an Explicit Rate Specifier (ERSpec). The ERSpec contains three fields: *Explicit Rate*, *Minimum Rate*, and *Weight*. The ERSpec will be appended to the SENDER_TSPEC and FLOWSPEC objects. Figure 31 shows a FLOWSPEC object with both a TSpec and an ERSpec. RSVP PATH/RESV messages containing the ERSpec will

be sent at regular intervals for each active LSP. The source and intermediate LSRs will use the ERSpec in a manner similar to the use of RM cells in ABR.

The ERSpec *Explicit Rate* (ER) field is set by the source to the desired peak rate for the LSP. The fair rate algorithm running on each LSR in the path modifies the value of the ER field such that, at the end of the round trip it contains a fair explicit rate.

The ERSpec *Minimum Rate* (MR) field is set by the source to the minimum rate necessary to properly serve this LSP. This field is similar to the Rate field in the RSpec in that accepting the value of this field indicates that an LSR will make the specified minimum rate available at all times. An LSR that cannot service the minimum rate requirements can reject the request.

The ERSpec *Weight* (W) field carries an integer weight used by the fair rate algorithm to calculate differentiated explicit rates for different LSPs. The network provider can set the Weight field. Alternately, it has been shown that allowing customers to set the *Weight* field (and charging accordingly) results in equilibrium. In equilibrium the utility is maximized and the rates allocated are WPMM fair [7]. The ATM ABR congestion control mechanism did not support differentiating service across VCs.

The ABR RM cell layout [8] contained a number of fields that have been deliberately excluded from the ERSpec. The RM cell format included bit fields such as *Congestion Indication* (CI), and *No Increase* (NI). These fields were used in binary congestion control schemes that proved to be unstable in dynamically changing networks [9]. Additionally, the *Queue Length* field has been omitted. Both the *Queue Length* and the binary congestion indicators relied on rapid in-band feedback typically employed in ABR networks, and are inconsistent with our design goals.

The ERSpec also lacks an equivalent to the RM cell *Current Cell Rate* field. The *Current Cell Rate* field is used to communicate the current explicit rate assigned to a VC. Many ABR congestion control algorithms calculate explicit rate in the reverse path, and inform switches of the result in the forward path of the next feedback iteration. Since in ABR, the interval between feedback iterations is small, the delay between calculating an explicit rate and informing switches in the path of the result is small.

Since we envision the round trip time of the ERSpec to be much shorter than the interval between feedback iterations, informing LSRs of the explicit rate result in the next iteration is not timely. MPLS congestion control algorithms should set explicit rate in the forward direction (PATH Message) and update LSRs with the result of the explicit rate calculation on the reverse trip (RESV Message). This timely feedback significantly enhances the stability of distributed fair rate algorithms in low frequency signaling scenarios.

Note that the RESV FLOWSPEC object may also contain a Resource Specifier (RSpec) that carries terms *Rate*, and *Slack Term* that can be used to specify hard bandwidth reservations and jitter tolerances for the LSP. Though not incompatible, we expect the use of the RSpec and ERSpec to be mutually exclusive in the FLOWSPEC object. Also note, we do not outline a method for LSP merging in this paper. Details about LSP merging should be addressed in a standardization effort [3].

4.5 Simulation Results

Our goal in these simulations is to show that the TERM signaling mechanism and algorithm will converge to WPMM fair rates, and that these fair rates coupled with the TERM packet marking and queue management system drive the network to a fair allocation of best effort traffic. Simulation results clearly show that this method is effective in controlling both TCP and UDP traffic. Additionally, results show significant benefits of using explicit rate congestion control on network utilization.

4.5.1 Convergence to Fair Rates

Figure 32 shows a 5-switch "parking lot" WAN configuration called the General Fairness Configuration 1 (GFC1). It was designed by the ATM Forum to test ABR congestion control schemes for max-min fairness [8]. Table 5 summarizes the number of LSPs, layer 4 IP protocol, ingress node, egress node, and fair rate for the LSPs in this simulation.

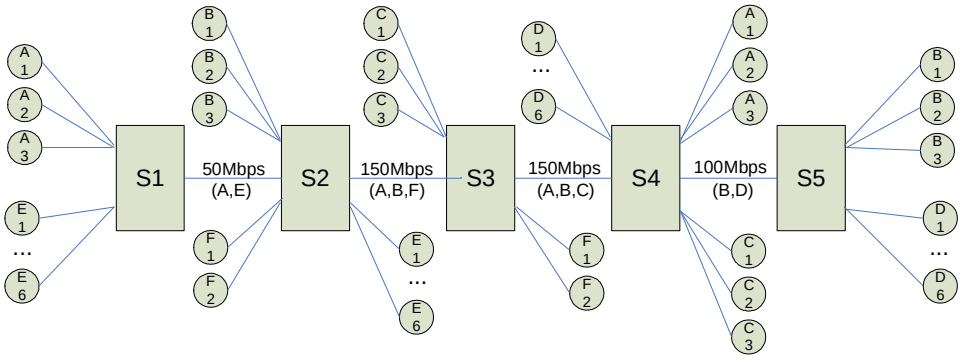

Figure 32 – General Fairness Configuration (GFC1)

A combination of TCP and UDP sources were configured to transmit traffic at a constant rate of 100Mbps, and each had an equal weight. The TERM signaling mechanism and WPMM explicit rate algorithm were used to assign fair rates for LSPs. The TERM packet marking and queuing management system was used to manage the drop priority of packets on the LSPs.

	LSPs	Type	Ingress	Egress	Fair Rate
A	3	TCP	S1	S4	5.56
B	3	TCP	S2	S5	11.11
C	3	UDP	S3	S4	33.33
D	6	TCP	S4	S5	11.11
E	6	UDP	S1	S2	5.56
F	2	TCP	S2	S3	50.00

Table 5 - GFC1 LSPs

Figure 33 shows the end-to-end throughput for each of the LSPs in the network. From this simulation it is clear that in a steady state, with all LSPs transmitting at a constant rate, the network flows converge to fair rates. This simulation demonstrates a number of important points. First, the signaling mechanism and WPMM fair rate algorithm are working properly to assign fair rates to LSPs. Second, the packet marking and queue management system is correctly marking traffic based on fair rates, and in times of congestion, traffic marked LBE is being dropped before traffic marked BBE. Third, the interactions between responsive traffic (TCP) and unresponsive traffic (UDP) are being managed fairly.

