MULTIPATH AND EXPLICIT RATE CONGESTION CONTROL ON DATA NETWORKS

by

Soonyong Sohn
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Brian L. Mark, Dissertation Director
_____	Dr. Bijan Jabbari, Committee Member
_____	Dr. Duminda Wijesekera, Committee Member
_____	Dr. Songqing Chen, Committee Member
_____	Dr. Daniel Menasce, Senior Associate Dean
_____	Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: _____ Spring Semester 2010
George Mason University
Fairfax, VA

Multipath and Explicit Rate Congestion Control on Data Networks

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Soonyong Sohn
Master of Science
George Mason University, 2002
Bachelor of Science
Hankuk Aviation University, 1996

Director: Dr. Brian L. Mark, Professor
Department of Information Technology

Spring Semester 2010
George Mason University
Fairfax, VA

Dedication

This is dedicated to my wife, Gunna Yun.

Acknowledgments

I give a very special thanks to my academic advisor, Dr. Brian L. Mark, and committee members, Drs. Bijan Jabbari, Duminda Wijesekera, and Songqing Chen for their guidance, caring, and support. Finally and most importantly, I thank my wife, Gunna Yun, for her ongoing encouragement, mentoring, and love.

Table of Contents

List of Tables

List of Figures

Abstract

MULTIPATH AND EXPLICIT RATE CONGESTION CONTROL ON DATA NETWORKS

Soonyong Sohn, PhD

George Mason University, 2010

Dissertation Director: Dr. Brian L. Mark

Computer networks based on the TCP/IP (Transmission Control Protocol/Internet Protocol) employ TCP congestion control and shortest path routing. However, TCP congestion control can result in under-utilization of link capacity, low session throughput, and unfairness in its throughput performance over impaired links. Conventional shortest path routing can lead to network congestion and under-utilized links due to uneven distribution of traffic in the network.

To address these problems, this thesis proposes multipath congestion control algorithms for data networks, which combine multipath routing with network congestion control. First, an efficient multipath route discovery algorithm is proposed to find multiple paths in the underlying network infrastructure. The multipath route discovery algorithm can find multipath routes with varying degrees of disjointedness. Second, we develop multipath traffic distribution algorithm to alleviate network congestion by exploiting multipath routes. The proposed "congestion-triggered multipath protocol" requires relatively minor upgrades to the existing Internet architecture.

Recently, there have been proposals to introduce explicit rate signaling into the Internet. Explicit rate signaling has the potential to substantially improve network performance, but requires routers that can support signaling on a per-flow basis. Along these lines, we propose an adaptive dynamic rate controller that computes the rate for flows in response to network status (e.g., network congestion, link underutilization) in order to minimize network congestion and fully utilize the link capacity. We evaluate its performance in conjunction with a rate-based transport protocol.

Chapter 1: Introduction

The Internet was originally designed to provide a single level of service (i.e., best-effort) in which the network does not provide any performance guarantees on data delivery or on the Quality-of-Service (QoS) level. Most computer networks are designed to provide best-effort service and employ

1. shortest path routing protocols that provide a shortest path between source and destination pairs, and

2. end-to-end congestion control mechanisms that react to congestion by adapting the sender transmission rate based on indirect network congestion indications, e.g., packet loss [2].

The route between source and destination is typically determined by means of shortest path routing protocols such as Routing Information Protocol (RIP) [3], Open Shortest Path First (OSPF) [4], and Border Gateway Protocol (BGP) [5]. In these routing protocols, the path length is typically taken as the hop count, i.e., the link cost is taken to be unity by default. Another link metric that is often used in conjunction with shortest path routing algorithms is link capacity. Under shortest path routing, all packets associated with a given source-destination pair generally traverse a single path of shortest length, even though other paths may be available. Consequently, shortest path routing can lead to network congestion due to excessive traffic routed on a single path, even though other paths may be under-utilized.

Multipath routing has been proposed as a means to alleviate the limitations of shortest path routing algorithm. In multipath routing, packets belonging to a given source-destination pair may be transmitted over multiple paths. Some of the potential benefits of

multipath routing include load balancing [6], higher network throughput [6,7], reduction of routing oscillation, the alleviation of congestion [8,9], and improved packet delivery reliability [10]. In prior work dating to the 1950's, the K shortest paths problem has been studied in a number of papers (e.g., [11–20]). The K shortest paths problem is to determine the K shortest paths (i.e., the shortest path, the second shortest path, ..., the K-th shortest path) between the given pair of nodes. However, K shortest paths problem may not be ideally suited to controlling congestion over multiple paths.

Equal-Cost MultiPath (ECMP) [4] and Optimized Multipath (OMP) [21] are protocols that have been proposed to implement multipath routing in the current Internet. ECMP finds equal cost multipath routes (i.e., multiple paths with equal number of hops or equal metric) and distributes traffic equally over a multipath route. OMP is an improved version of ECMP, which allows unequal traffic distribution (i.e., distribution in inverse proportion to the utilizations of the constituent paths). The paths in the multipath route need not be of equal length. The set of paths in a multipath route includes the shortest path, obtained from the shortest path routing protocol, plus alternative paths derived from shortest paths from each of the neighbors of the source node to the destination node.

Transmission Control Protocol (TCP) [22] has been a dominant protocol for end-to-end congestion control in Internet (see [23]). The current implementation of TCP congestion control uses a strategy called Additive Increase Multiplicative Decrease (AIMD) [24]. The basic idea is that when a packet is acknowledged, the window size is increased by one, but when a packet is lost, the window is reduced to half of its previous size. However, TCP has often been found to be inefficient over links with large delays and/or high packet loss rates [25–28]. TCP throughput is inversely proportional to round trip time (RTT), which may lead to efficiency and fairness problems. Among multiple connections with different RTTs sharing a link, connections with shorter RTT will obtain more bandwidth resources. Due to these problems in TCP, some applications use User Datagram Protocol (UDP) [29] rather than TCP and are unresponsive to congestion indicators. However, excessive use of unresponsive applications can lead to congestion collapse of the computer network [30,31].

To solve the problems of TCP, protocols based on explicit congestion control have been proposed, such as eXplicit Control Protocol (XCP) [25, 26], Rate Control Protocol (RCP) [27], and TCP-Explicit Rate (TCP-ER) [28]. In these protocols, congestion control is based on the feedback of an explicit sending rate determined by the network elements (e.g., routers or switches) along the path from source to destination. Rather than increase the window size by one during each RTT, the sender transmits data at an explicit rate signaled by the network. If congestion occurs in the network, the network feeds back an adjusted explicit rate to the sender in order to alleviate the congestion.

1.1 Problem Statement

ECMP and OMP provide multipath routing to solve the problem of shortest path routing. However, ECMP is not guaranteed to determine a multipath route for each source-destination pair because only equal cost paths are considered for a multipath route. Also, packets are forwarded in equal proportion, on a packet-by-packet basis, over the paths in the multipath route, without considering network congestion. As a result, packets may arrive out-of-order within a flow at the destination which leads to degraded TCP performance. In OMP, the characteristics (i.e., disjointedness/partially disjointedness) of the multipath route are not taken into account. OMP is triggered by path utilization information so multiple nodes sharing common subpaths may simultaneously begin distributing traffic over multipath routes sharing common links. Although congestion may be avoided over the original path, other paths may become congested as an unwanted side effect of the OMP traffic distribution policy. This could lead to the triggering of OMP at further nodes, which may eventually result in network instability. In both ECMP and OMP, each path in a multipath route may have a different propagation delay. Providing a longer delay path to a TCP flow in a multipath route could result in degraded TCP performance.

Explicit rate control protocols such as XCP [25, 26], and RCP [27] have been proposed to solve the problems of TCP. The sending rate is determined by the network and fed back to the sender as an explicit rate. Then, the sender transmits data at the given explicit rate.

However, these protocols may underestimate or fail to take into account the effect of the feedback propagation delay between the network and sender. This may lead to congestion collapse due to the use of outdated rate information. Dynamic rate controllers proposed in [32–35] take into account this propagation delay in the explicit rate computation. These controllers are designed based on queue dynamics so that their mechanisms are optimized for relatively heavy traffic conditions, but not for light traffic conditions. Also, these controllers result in rate fluctuations even in steady-state conditions, that is, they do not provide a constant rate in steady state.

1.2 Research objective

The primary goal of this research is to develop multipath and explicit rate congestion control algorithms to optimize network performance. An essential step towards this goal is the development of an algorithm to determine multipath routes, which can be implemented on top of a conventional shortest path routing algorithms without requiring major changes to the underlying network infrastructure (see Chapter 3). We consider a traffic distribution mechanism that can distribute traffic over a multipath route using the current Internet infrastructure (see Chapter 4). Then, we develop an implementation of a rate-based transport protocol to improve the throughput performance of the Internet via explicit rate signaling (see Chapter 5). Finally, we discuss a novel rate computation algorithm for determining the flow sending rates, which can minimize network congestion and enable high utilization of the network capacity (see Chapter 6).

The research contributions of this thesis may be divided into two distinct parts:

1. multipath route discovery and traffic distribution (Chapters 3 and 4), and

2. explicit rate transport and explicit rate computation (Chapter 5 and 6).

The first part is targeted at alleviating network congestion by exploiting path redundancy in the network. The second part focuses on congestion avoidance via explicit rate control to improve end-to-end throughput for a flow traversing a given path. The two approaches

to congestion control studied in this thesis can be integrated under a unified framework to achieve even better network performance and end-to-end throughput. The possibility of combining multipath routing and explicit rate control is briefly discussed in Chapter 7 as a potential future research direction.

1.3 Main contributions

The main contributions of this dissertation are summarized as follows.

1.3.1 Multipath route discovery

We propose an algorithm to determine multipath routes, which can be implemented on top of shortest path routing algorithms without major changes to the network infrastructure. As a result, the proposed algorithm can eliminate the need for path pre-establishment (e.g., RSVP [36] or ATM UNI [37]) or the use of source routing protocols (e.g., DSR [38]) for packet forwarding, which are required by many of the multipath routing protocols proposed in the literature (e.g., [10, 39–42]).

The proposed algorithm for finding multiple paths between a source and destination node involves examining and classifying the set of shortest paths to the destination from the neighbors of the source node. Ideally, the set of paths in a multipath route should be link disjoint to maximize routing diversity, but partially disjoint paths may be sufficient for the purposes of congestion avoidance. An important feature of the proposed algorithm is that it can discover a larger set of paths by systematically relaxing the requirement of path disjointedness.

1.3.2 Multipath traffic distribution mechanism

We propose a "Congestion-triggered Multipath Protocol (CTMP)" to respond to link congestion by marking use of multipath routes, which improves the network utilization and overall end-to-end performance. A link is identified as being congested if the average link utilization exceeds a given threshold. In case of light link congestion, only the local router

(i.e., a router attached to the congested link) responds to this congestion by distributing traffic over multipath. By contrast, in OMP, path congestion results in a response by all routers in the network. In CTMP, when heavy congestion occurs such that the local router alone cannot resolve it, its neighbors (i.e., routers adjacent to the router) are signaled to respond to the congestion.

1.3.3 Explicit rate congestion control

We propose an explicit rate transport protocol that takes into account both the sending rate and the feedback propagation delay. The proposed protocol signals both the explicit rate (ER) and the round trip time (RTT) along the path between a source and destination pair. This information enables the routers to compute flow sending rates to maximize throughput while maintaining a desired level of QoS. We study the performance of the proposed protocol with respect to various link delays and loss rates. In addition, we compare the throughput performance of the proposed protocol with that of other transport protocols. In particular, our protocol has better end-to-end performance than existing protocols such as TCP, UDT, and TCP-ER over impaired links (i.e., lossy links).

1.3.4 Explicit rate computation

We propose a dynamic algorithm to compute the local explicit rate at a given router. The proposed rate controller computes a per flow sending rate but does not require the maintenance of per flow state. The rate controller computes a single fair share sending rate for all flows on an outgoing link. Finally, this can eliminate legacy TCP's unfairness to flows with long propagation delays. The proposed controller can resiliently respond to high/low link utilization by taking into account the link utilization and queue dynamics simultaneously. This leads to full utilization of the network link capacity, and avoids packet loss and waste of available bandwidth. Thus, the controller can effectively and efficiently respond to dynamic changes in the state of the network.