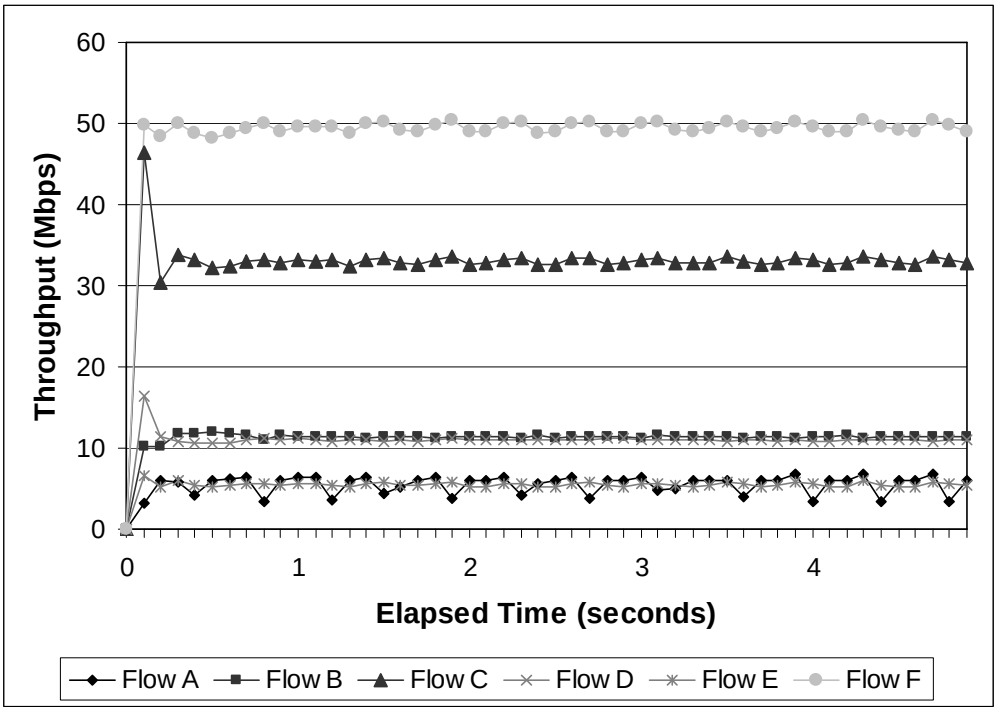

Figure 33 - GFC1 End-to-End Throughput

Figure 34 shows the identical scenario with unconstrained traffic. From the picture it is clear that the LSPs are not receiving fair rates as documented in Table 5. Additionally, the TERM scenario had an aggregate throughput of 350 Mbps while the unconstrained scenario had a throughput of only 316 Mbps. This shows that in addition to enforcing fair rates on LSPs, the TERM system increased throughput in this scenario by 11%.

When compared to Figure 21 which represents the throughput for the original thesis approach, it is clear that the intelligent queuing and marking employed in the TERM system is effective in managing TCP traffic in this scenario. The TERM throughput is 52% greater than the 230 Mbps achieved with the initial approach using constant bit rate traffic.

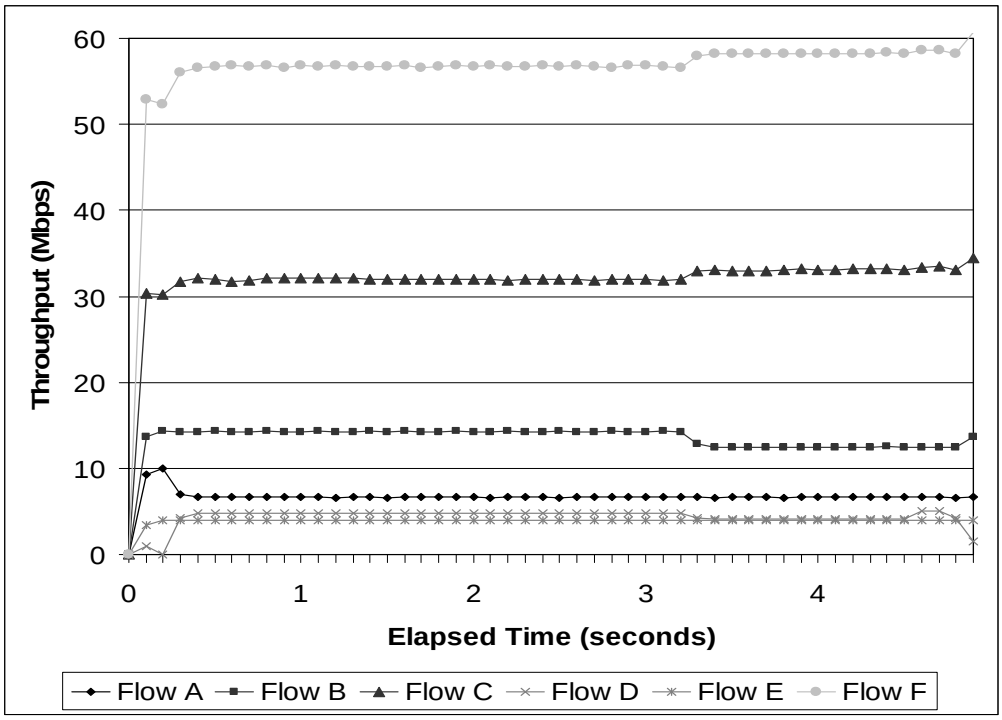

Figure 34 - GFC1 Unconstrained Throughput

Using the same network configuration, this simulation was run with all LSPs configured to use Pareto on/off sources. Three measurements were made, one with TERM, one using the initial approach to congestion control, and one on an unconstrained legacy TCP network. The TERM network had an aggregate throughput of 286 Mbps, which was 27% more than the 225 Mbps throughput accomplished using the initial approach, and 20% greater than the throughput of the legacy TCP network. The increase in throughput was in addition to the enforcement of fair rates in the TERM system.

4.5.2 TCP/UDP Weighted Fair Rates

This simulation was designed to show that TCP and UDP sources will converge to fair rates, and that this convergence will happen even in the event that there are queue

collisions between TCP and UDP sources. Table 6 summarizes the number of LSPs, layer 4 IP protocol, queue, weight, and fair rate for the LSPs in Figure 35. In this simulation, sources A and C are wired to collide to the same WPDRR queue.

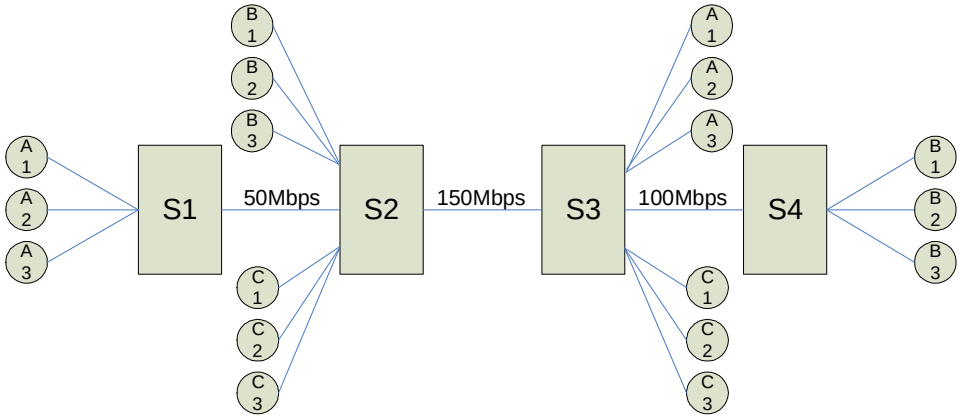

Figure 35 – Queue Collision Configuration

	LSPs	Type	Queue	Weight	Fair Rate
A	3	TCP	1	3	33.33
B	3	TCP	2	2	16.66
C	3	UDP	1	1	8.33