1.3.5 Summary of thesis

The remainder of this thesis is organized as follows. We provide background and a literature survey of related work on multipath routing and end-to-end congestion control in Chapter 2. In Chapter 3, we propose a multipath route discovery algorithm based on shortest path information. In Chapter 4, we develop an algorithm to distribute traffic over multipath routes in order to alleviate network congestion. In Chapter 5, we develop an explicit rate signaling and transport protocol that takes into account the feedback delay and the sending rates computed by routers along a path. In Chapter 6, we develop and study the performance of an explicit rate computation algorithm. Finally, we conclude this thesis in Chapter 7.

Chapter 2: Background and Literature Survey

We first discuss background and related research on multipath routing and then discuss related work on explicit rate congestion control protocols.

2.1 Multipath routing

2.1.1 Basic concepts from graph theory

A network can be represented by a directed graph $G = (V, E)$ with node set V and link or edge set E. The number of nodes and links in the network are denoted by $|V|$ and $|E|$, respectively. A link in E between nodes i and j is denoted as an ordered pair (i, j), where i is referred to as the *tail* of the link and j is the *head* of the link. Link (i, j) has an associated positive *cost* or *length* $D(i, j)$.

A *path* p is a sequence of nodes such that from each node in the path, there is a link to the next node in the sequence. The first node in the sequence is called the *source* node and the last node is called the *destination* node. The remaining nodes are known as *intermediate* nodes. As an example, the path

$$p = \{s, i_1, i_2, \cdots, i_n, d\}$$

consists of the source node s, intermediate nodes i_1 through i_n, and the destination node d. The path p can also be represented as a sequence of links as follows:

$$L(p) = \{(s, i_1), (i_1, i_2), \cdots, (i_{n-1}, i_n), (i_n, d)\},$$

where $L(p)$ is called the *link representation* of the path p. A *cycle* or *loop* is a path such

that the source and destination are the same. A path with no repeated nodes is called a *simple* or *loop-free* path.

Two paths are said to be *link disjoint* or simply *disjoint* if the link representations of the paths are disjoint, i.e., the two paths do not share a common link. A *multipath route* is a set of paths, each of which has the same source and destination node. We also refer to each path in a multipath route as an *alternative* path. In a network, a packet sent from the source node on any of the alternative paths will arrive at the same destination node. For a multipath route, link disjoint paths are desirable because the traffic distributed over the alternative paths in a multipath route do not contend for common network resources.

2.1.2 Finding multipath routes

The K shortest paths problem has been widely researched since the 1950's [11–20] in order to find multipath routes. The K shortest paths problem is to find the K paths between a source and destination pair with minimum total length. There are two versions of the K shortest paths problem: loop-allowed and loop-free. We focus on the loop-free problem in order to avoid routing loops in the path. The algorithms discussed in [12, 14, 16–19] find loop-free K shortest paths.

The removing path algorithm [17, 18] and the deviation path algorithm [14, 19, 20] have been mostly utilized to find K shortest paths. The removing path algorithm determines the K shortest paths such that it finds the shortest path in the current network topology, removes the path in the current topology, and finds the shortest path in the resulting topology. This procedure is repeated until the K shortest paths have been determined or there is no path between source and destination in the resulting topology. However, this algorithm requires path pre-establishment or source routing in order to deliver data from source to destination.

The deviation path algorithm is based on the construction of a pseudo-tree. This algorithm attempts to build a tree of K shortest paths. It first determines the first shortest path in the network topology, then finds the second shortest path based on the first shortest path.

The third shortest path is determined based on the first and second shortest paths. This procedure is repeated until K shortest paths are determined. References [14, 19] proposed protocols that can find disjoint multipath routes using the deviation path algorithm.

Unlike the removing path algorithm, the deviation path algorithm may not require source routing or path pre-establishment. However, the deviation path algorithm provides no control on the degree of path disjointedness among alternative paths. This is an important issue in controlling congestion via multipath routes, as we shall see in the proposed CTMP (Congestion-Triggered MultiPath) protocol to be discussed in Chapter 4.

In order to find sufficient number of disjoint paths between source and destination nodes in a network, a number of algorithms have been proposed in the literature [10, 39–42]. The shortest pairs of disjoint paths problem (SPDP) can be defined as follows: Given a destination node d and for each node $s \neq d$, find a pair of disjoint paths from s to d of minimum total length. Ogier *et al.* [40] present a distributed algorithm to solve SPDP by reducing the problem to a shortest path problem on a modified graph. Most of the existing algorithms to find disjoint paths have significant communication and time complexity requirements and cannot easily be bootstrapped onto an existing shortest path routing infrastructure. Moreover, forwarding packets over a set of disjoint paths generally requires the support of source routing [43] or the pre-establishment of switched paths as in ATM (Asynchronous Transfer Mode) [44] or MPLS (Multi-Protocol Label Switching) [45]. An interesting alternative approach for packet forwarding over disjoint paths is proposed in [40], which incurs significantly less overhead than source routing, but still requires modifications to the routing infrastructure.

Another approach to finding multipath routes is to exploit shortest path information derived from a shortest path routing protocol [46–51]. In this approach, the set of paths in a multipath route includes the shortest path, obtained from the shortest path routing protocol, plus alternative paths derived from shortest paths from each of the neighbors of the source node to the destination node. Here, the set of paths in the multipath route is not guaranteed to be disjoint. An alternative path via a neighbor node j is included in

the multipath route only if the length of the shortest path from j to the destination node is strictly less than the length of the shortest path between the source and destination. Application of this rule avoids the formation of routing loops in the alternative path and ensures that the length of the alternative path is not significantly greater than that of the shortest path.

In Chapter 3, we propose an algorithm for finding multipath routes based on shortest path routing information, but unlike the earlier approaches, it does not require the path from the neighbor node to the destination to be less than the length of the shortest path between the source and the destination. This allows us consider more paths to be considered for possible inclusion in the multipath route. In particular, the paths are classified according to how disjoint they are with respect to each other. Generally, disjoint paths are more desirable to improve routing resiliency, but if an insufficient number of disjoint paths is available, partially disjoint paths may be sufficient to alleviate network congestion. Since our proposed mechanism for multipath traffic distribution is triggered by local link utilization, partially disjoint paths may be just as effective as disjoint paths alleviating congestion at.

2.1.3 Multipath traffic distribution

The Equal-Cost MultiPath (ECMP) protocol [4] is a multipath routing extension for Internet routing protocols such as OSPF and RIP [3]. In ECMP, a node implements multipath routing when it discovers two or more shortest paths (of equal length) to a destination node. These paths can be determined via relatively simple extensions of standard shortest path algorithms such as Dijkstra's algorithm or the Bellman-Ford algorithm. The paths making up a multipath route need not be disjoint. Under ECMP, once a multipath route is discovered, packets are forwarded in equal proportion over the set of paths in the multipath route. There are several drawbacks of this approach: (1) ECMP is not guaranteed to determine a multipath route for each source-destination pair; (2) The characteristics of the multipath route are not taken into account; (3) Packets are forwarded in equal proportion, on a packet-by-packet basis, over the paths in the multipath route, without considering network

congestion. As a result, packets may arrive out-of-order within a flow at the destination. Moreover, ECMP may actually cause more congestion than single path routing in some scenarios.

The Optimized MultiPath (OMP) protocol [21] is an improved version of ECMP for link state routing protocols such as OSPF, which allows unequal traffic distribution. Multipath routes are found using the approach based on shortest path routing. The paths in the multipath route need not be of equal length. Traffic is distributed over a multipath route in inverse proportion to the utilizations of the constituent paths. Path utilization information is inferred from link state information. Unlike ECMP, OMP ensures that packets belonging to a flow are always forwarded on the same path. Thus, traffic is distributed over a multipath route at the granularity of a flow, which avoids the out-of-order packet problem of ECMP.

Since OMP is triggered by path utilization information, multiple nodes sharing common subpaths may simultaneously begin distributing traffic over multipath routes sharing common links. Although congestion may be avoided over the original path that triggered the OMP protocol, other paths may become congested as an unwanted side effect of the OMP traffic distribution policy. This could lead to the triggering of OMP at further nodes, which may eventually result in network instability.

The multipath traffic distribution mechanism proposed in Chapter 3 is triggered by link utilization rather than path utilization. The node attached to the local congested link attempts to resolve congestion by shifting traffic away from the congested link onto alternative paths. If the congestion cannot be resolved at the local node, the multipath traffic distribution is pushed back to an upstream neighbor node. This approach introduces multipath routing in a more controlled fashion in comparison to OMP.

2.2 End-to-end congestion control

Approaches to solve problems of TCP can be categorized as (1) implicit congestion control or (2) explicit congestion control. In implicit congestion control, the sender infers congestion from packet losses whereas in explicit congestion control, the sender is notified of the sending

rate by network elements (e.g., routers or switches).

The basic TCP congestion control algorithm is an example of implicit congestion control. Variants of the basic TCP congestion control mechanism have been proposed such as TCP-Tahoe [24], TCP-Reno [52], TCP-NewReno [2], Scalable TCP [53], HighSpeed TCP [54], and TCP-SACK [55–57]. These protocols provide implicit packet loss information to the sender in various ways, which the sender uses to adjust its transmission rate accordingly.

Another approach to adjust the transmission rate is based on queue occupancy. The so-called Active Queue Management (AQM) schemes include Random Early Detection (RED) [58], Random Early Marking (REM) [59], Proportional Integral Controller [60], Virtual Queue [61], Adaptive Virtual Queue [62], and Explicit Congestion Notification (ECN) [63]. These approaches involve signaling the sender about the existence of congestion to sender based on a queue occupancy threshold at the local router. Low *et al.* [64] and Katabi and Blake [65] demonstrate that AQM schemes can make the sender's transmission rate become oscillatory and unstable, eventually leading to low link utilization.

In explicit congestion control, rather than using feedback information in the form of packet loss or congestion indication based on queue occupancy, congestion control is based on the feedback of an explicit sending rate determined by the network elements (e.g., routers or switches) along the path from source to destination. According to the type of feedback, explicit feedback schemes can be categorized as (1) window-based congestion control, such as eXplicit Control Protocol (XCP) [25, 26], and (2) rate-based congestion control, such as Rate Control Protocol (RCP) [27] and TCP-Explicit Rate (TCP-ER) [28]. In window-based congestion control, routers compute an increment or decrement for the sending window size, which is fed back to the sender. By contrast, in rate-based congestion control, routers calculate the sending rate and feed this rate back to the sender. While window-based control allows a sender to inject all the data in a window into the network and then wait for the window to re-open, a sender employing a rate-based scheme generates a smooth data flow, similar to the traffic shaping mechanism in ATM (e.g., [37, 44, 66, 67]). References [68, 69]

demonstrate that rate-based congestion control provides a more efficient solution for high-speed networks than window-based congestion control.

However, the existing rate-based protocols generally underestimate the effect of the feedback propagation delay between the network and sender or do not take it into account. This can potentially lead to congestion collapse. For example, suppose that the rate R is signaled to the sender from a group of routers along a path and then, the routers decrease the feedback rate to a value $R' < R$. During the time it takes for the new rate R' to be signaled to the sender, the sender will continue to send at rate R, which may lead to network congestion. References [32–34] provide solutions for this problem by utilizing dynamic rate controllers taking into account the propagation delay. These controllers are designed based on queue dynamics so that performance is optimized under heavy load traffic conditions. However, under light traffic conditions (i.e., the queue is empty or almost empty), these controllers take a longer time to stabilize and in the mean time may cause network congestion. Reference [34] proposes a queue-based controller, using a so-called a High Gain Controller (HGC). The HGC responds to light traffic conditions by changing the controller gains to high values. Nonetheless, the HGC does not guarantee prevention of queue overflow.