Table 6 – Queue Collision LSPs

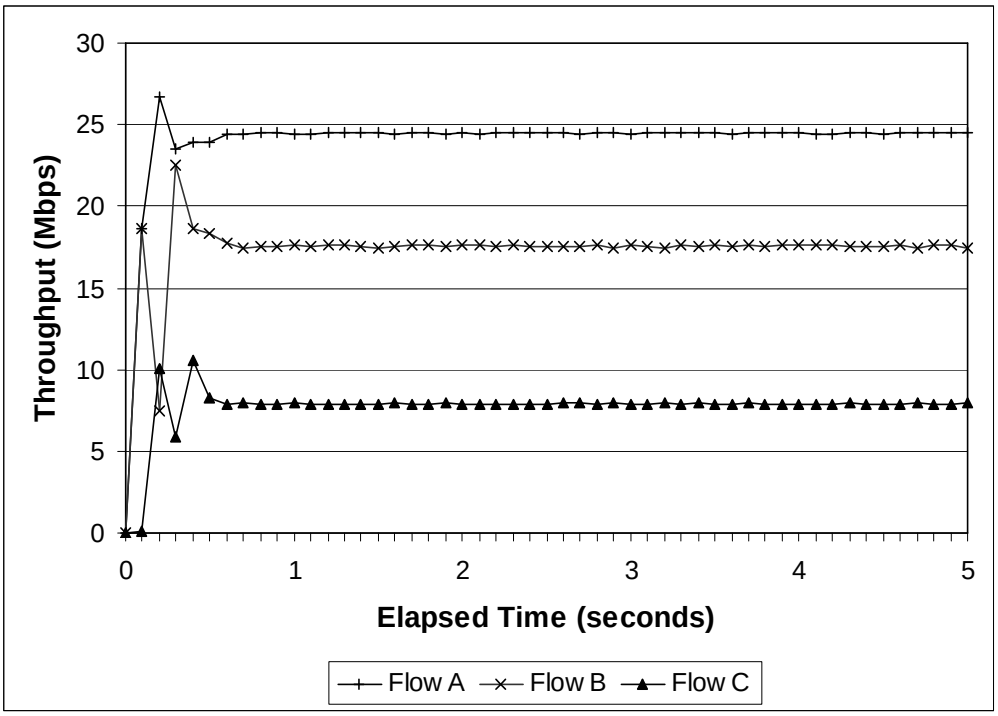

Figure 36 – Queue Collision Throughput

The results in Figure 36 show that the sources converge to weighted fair rates, and that the weighted fair rates between responsive traffic (TCP), and unresponsive traffic (UDP) is managed even in the event of queue collisions.

4.5.3 Cooperating Sources

Figure 23 shows a network with two LSPs traversing a single link of 50 Mbps. In this test scenario, both LSPs are configured to transmit UDP traffic at a rate of 100 Mbps. LSP1 (Flow A) is given a weight of 1 while LSP2 (Flow B) is given a weight of 2. The source for LSP2 transmits at a constant rate while the source for LSP1 uses a Pareto On/Off source that switches the transmission rate between 0 Mbps and 100 Mbps at time intervals determined by a Pareto distribution.

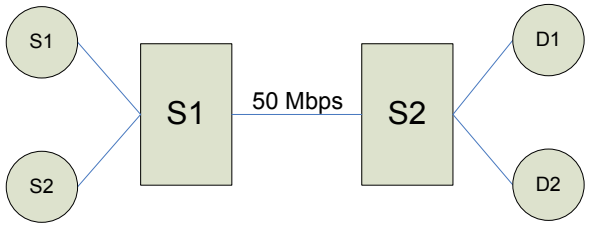

Figure 37 – Cooperating Source Configuration

With link capacity of 50 Mbps available to Best Effort traffic, the explicit rate algorithm grants LSP1 (weight 1), and LSP2 (weight 2) fair rates of 16.66 Mbps and 33.33 Mbps respectively. Since LSP2 is transmitting at a constant rate, it will always use at least its fair rate of 33.33 Mbps. Additionally, in the periods in which LSP1 is not using its capacity (off periods), LSP2 will make use of the available capacity by sending LBE traffic. Figure 38 shows that the results of this simulation are consistent with our expectation.

Figure 39 shows the above scenario with the source for LSP2 changed from constant bit rate Pareto On/Off in a manner identical to LSP1. Here we can clearly see the two LSPs cooperating. When both are transmitting, each is confined to its fair rate. If either is off while the other is transmitting, the active LSP will make use of the available bandwidth by sending LBE traffic.

Figure 40 shows this scenario with the configuration changed to only allow LSPs at a maximum rate equal to its fair rate. By applying the intelligent marking and queuing mechanism, link utilization is increased by 17% in this simple configuration.

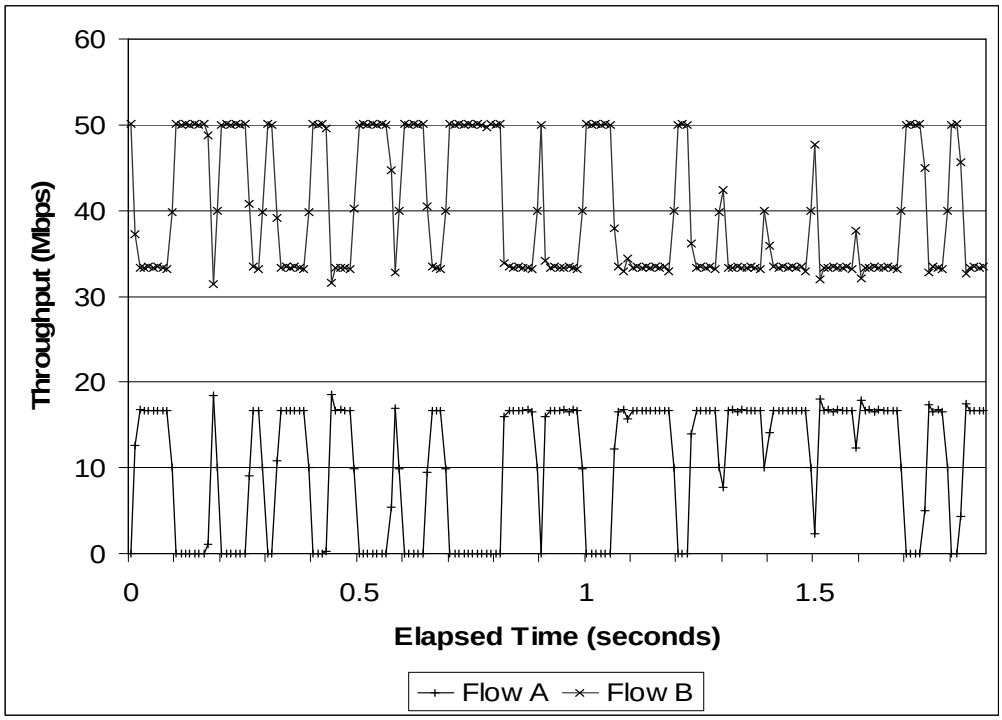

Figure 38 – Cooperating Source Throughput

In a more appropriate scenario, we performed the same experiment on the GFC1 configuration pictured in Figure 14. The simulation setup was changed to make all LSPs transmit traffic in a Pareto On/Off pattern. Two measurements were made: one where the LSPs were allowed to transmit at a rate up to its assigned fair rate as in the initial implementation, and another where LSPs could use LBE traffic to transmit traffic at a rate above their fair rate. We found that in this configuration, aggregate network throughput was increased by 44% while still enforcing fair rates.

Through this simulation it is clear that the issues affecting cooperation between sources identified in the initial thesis approach of Section 3.4.8 has been remedied by the introduction of fair rate marking and queuing.

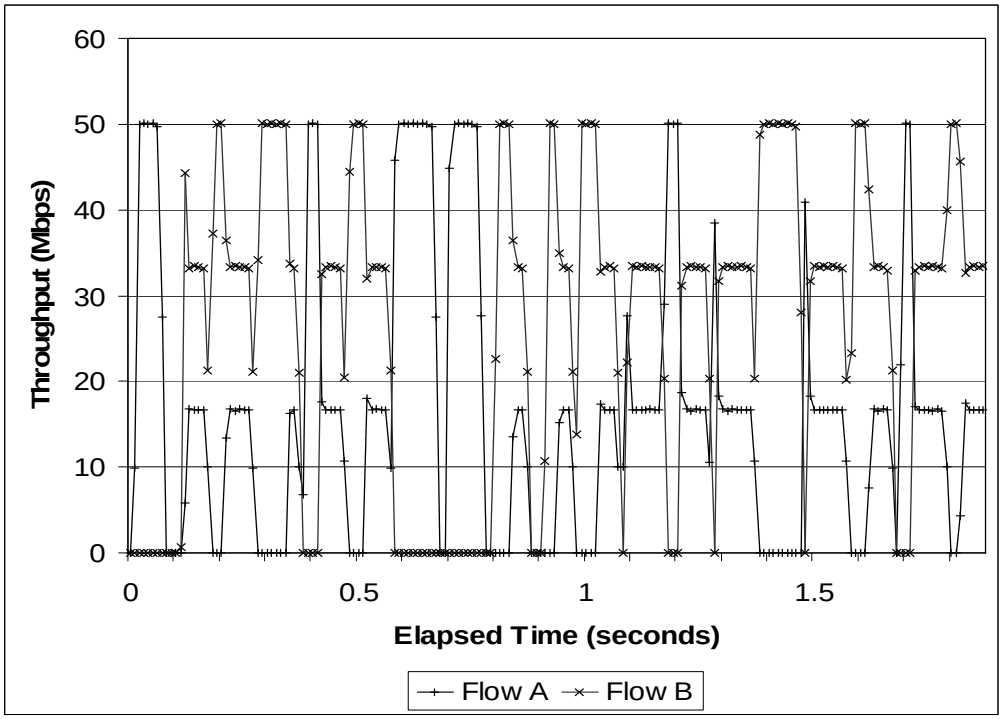

Figure 39 – Two Pareto Source Throughput

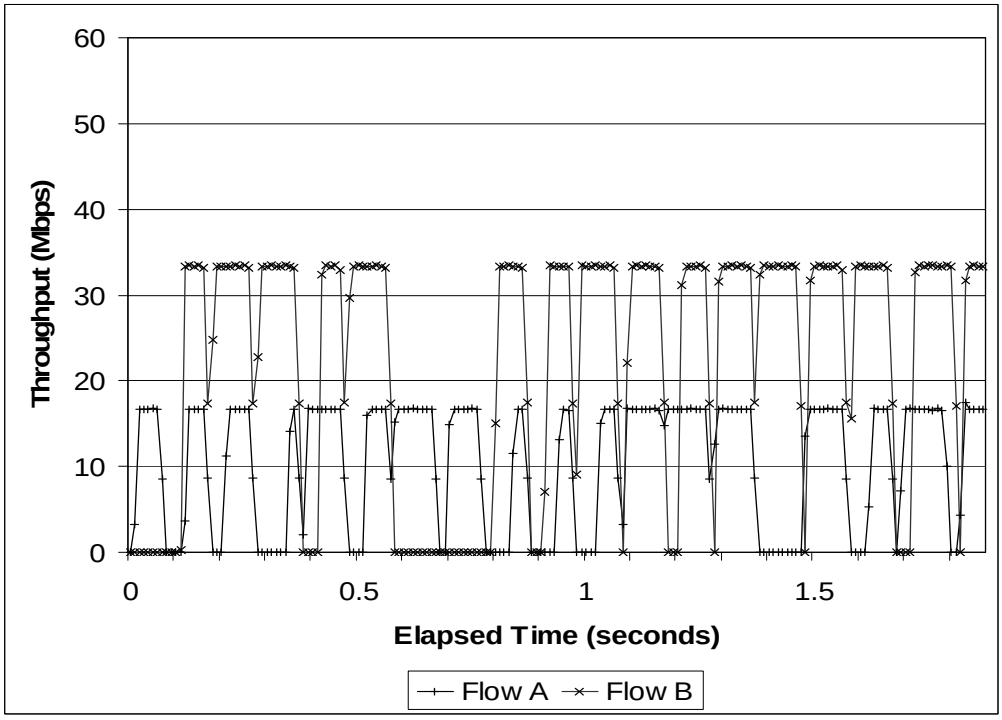

Figure 40 – No Active Queuing Throughput

4.5.4 Large-scale Network

To judge the performance of the TERM system in a large-scale network deployment, we simulated TERM using the topology of the AT&T domestic IP network as reported in the Rocketfuel database [10]. This network consists of 115 nodes connected by 296 links geographically dispersed across the United States. Bandwidth of links was assigned by scaling the route weights for the appropriate links, and propagation delay was implemented as reported in Rocketfuel. 1000 LSPs were created and randomly assigned source and destinations across the 115 network endpoints. 2/3 of sources were TCP, 1/3 were UDP. Each source had a constant peak rate of 100 Mbps and a weight of 1.

At the beginning of the simulation, LSPs were established across the network. The routes for LSPs were determined using the Shortest Path First algorithm in OSPF [11]. Upon calculating routes, the simulation was started and the WPMM Distributed Fair Rate algorithm was used to determine fair rates. Once fair rates were determined for our topology, it was determined that 132 LSPs were satisfied with a fair rate of 100 Mbps, while the remaining 868 LSPs were bottlenecked at some link in the network. By summing the calculated fair rate values for each LSP in the network, a theoretical maximum throughput of 70.7 Gbps was established. The performance of the TERM system is benchmarked against an identical network configuration using no explicit rate congestion control and DRR queuing.

Parameter	Value
Number of Nodes	115
Number of Links	296
Number of LSPs	1000
TCP LSPs	667
UDP LSPs	333
Satisfied LSPs	132
Bottlenecked LSPs	868
LSP Peak Rate	100 Mbps
Theoretical Max Throughput	70.7 Gbps
LSP Weight	1

Table 7 - Large-scale Network Parameters

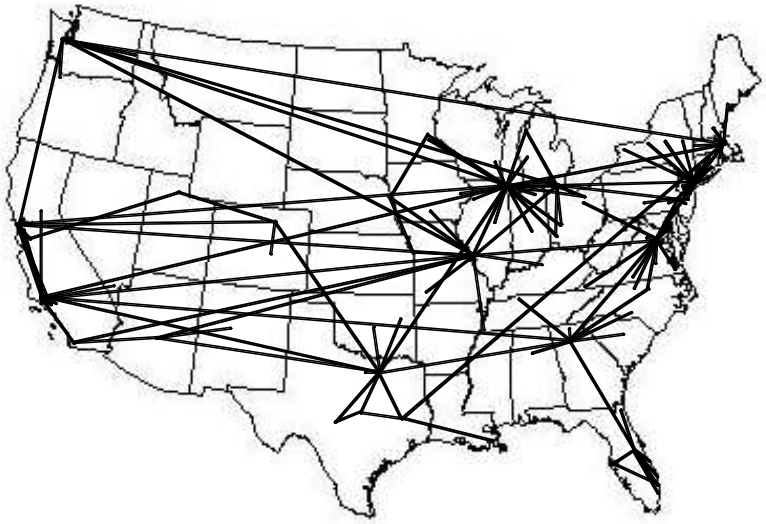

Figure 41 - AT&T Domestic IP Network

4.5.4.1 Constant Bit Rate Scenario

In this scenario, all sources were set to transmit at a constant bit rate. The goal of the scenario is to show that, in the steady state, the TERM system works to enforce fair rates while increasing network throughput.