Chapter 3: Multipath Route Discovery

In this Chapter, a multipath route discovery algorithm is proposed that can reduce the network overhead for path establishment and provide multiple paths with various degrees of disjointedness [70]. The proposed algorithm finds a set of paths forming a multipath route, assuming an underlying shortest path routing infrastructure. Thus, this approach does not require source routing or additional signaling mechanisms for path setup. We first formally define the notions of shortest path routing and disjoint alternative paths. Based on these definitions, we introduce the concept of class-c routes and propose an algorithm to find class-c routes to form a multipath route.

3.1 Shortest path routing

For the network $G = (V, E)$, a shortest path routing algorithm determines a unique *shortest path*, p_{sd}, from each node s to every other node d in the network. Shortest paths determined by a shortest path routing algorithm possess the following important property [71].

Definition 1. Shortest path property: *Let p_{sd} be the shortest path in G from node s to node d. Then any subpath of p_{sd} is also the shortest path between its two end nodes.*

Consider the (unique) shortest path p_{sd} from node s to node d, determined by a routing algorithm. Let $\mathcal{N}(s)$ denote the set of neighbor nodes of node s. To obtain alternative paths from s to d, we consider paths of the form

$$p_{sd}^{j} = \{s\} \oplus p_{jd}, \tag{3.1}$$

where '$\oplus$' denotes the concatenation of two (finite) sequences. Thus, the path p_{sd}^{j} begins

at node s, proceeds to neighbor node j, and then follows the shortest path route p_{jd} from node j to node d.

However, p_{sd}^j might form a routing-loop by having node s as an intermediate node. In order to prevent routing-loops, a restriction is imposed on establishing p_{sd}^j. Note that the path p_{sd}^j is loop-free provided that $s \notin p_{jd}$.

Lemma 3.1.1. *If $s \notin p_{jd}$ then p_{sd}^j is loop-free.*

Proof. Suppose $p_{sd}^j = \{s\} \oplus p_{jd}$ contains a loop. Since p_{jd} is a shortest path, it cannot contain a loop. Therefore, the loop must contain node s, which implies that $s \in p_{jd}$. □

Define a set, $\mathcal{A}_{sd}$, of alternative paths from node s to node d via neighbor nodes as follows:

$$\mathcal{A}_{sd} = \{p_{sd}^j : j \in \mathcal{N}(s), s \notin p_{jd}\}. \tag{3.2}$$

Thus, $\mathcal{A}_{sd}$ is the set paths from s to d via a neighbor node j that is not in the path p_{sd}. To form a multipath route from node s to d, we shall consider the set of alternative paths in the set $\mathcal{A}_{sd}$. To make use of an alternative path $p_{sd}^j \in \mathcal{A}_{sd}$, node s merely forwards a packet to node j. Under conventional single path routing, node j will then forward the packet along the shortest path p_{jd} to the destination node d.

An important consequence of the shortest path property is as follows.

Proposition 3.1.2. *Two alternative paths from a source node s to a destination node d are disjoint if and only if the only common nodes are s and d, i.e., the paths do not have any common intermediate nodes.*

Proof. Let u and v be two alternative paths from s to d. If u and v share a common link, it is clear that they must share at least one common intermediate node. Conversely, suppose that u and v share a common intermediate node i. Then the shortest path property implies that the shortest path p_{sd} is a subpath of both u and v. Hence, u and v are not disjoint in this case. □

In particular, any two paths in $\mathcal{A}_{sd}$ are disjoint if and only if the only common nodes are s and d.

3.2 Disjoint alternative paths

To maximize routing diversity and resilience, it is often desirable to form a multipath route from a set of pairwise disjoint paths. Define the set of paths

$$\mathcal{P}_{sd} = \{p_{sd}\} \cup \mathcal{A}_{sd}.$$

The following lemma gives a condition for two shortest paths to a common destination node to be disjoint.

Lemma 3.2.1. *Consider two shortest paths p_{sd} and p_{jd} from two distinct nodes s and j, respectively, to a common destination node d. If the two paths do not have a common penultimate (second to last) hop k, then they must be disjoint. Conversely, if paths p_{sd} and p_{jd} are not disjoint, they must share a common subpath p_{ad}, which is a shortest path from node a to d.*

Proof. Let k and m be the penultimate hops in paths p_{sd} and p_{jd}, respectively, and assume that $k \neq m$. Suppose that there exists a common link (a, b) shared by p_{sd} and p_{jd}. From the shortest path property of Definition 1, path p_{ad} is a shortest path from node a to node d. Furthermore, path p_{ad} must be a subpath of both p_{sd} and p_{jd}. But this implies that $k = m$, which contradicts our original assumption. Hence, p_{sd} and p_{jd} must be disjoint.

If p_{sd} and p_{jd} are not disjoint, then they must share a common link (a, b). By the shortest path property, the subpath between a and d in p_{sd} corresponds to the shortest path p_{ad}. Similarly, the subpath between a and d in p_{jd} corresponds to the shortest path p_{ad}. Thus, p_{ad} is a subpath of both p_{sd} and p_{jd}. □

Corollary 3.2.2. *Consider path p_{sd} and path $p_{sd}^j \in \mathcal{A}_{sd}$. If p_{sd} and path p_{sd}^j do not share a common penultimate hop k, then they must be disjoint.*

Proof. If p_{sd} and p^j_{sd} do not share a common penultimate hop, then neither do p_{sd} and p_{jd}. By Lemma 3.2.1, paths p_{sd} and p_{jd} must be disjoint. By the definition of $\mathcal{A}_{sd}$, link (s, j) is not contained in p_{sd}. Hence, paths p_{sd} and p^j_{sd} must be disjoint. □

Using Lemma 3.1.1 and Corollary 3.2.2, the following results are given.