To judge efficiency, a measurement was taken that summed the average throughput of all LSPs in the network; this is referred to as aggregate throughput. In the DRR scenario, the LSPs had an aggregate throughput of 65.4 Gbps while in the TERM scenario, LSPs had an aggregate throughput of 69.3 Gbps.

In each of the scenarios, the actual throughput of each LSP was compared against the theoretical fair rate for the LSP. Figure 42 graphs the percent deviation from fair rates for each of the 1000 LSPs in the simulation sorted from lowest to highest. In the ideal case, all LSPs would have throughput equal to their assigned fair rate, and the network would be 100% efficient. In this ideal case, the profile in Figure 42 would be a straight line at x = 0%. This would indicate that each LSP exhibited throughput that was exactly its fair rate. Note that measurements are relative to the fair rate, thus an LSP allocated a fair rate of 1Mbps with actual throughput of 800 Mbps is weighted identically to an LSP granted 100 Mbps with actual throughput of 80 Mbps.

Figure 42 shows the profiles for both the TERM simulation, and the identical simulation performed on the Benchmark network. From this graph, it is clear that the TERM system exhibits a much flatter profile than the benchmark. In this scenario LSPs in the TERM system were granted an average of 4% less than the ideal fair rate with a standard deviation of just 6%.

The Benchmark network exhibited a profile that was far from ideal. While LSPs in the Benchmark network were granted an average of 14% less than their fair rate, the standard deviation was 84% indicating that there was significant disparity in the distribution. The figure clearly shows that in the Benchmark network, very LSPs had throughput that matched fair rates.

With TERM 690 LSPs had throughput within +/- 5% of the assigned fair rates. This is compared with only 64 LSPs within +/- 5% for the Benchmark network. The enforcing of fair rates was accomplished with no loss of network throughput. In fact, the actual throughput of the TERM network was 6% greater than the throughput in the Benchmark network. These results are summarized in Table 8.

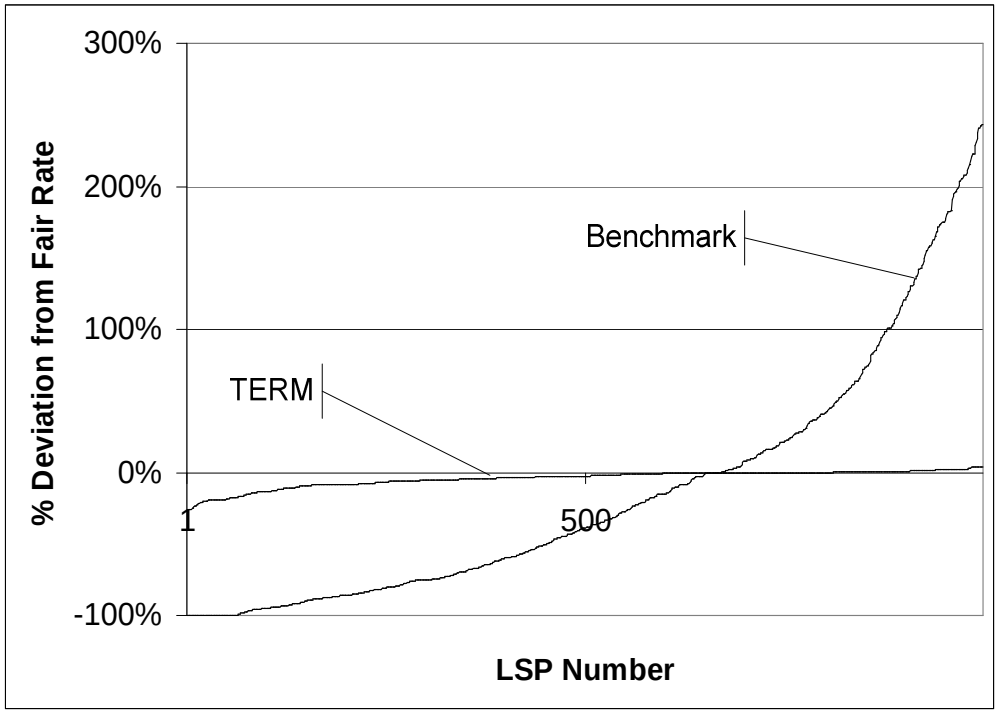

Figure 42 - CBR LSP % Deviation from Fair Rates

Parameter	TERM	DRR
Theoretical Maximum Throughput	70.7 Gbps	70.7 Gbps
Actual Throughput	69.3 Gbps	65.4 Gbps
Efficiency	98%	92%
LSPs within 5% of FR	690	64
Fair Rate Std Dev	6%	84%

Table 8 - Large-scale Network CBR Results

To demonstrate the effectiveness of TERM over the initial thesis approach with constant bit rate traffic, the identical scenario was run with using the initial congestion control system. It was found that the throughput of this scenario under the initial congestion control approach was only 59.4 Gbps. The TERM approach increased the throughput over the initial approach by 17% while enforcing fair rates on TCP and UDP sources.

4.5.4.2 Variable Bit Rate Scenario

The second simulation scenario is designed to more closely match actual network conditions. The network and LSPs were configured in a manner identical to the above simulation with the only difference being that, rather than transmitting at constant rate, each LSP was configured to transmit using an on/off pattern. The on and off durations were generated using Pareto distribution to simulate self-similar network traffic.

The deviations from fair rates for TERM and the Benchmark network are shown in Figure 43. In this simulation, the TERM system again showed a far more favorable convergence to fair rates with the average LSP throughput of 8% less than the ideal fair rate with a standard deviation of 15%.

The Benchmark network had an average LSP throughput of 12% less than fair rates with a standard deviation of 80%. In this scenario, 333 TERM LSPs were within +/- 5% of fair rates, while only 43 LSPs in the benchmark network were within that range. Throughput for the TERM network was 15% greater than the Benchmark network showing that, in addition to enforcing fair rates, the TERM system has a strong positive effect on network throughput.

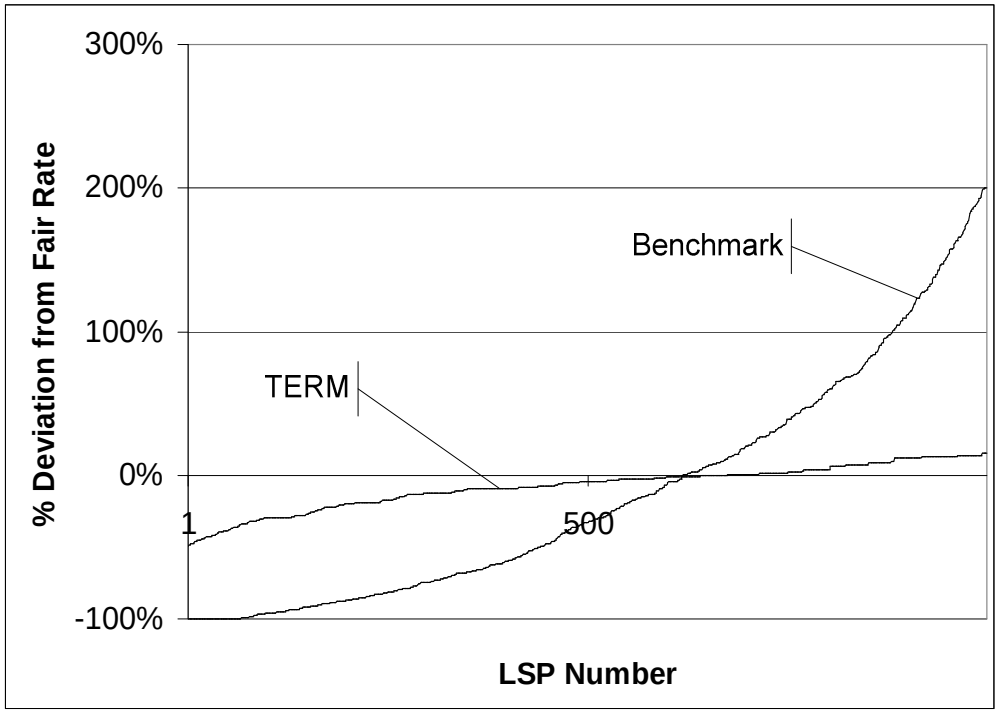

Figure 43 - VBR LSP % Deviation from Fair Rates

Parameter	TERM	DRR
Theoretical Maximum Throughput	70.7 Gbps	70.7 Gbps
Actual Throughput	60.8 Gbps	52.8 Gbps
Network Utilization	86%	75%
LSPs within 5% of FR	333	43
Fair Rate Std Dev	15%	80%

Table 9 - Large-scale Network VBR Results

To demonstrate the effectiveness of TERM over the initial thesis approach with variable bit rate traffic, the identical scenario was run with using the initial congestion control system. It was found that the throughput of this scenario under the initial congestion control approach was 45.4 Gbps. The TERM approach increased the throughput over the initial approach by 34% while enforcing fair rates on TCP and UDP sources.

Chapter 5 - Quality of Service (QoS) Model

In this section we explore a mechanism for integrating explicit rate best effort traffic into a framework that will cover best effort as well as varying degrees of real-time, guaranteed bandwidth services.

5.1 Class Based Queuing

CBQ is a powerful and flexible mechanism for representing hierarchical relationships between traffic classes. This hierarchy describes relationships that define how excess bandwidth is shared between the classes. We describe enough detail about CBQ to properly explain our hierarchy. A comprehensive description of CBQ is included in [1].

We use Class Based Queuing (CBQ) to implement the class structure necessary to properly represent a QoS model that includes real time services with the LBE and BBE services available through the TERM system.

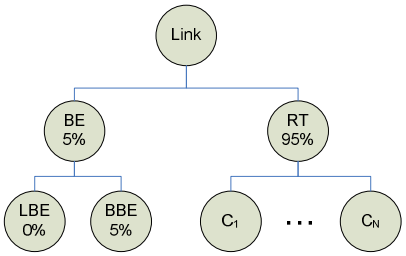

Figure 44 - CBQ Hierarchy

Figure 44 shows a CBQ hierarchy that represents our current model of Real Time (RT) and Best Effort (BE) Traffic. Each of the leaf nodes in the hierarchy is assigned a percent link allocation and a priority. The percent link allocation is the portion of the link capacity that is dedicated to a particular class.

As per CBQ definitions, a class is considered over-limit if it is using more than its percent link allocation, under-limit if it is using less, and at-limit otherwise. A class is considered unsatisfied if it is under-limit and has a persistent backlog; otherwise it is considered satisfied.

Classes can be either regulated or un-regulated. Un-regulated classes can transmit data at an unconstrained rate; regulated classes have their rate constrained by the scheduler. Classes can transmit data un-regulated if either:

1 – The class is under-limit, OR

2 – The class has an under-limit ancestor at level *i*, and there are no unsatisfied classes in the hierarchy at level lower than *i*.

If neither of the above two conditions are met, the class is regulated. Queuing for regulated classes is done on a strict priority basis meaning that queues of higher priority classes are completely drained before servicing lower priority class queues. In our model,

we assign a priority of zero to LBE traffic and a priority of one to BBE traffic. Real time classes $C_1 \ldots C_N$ are assigned a priority greater than one.

5.2 QoS Model

For simplicity, we will first explain a system with only BE traffic. We will then extend this to handle RT traffic.

In the absence of RT traffic the full link bandwidth will be dedicated to BE traffic, thus B(t) (the bandwidth available to BE traffic) will be the link capacity. The TERM signaling mechanism and TERM distributed explicit rate algorithm will assign explicit rates to BE LSPs. Using the estimator detailed above, traffic up to the explicit rate will be marked BBE and traffic exceeding the explicit rate will be marked LBE. Should an LSP use less than its explicit rate, the additional bandwidth will be available for LBE traffic. Given the lower priority, LBE packets will be dropped before BBE packets. Note that, strictly speaking, the nodes in the CBQ tree represent unique queues. While the drop priority of the packets determines the traffic class, the queue implementation must be such that re-ordering of packets does not occur, as there will be both LBE and BBE packets in the same flow.

RT traffic may make guaranteed rate reservations to the extent that the reservations on a link don’t exceed the percent allocation of the RT node (in our example 95%). Reserved bandwidth will be subtracted from the bandwidth available to BE traffic (B(t)). As per the CBQ hierarchy, bandwidth reserved by a RT LSP but not used will be available to BE traffic. Since the explicit rates allocated to BBE traffic are based on B(t), and B(t) is a function of bandwidth reservations, traffic not used by RT LSPs will be used by the LBE traffic class. In this model, RT LSPs transmit at their reserved rate or less.

5.3 Example Implementation

Though network operators will require the flexibility to determine QoS service levels and parameters, here we outline a sample implementation of the TERM QoS system using CBQ. We construct a CBQ Hierarchy and use this hierarchy to enforce both queuing priorities and bandwidth reservation bounds.

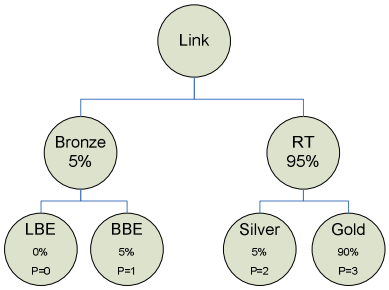

Figure 45 - TERM QoS Implementation

Figure 45 shows an implementation of a CBQ hierarchy for a typical implementation of TERM QoS. In this implementation, there are three levels of service, Gold, Silver, and Bronze. The Gold and Silver services are real time, and the Bronze service is Best Effort using TERM congestion control.