Proposition 3.2.3. *Let p_{sd} be the shortest path from node s to node d and let p_{jd} be the shortest path from node j to node d. If p_{sd} and p_{jd} are disjoint, then p_{sd} and p^j_{sd} are disjoint and p^j_{sd} is loop-free.*

Proof. If p_{sd} and p_{jd} are disjoint, they cannot have a common penultimate hop. Hence, by Corollary 3.2.2, p_{sd} and p^j_{sd} must be disjoint. Also, since p_{sd} and p_{jd} are disjoint, node s should not be in p_{jd}. Hence, by Lemma 3.1.1, p^j_{sd} is loop-free. □

Proposition 3.2.4. *Consider paths $p^j_{sd}, p^l_{sd} \in \mathcal{A}_{sd}$, where $j \neq l$. If p^j_{sd} and p^l_{sd} do not share a common penultimate hop, then p^j_{sd} and p^l_{sd} are disjoint.*

Proof. If p^j_{sd} and p^l_{sd} do not share a common penultimate hop, than neither do the paths p_{jd} and p_{ld}. Hence, by Corollary 3.2.2, p_{jd} and p_{ld} are disjoint. Since $j \neq l$, p^j_{sd} and p^l_{sd} are also disjoint. □

Proposition 3.2.3 is used to choose a path p^j_{sd} that is loop-free and disjoint from the shortest path p_{sd}. Proposition 3.2.4 is used to ensure that two alternative paths p^j_{sd} and p^l_{sd} are disjoint from each other.

3.3 Class-c routes

To form a multipath route, we define a set $\mathcal{P}^{(c)}_{sd}$ of *class-c routes* between s and d, which have the property that any two paths $p, q \in \mathcal{P}^{(c)}_{sd}$ have at most c common links, i.e.,

$$|L(p) \cap L(q)| \leq c. \tag{3.3}$$

The class $\mathcal{P}_{sd}^{(c)}$ is defined by Algorithm 1. In particular, the set of class-0 paths consists of pairwise disjoint paths from s to d. The set of class-1 paths has the property that every pair of paths shares at most a single common link, i.e., a link (k, d), where k is a neighbor of node d. The set of class-2 paths has the property that every pair of paths shares at most two common links, say (k, d) and (k', k), which form a subpath $\{k', k, d\}$. The sets of class-3, class-4, etc., can be characterized similarly. Note that the set of class-c routes is contained in the set of class-$(c+1)$ paths for $c = 0, 1, 2, \cdots$. Thus,

$$P_{sd}^{(0)} \subseteq P_{sd}^{(1)} \subseteq P_{sd}^{(2)} \subseteq \cdots \subseteq P_{sd}^{(c)} \subseteq P_{sd},$$

for $c > 2$.

In Algorithm 1, the set $P_{sd}^{(c)}$ is initialized to contain the shortest path p_{sd}. The other paths in $P_{sd}^{(c)}$ are selected from the set of alternative paths $\mathcal{A}_{sd}$ in increasing order of path length. The **while** loop iterates over all paths in the set $\mathcal{A}_{sd}$. In lines 4 and 5, the shortest path $\tilde{p}$ in the (current) set $\mathcal{A}_{sd}$ is removed from the set. In lines 6-13, the candidate path $\tilde{p}$ is tested against all of the paths in the (current) set $\mathcal{P}_{sd}^{(c)}$ to check whether the condition (3.3) is satisfied. If $\tilde{p}$ satisfies (3.3) for all paths $p \in \mathcal{P}_{sd}^{(c)}$, then $\tilde{p}$ is added to the set $\mathcal{P}_{sd}^{(c)}$ (lines 14-16). The computational complexity of Algorithm 1 is $O(|\mathcal{N}(s)|^2)$ where $|\mathcal{N}(s)|$ is the number of neighbors of node s.

3.4 Finding class-c routes on various network topologies

We applied Algorithm 1 to various network topologies given in [72]. The results are shown in Table 3.1. Each row in the table indicates the average number of class-0, class-1, and class-2 paths found by the Algorithm 1 for a given network topology. Each network topology is characterized by the number of nodes N, the number of (unidirectional) links L, and the average node degree d. In the example of the "six-node" topology, each node has, on average, 1.93 class-0 paths, 2.13 class-2 paths, and 2.2 class-3 paths from each source node

Algorithm 1 Finding class-c routes from s to d.

1: **Input:** $\mathcal{A}_{sd}$, c, s, d; **Output:** $\mathcal{P}_{sd}^{(c)}$
2: $\mathcal{P}_{sd}^{(c)} \leftarrow \{p_{sd}\}$
3: **while** $\mathcal{A}_{sd} \neq \emptyset$ **do**
4: $\tilde{p} \leftarrow \text{argmin}\{D(q) : q \in \mathcal{A}_{sd}\}$
5: $\mathcal{A}_{sd} \leftarrow \mathcal{A}_{sd} \backslash \{\tilde{p}\}$
6: size $\leftarrow |\mathcal{P}_{sd}^{(c)}|$
7: **for** each $p \in \mathcal{P}_{sd}^{(c)}$ **do**
8: **if** $|L(\tilde{p}) \cap L(p)| \leq c$ **then**
9: size $\leftarrow$ size - 1
10: **else**
11: **break**
12: **end if**
13: **end for**
14: **if** size $= 0$ **then**
15: $\mathcal{P}_{sd}^{(c)} \leftarrow \mathcal{P}_{sd}^{(c)} \cup \{\tilde{p}\}$
16: **end if**
17: **end while**

to every other node. The bottom row of the table shows the average numbers of class-c routes averaged over all ten topologies. Since the average number of class-1 paths is 2.35, a given node has, on average, at least two class-1 paths to every other node.

3.5 Conclusion

We have characterized multipath routes and proposed class-c path algorithm to determine multipath routes over shortest path routing. The proposed algorithm does not require pre-establishment of paths or support for source routing. The algorithm provides a mechanism to find both disjoint paths and partially disjoint paths between two nodes. We applied the algorithm to various network topologies. The results show that there are at least two class-1 paths between each pair of nodes, on average. The multipath route discovery algorithm proposed in this Chapter is used in the multipath traffic distribution algorithm developed in Chapter 4.

Network Topology	**Parameters**			**Class-c routes**		
	N	L	d	$c=0$	$c=1$	$c=2$
Six-node	6	16	2.66	1.93	2.13	2.20
Smallnet	10	44	4.4	2.77	3.78	3.88
LATA	11	46	4.18	2.38	3.37	3.58
NSFNET	14	42	3	1.89	2.12	2.23
Citi Multi-ring	15	40	2.67	1.38	1.67	1.83
Bellcore	15	54	3.6	1.72	2.43	2.73
EON	19	64	3.89	1.76	2.61	3.07
ARPA	20	62	3.1	1.66	1.90	2.12
ARPA2	21	50	2.38	1.31	1.41	1.46
US IP backbone	24	86	3.58	1.59	2.05	2.38
Average	15.5	51.4	3.346	1.84	2.35	2.55

Table 3.1: Number of class-c routes for different network topologies.

Chapter 4: Congestion-Triggered Multipath Routing

We now propose a scheme to alleviate network congestion by means of class-c multipath routes discussed in Chapter 3. The basic idea of our proposed *Congestion-Triggered MultiPath routing (CTMP)* scheme is that when a node detects congestion on a local link, it distributes traffic over class-c routes according to link utilization and path utilization measurements so as to resolve the local link congestion.

4.1 Network parameters

In the proposed CTMP scheme, each node manages network information about link and path utilizations. Let s denote the source node and j be a neighbor of node s under consideration. Link utilization is a good indicator of link congestion, whereas the path utilization provides an information on path congestion. Link and path utilization can be measured by the method introduced in [21]. CTMP controls the link utilization and the path utilization through three parameters, β, γ, and η. The values of all three parameters can be set by the network administrator. The parameters are defined below.

- β: A parameter indicating that the link or the path is about to be congested (i.e., overloaded). Whenever the utilization of link (s, j), LU_{sj}, reaches β (e.g., $\beta = 95$ %), the node executes multipath routing in order to alleviate the local link congestion.

- γ: A parameter indicating whether an alternative path p_{sd}^{j} can accept detour traffic or not, where detour traffic is the traffic moved from shortest path to alternative path. If the utilization of shortest path from neighbor node j to the destination PU_{jd} is less than γ (e.g., $PU_{sj} < 85$ %), this path is considered as a subset of multipath to the destination d. This condition prevents the path from the congestion caused by the

injection of detour traffic to the path. It also ensures that the subset of multipath keeps the available bandwidth.

- η: The expected PU_{sd} in steady-state (e.g., 90% of PU_{sd}). Our approach aims at managing all of links such that the utilizations of all links in the network are close to η by distributing traffic over multiple paths.

4.2 Routing and network information exchange

We modify Path Vector (PV) routing to compute class-c routes and to include additional congestion-related information in the routing control messages. PV routing is similar to Distance Vector (DV) routing, except that shortest path information is maintained by storing the ordering of intermediate nodes and the destination by means of a path vector. In addition to the path vector for PV routing, the CTMP scheme requires the storage and exchange of additional network state information as follows:

- ECI (Explicit Congestion Indication) and MPI (Multipath Indication).
- The path utilization $PU(p)$ and the path capacity $PC(p)$ along a path p,

The ECI bit is a one-bit flag stored in the routing table for every destination. The ECI bit is set to 0 by default to indicate no congestion and 1 to indicate congestion on the path to the destination. The MPI bit is also a 1-bit flag stored in the routing table for every destination. The MPI bit has a default value of 0 and is set to 1 if a class-c multipath route is established to the corresponding destination. We assume that every node s in the network can estimate $LU(l)$, the utilization of each local outgoing link l. Let $LC(l)$ denote the capacity of link l. The capacity and utilization of path p are defined, respectively, as

$$PC(p) = \min\{LC(l) : l \in p\}, \tag{4.1}$$

$$PU(p) = \min\{LU(l) : l \in p\}. \tag{4.2}$$

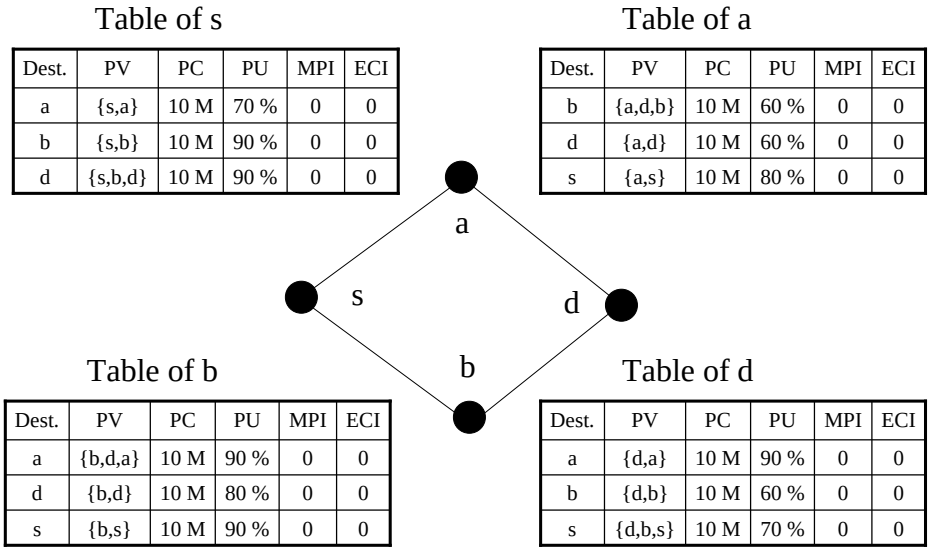


Figure 4.1: Routing tables for CTMP scheme.

The path capacity and utilization can be computed by a given node in a similar way to the path vector in PV routing. For example, a given node s estimates $PC(p_{sd})$ and $PU(p_{sd})$ by using routing/network information sent by its neighbors as follows:

$$PC(p_{sd}) = \min\{LC(s,j), PC(p_{jd})\}, \tag{4.3}$$

$$PU(p_{sd}) = \min\{LU(s,j), PU(p_{jd})\}. \tag{4.4}$$

Thus, node s does not need to directly exchange its local link capacity and utilization information with every other node in the network.

Figure 4.1 shows an example of the routing/network information exchange. Each node maintains its own routing table. Whenever the routing table of a node is updated, it sends a copy of the table to its neighbors. The routing table for each node is shown in the figure. Each row of the table shows the routing/network information for a given destination, including the destination node, the path vector (PV), the path capacity (PC), the path utilization (PU), the MPI bit, and the ECI bit.