In this implementation, the Gold service is given the highest priority, Silver second priority, Bronze/BBE third priority, and Bronze/LBE the lowest priority. Gold is allocated an allocation of 90% of bandwidth, and Silver an allocation of 5%. Both Gold and Silver are children of the Real Time node, which is given an allocation of 95%. Bronze service is given an allocation of 5%, with its children BBE and LBE allocations of 5% and 0% respectively.

This CBQ can be used to enforce the following QoS constraints.

- Gold and Silver services combined can reserve and use up to 95% of the link bandwidth
- In this scenario Gold and Silver services may have a minimum allocation of 90% and 5% of link bandwidth, though in the case where one is not using its allocation, the other can share the bandwidth
- In times of network contention, Gold traffic will be queued before Silver traffic
- Bronze service gets a minimum of 5% of the link bandwidth, though it can use bandwidth not used by either the Silver or Gold services
- In times of contention, BBE service will be queued after the Gold or Silver service, but before the LBE service.

Chapter 6 - Conclusions

6.1 Summary of Results

This paper introduced TERM. TERM includes a novel approach for signaling explicit rate congestion control information using RSVP, a distributed WPMM fair rate algorithm, an estimator for marking packet drop priority, and a queue management system for enforcing explicit rates based on drop priorities. Through simulations, we've shown that the TERM throughput converges to weighted fair rates in non-trivial situations with both TCP and UDP traffic. Simulations have also shown significant increase in network throughput using the TERM system.

In addition to the base TERM components, we have presented a method for using Class Based Queuing to develop a system for QoS that includes TERM best effort service as well as varying degrees of real-time guaranteed bandwidth offerings.

6.2 Future Research Opportunities

While this work has thoroughly addressed the area of providing explicit rate congestion control for MPLS networks, there exist opportunities to extend the design to include a number of different areas.

One significant extension of this work would be to use the fair rates and weights traversing links in the network to feed a constraint routing algorithm. This would enable maximizing the utility of the network based not only on maximizing throughput, but also on maximizing the utility based on weights. Determining an appropriate metric from the TERM WPMM Fair Rate algorithm would enable the use of various constraint routing techniques. This would enable LSP setup in a manner consistent with the work done in TERM.

Another area for future exploration is extending the boundaries of the max-min calculation to include resources outside the core network. Examples of these resources include stub networks and grid resources. A framework for extending the boundaries of a max-min domain was explored in detail in [1]. This work could be extended to fit the TERM model.

Bibliography

Chapter 1 – Introduction

[1] V. Cerf, Y. Dalal, C. Sunshine, “Specification of the Internet Transmission Control Program,” RFC 675, December 1974, http://www.ietf.org

[2] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture,” RFC 3031, January 2001, http://www.ietf.org

[3] The ATM Forum Technical Committee, “Traffic Management Specification version 4.0,” af-tm-0056.000, April 1996, http://www.atmforum.com

[4] T. Ott and N. Aggarwal, "TCP over ATM: ABR or UBR?," Proceedings of the 1997 ACM SIGMETRICS, Seattle, WA, pp. 52-63, June 1997

[5] S. Manthorpe, Jean-Yves Le Boudec, “A comparison of ABR and UBR to support TCP traffic,” Technical report Laboratoire de Réseaux de Communication, EPFL, 1997

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource Reservation Protocol (RSVP),” RFC 2205, September 1997, http://www.ietf.org

[7] Y. Thomas Hou, Henry Tzeng, Shivendra S. Panwar, Vijay P. Kumar, “A Generic Weight-Proportional Bandwidth Sharing Policy for ATM ABR Service,” Performance Evaluation, Vol 38, 1999, pp. 21-44

[8] C. Marty, M. A. Ali, “A System for Weighted Max-Min Congestion Control in MPLS Networks,” Journal of Network and Systems Management, Vol. 14, No. 4, December 2006

[9] C. Marty, M. A. Ali, “TERM: TCP Friendly Explicit Rate Congestion Control for MPLS Networks,” IEEE GLOBECOM 2006, San Francisco, CA, December 2006 (To Appear)

[10] C. Marty, M. A. Ali, “An Integrated Framework for Managing Best Effort Traffic in MPLS Networks,” Proc. of IEEE INFOCOM Global Internet Symposium, Barcelona, Spain, April 2006

[11] S. Floyd, and V. Jacobson, “Link-sharing and Resource Management Models for Packet Networks,” IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, August 1995

Chapter 2 – Background Information

[1] V. Cerf, Y. Dalal, C. Sunshine, “Specification of the Internet Transmission Control Program,” RFC 675, December 1974, http://www.ietf.org

[2] IEEE Standard for Information Technology 802.3 “Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,” December 2005, http://standards.ieee.org

[3] IEEE Standard for Information Technology 802.11a, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” June 2003, http://standards.ieee.org

[4] IEEE Standard for Information Technology 802.16, “LAN/MAN Broadband Wireless LANS,” February 2004, http://standards.ieee.org

[5] American National Standards Institute, "Synchronous Optical Network (SONET) - Basic Description including Multiplex Structure, Rates and Formats," ANSI T1.105-1995

[6] The ATM Forum Technical Committee, “Traffic Management Specification version 4.0,” af-tm-0056.000, April 1996, http://www.atmforum.com

[7] American National Standards Institute, “Frame Relaying Bearer Service---Architectural Framework and Service Description,” ANSI T1.606, 1990, http://www.ansi.org

[8] D. Plummer, “Ethernet Address Resolution Protocol,” RFC 826, November 1982, http://www.ietf.org

[9] R. Finlayson, T. Mann, J. Mogul, M. Theimer, “A Reverse Address Resolution Protocol,” RFC 902, June 1984, http://www.ietf.org

[10] J. Postel, “Transmission Control Protocol – DARPA Internet Program Protocol Specification,” RFC 793, September 1981, http://www.ietf.org

[11] J. Postel, “User Datagram Protocol,” RFC 768, August 1980, http://www.ietf.org

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “Hypertext Transfer Protocol – HTTP/1.1,” RFC 2616, June 1999, http://www.ietf.org

[13] J. Postel, “Simple Mail Transfer Protocol,” RFC 821, August 1982, http://www.ietf.org

[14] J. Moy, “OSPF Version 2,” RFC 2328, April 1998, http://www.ietf.org

[15] Y. Rekhter, T. Li, “A Border Gateway Protocol 4,” RFC 1771, March 1995, http://www.ietf.org

[16] V. Jacobson, “Congestion control and avoidance,” in Proc. ACM SIG-COMM Symp. Communications Architectures and Protocols, 1988, pp. 314–329

[17] S. Floyd, T. Henderson, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” RFC 2582, April 1999, http://www.ietf.org

[18] A. Medina, M. Allman, S. Floyd, “Measuring the Evolution of Transport Protocols in the Internet,” Computer Communications Review, April 2005, pp. 37-52

[19] ATM Forum, http://www.atmforum.com

[20] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture,” RFC 3031, January 2001, http://www.ietf.org

[21] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource Reservation Protocol (RSVP),” RFC 2205, September 1997, http://www.ietf.org

[22] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, “RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, December 2001, http://www.ietf.org

[23] L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, “LDP Specification,” RFC 3036, January 2001, http://www.ietf.org

[24] J. Ash, Y. Lee, P. Ashwood-Smith, B. Jamoussi, D. Fedyk, D. Skalecki, L. Li, "LSP Modifications using CR-LDP," RFC 3214, January 2002, http://www.ietf.org

[25] T. Nadeau, C. Srinivasan, A. Viswanathan, "Multiprotocol Label Switching (MPLS) Forwarding Equivalence Class To Next Hop Label Forwarding Entry (FEC-To-NHLFE) Management Information Base (MIB)," RFC 3814, June 2004, http://www.ietf.org

Chapter 3 – Initial Approach

[1] R. Jain, "Congestion control and traffic management in ATM networks: recent advances and a survey," ATM Forum Contribution 95-0177, 1995

[2] J. Aweya, M. Ouellette, D.Y. Montuno, "Design and Stability Analysis of a Rate Control Algorithms Using the Routh-Hurwitz Stability Criterion," IEEE/ACM Transactions on Networking, Vol 12, No. 4, August 2004, pp. 719-732

[3] The ATM Forum Technical Committee, "Traffic Management Specification version 4.0," af-tm-0056.000, April 1996, http://www.atmforum.com

[4] S. Kalyanaraman, "Traffic Management for the Available Bit Rate (ABR) Service in Asynchronous Transfer Mode (ATM) Networks," Ph.D. Thesis, Ohio State University, Chapter 4, 1997

[5] A. Charny, D. Clark, R. Jain, "Congestion Control with Explicit Rate Indiciation", Proc. IEEE ICC'95, June 1995, pp. 1954-1963

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource Reservation Protocol (RSVP)," RFC 2205, September 1997, http://www.ietf.org

[7] J. Wroclawski, "Specification of the Controlled Load Quality of Service," RFC 2211, September 1997, http://www.ietf.org

[8] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, December 2001, http://www.ietf.org

[9] C. Marty, M. A. Ali, "A System for Weighted Max-Min Congestion Control in MPLS Networks," Journal of Network and Systems Management, Vol. 14, No. 4, December 2006

[10] S. Shenker, S. Partridge, R. Guerin, "Specification of the Guaranteed Quality of Service," RFC 2212, September 1997, http://www.ietf.org

[11] D. P. Bertserkas, R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, N.J., 1992

[12] Y. Thomas Hou, Henry Tzeng, Shivendra S. Panwar, Vijay P. Kumar, “A Generic Weight-Proportional Bandwidth Sharing Policy for ATM ABR Service,” Performance Evaluation, Vol 38, 1999, pp. 21-44

[13] F. Skivée, G. Leduc, “A Distributed Algorithm for Weighted Max-Min Fairness in MPLS Networks,” Proc. of 11th IEEE International Conference on Telecommunications (ICT'2004), Fortaleza, Brazil, 2004

[14] Quan Le T. and Tapio Erke, “A Combination of ERICA+ and Distributed WPMM Algorithms to Guarantee the Minimum Cell Rate for ABR Service in ATM Networks.” Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems L. (SPECTS), Montreal, Canada, July 2003

[15] L. Kalampoukas, A. Varma, K.K. Ramakrishnan, “An Efficient Rate Allocation Algorithm for ATM Networks Providing Max-Min Fairness,” Proceedings of the 6th IFIP International Conference on High Performance Networking, 1995

[16] Wangdong Qi, Xiren Xie, "The structural property of network bottlenecks in maxmin fair flow control," IEEE Communications Letters, pp.76-77, March 1998

[17] W.K. Tsai, M. Iyer, "Constraint precedence in max-min fair rate allocation," IEEE International Conference on Communications (ICC), pp.490-494, 2000

[18] The Network Simulator Version 2.28 (NS2), http://www.isi.edu/nsna

[19] The MPLS Network Simulator version 2, http://flower.ce.cnu.ac.kr/~fogl/nms

[20] D. Adami, C. Callefari, S. Giordano, F. Mustacchio, M. Pagno, F. Vitucci, “Overview of the RSVP-TE Network Simulator: Design and Implementation,” Proc. of Internet Performance, Simulation, Monitoring and Measurements (IPS-MOME), pp. 276-281, March 2005

[21] M. Greis, The RSVP Module for NS2, http://titan.cs.unibonn.de/~greis/

[22] RSVP-TE Patch for NS2, http://www.ideo-labs.com

[23] C. Williamson, “A Performance Study of ABR Traffic Control Strategies Using the ATM-TN Simulator”, TeleSim Project Report, Department of Computer Science, University of Saskatchewan, April 1998

[24] R. Jain, S. Fahmy, S. Kalyanaraman, R. Goyal, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks, Part II: Requirements and

Performance Evaluation", Paper submitted to the IEEE/ACM Transactions on Networking, January 1997

Chapter 4 – TCP Friendly Congestion Control

[1] C. Marty, M. A. Ali, “TERM: TCP Friendly Explicit Rate Congestion Control for MPLS Networks,” IEEE GLOBECOM 2006, San Francisco, CA, December 2006 (To Appear)

[2] C. Marty, M. A. Ali, “An Integrated Framework for Managing Best Effort Traffic in MPLS Networks,” Proc. of IEEE INFOCOM Global Internet Symposium, Barcelona, Spain, April 2006

[3] C. Marty, M. A. Ali, “RSVP Extensions for Explicit Rate Congestion Control in Multiprotocol Label Switched (MPLS) Networks,” IETF Internet Draft draft-marty-mpls-rsvp-er-00.txt, March 2006, http://www.ietf.org

[4] Floyd, S.; Jacobson, V., "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, Volume 1, Issue 4, pp. 397-413, Aug. 1993

[5] E. Hahne and R. Gallager, "Round Robin Scheduling for Fair Flow Control in Data Communication Networks," Tech. Rep. LIDS-TH-1631, LIDS, MIT, Dec. 1986

[6] M. Shreedhar, G. Varghese, “Efficient fair queuing using deficit round-robin,” IEEE/ACM Transactions on Networking, Volume 4, Issue 3, pp. 375 – 385, June 1996

[7] Marbach, P., “Priority service and max-min fairness,” IEEE/ACM Transactions on Networking, Vol. 11, Issue 5, pp. 733 – 746, Oct. 2003

[8] The ATM Forum Technical Committee, “Traffic Management Specification version 4.0,” af-tm-0056.000, April 1996

[9] R. Jain, “Congestion control and traffic management in ATM networks: recent advances and a survey,” ATM Forum Contribution 95-0177, 1995

[10] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP Topologies with Rocketfuel,” Proceedings of ACM/SIGCOMM '02, August 2002

[11] J. Moy, “OSPF Version 2,” RFC 2328, April 1998, http://www.ietf.org

Chapter 5 – Quality of Service (QoS) Model

[1] S. Floyd, and V. Jacobson, “Link-sharing and Resource Management Models for Packet Networks,” IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, August 1995

Chapter 6 – Conclusion

[1] Y. Zhou, and H. Sethu, “On Achieving Fairness in the Joint Allocation of Processing and Bandwidth Resources: Principles and Algorithms,” IEEE/ACM Transactions on Networking, Vol. 13 No. 5, pp. 1054-1067, October 2005