4.3 Congestion resolution procedure

We develop an approach to resolve network congestion by distributing the traffic over class-c multipath routes obtained using Algorithm 1. Every node periodically estimates the utilization of its outgoing link l to detect link congestion. The procedure for congestion-triggered multipath traffic distribution to resolve congestion at the *local link* is given as follows:

1. When node s detects local congestion on the outgoing link l (i.e., $LU(l) > \beta$), or receives an ECI=1 bit from its neighbor node connected through link l, it tries to move some of the traffic on link l onto alternative paths, which are class-c routes determined using Algorithm 1.

2. Let $\mathcal{M} = \{p \in \mathcal{P}_{sd}^{(c)} : PU(p) < \gamma\}$. The parameter γ is called the *path availability threshold*.

3. Distribute traffic over the path set $\mathcal{M}$ according to the *traffic splitting function* $\phi(\cdot)$ (see Section 4.4 for more details).

4. Signal ECI=1 to its neighbor nodes if congestion is not resolved by Step 3 or detour traffic is not acceptable at alternative paths. If the present node is an edge node, execute the packet drop procedure as described in Section 4.5.

When congestion is detected on link l in Step 1, node s considers all of its shortest paths which contain link l as alternative paths. Node s also generates the set of class-c routes, $\mathcal{P}_{sd}^{(c)}$, to a given destination node d, if these paths have not yet been computed (i.e., the MPI bit is set to zero). Each incoming flow is identified by means of a hash function using the 5-tuple consisting of source/destination address, source/destination port, and protocol ID, i.e., (source IP, destination IP, source port, destination port, protocol ID) forms a flow ID. Then, the flow is forwarded to one of the class-c routes according to a distribution function, which will be discussed next.

4.4 Traffic distribution function

CTMP employs a hash function to identify incoming flows and distribute incoming traffic over the outgoing multipath routes. Given a network with N nodes, each node maintains at most $N-1$ hash tables for all destination nodes (i.e., one table for each destination). Nonetheless, if there is no congestion on the network, none of the nodes maintains a hash table. The input to the hash function is composed of the 5-tuple identifying the flow, which can be obtained from the packet header. Each packet is forwarded on the path associated with the associated flow.

The hash function has the property that the output of the hash function (i.e., the hash value) should be uniformly distributed over the hash space regardless of the input distribution. A uniform distribution is one for which the probability of occurrence is the same for all values of the output X. If a uniform distribution is divided into several different spaces, the probability of occurrence in each space can be obtained by the size of each space divided by entire space. If a uniform distribution is divided into several equal spaces, each space has the same probability of occurrence as others. Based on this property, the equal/unequal traffic distribution over multipath routes has been organized. The entire hash space is matched to the link utilization. If the link utilization exceeds the congestion threshold β, the hash table is partitioned a number of hash bins corresponding to the number of multipath routes. The hash space is determined by the traffic distribution function $\phi(\cdot)$.

Figure 4.2 shows the overview of our proposed traffic distribution algorithm. The proposed algorithm is similar to the table-based hashing algorithm introduced in [73]. However, the proposed algorithm is designed to adjust hash thresholds according to the path utilization, while the table-based hashing algorithm assigns the given hash thresholds (i.e., fixed hash thresholds). Assume that there are N different paths (i.e., N multipath routes) and M bins in the hash table. The M bins map into N paths according to the traffic splitting function. Each packet is forwarded to the associated path according to this mapping. For example, any packet with flow ID from 1 to S_1 is forwarded to path 1. However, this partition is maintained to agree with the traffic distribution function.

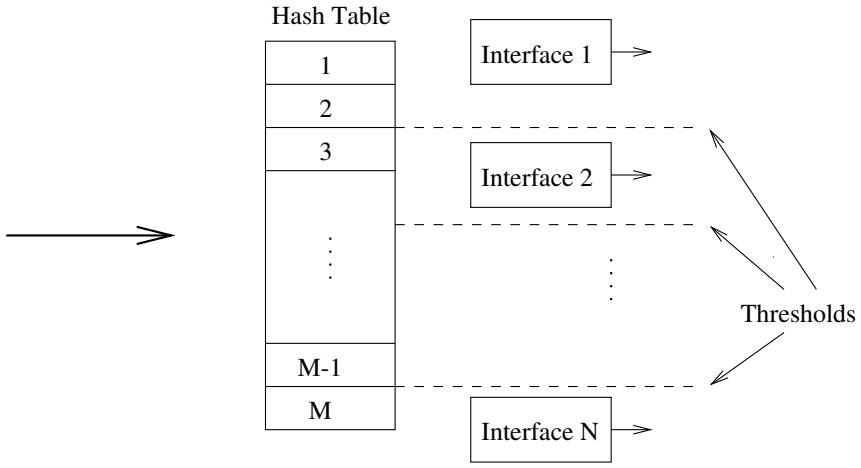


Figure 4.2: The hash table mapping to a multipath routes.

As discussed in Section 4.3, each node estimates the utilization of its local link. The amount of traffic F_l traversing the outgoing link l can be expressed as follows:

$$F_l = \sum_k \sum_{p:l \in p} f_p^k, \tag{4.5}$$

where f_p^k is kth flow traversing path p, where $f_p^k \geq 0$. The link utilization is given by $LU(l) = F_l/C_l$ where C_l is the capacity of link l. Each flow f_p^k is identified by hashing over the 5-tuple to obtain the flow ID at each node. Since flow is classified and maintained according to destination at node, F_l can be rewritten in terms of the destination node. Let X_l^d denote aggregate traffic flow traversing link l towards destination d. The aggregate traffic X_l^d has an associated hash table when link l is congested and its value is given by

$$X_l^d = \sum_k \sum_{p:d(p)=d, l \in p} f_p^k, \tag{4.6}$$

where $d(p)$ is the mapping function of $d : p \rightarrow d$. The total traffic on link l, F_l, can also be

rewritten in terms of destination node.

$$F_l = \sum_d X_l^d = \sum_d \sum_k \sum_{p:d(p)=d, l \in p} f_p^k. \tag{4.7}$$

When $LU(l)$ exceeds the link congestion threshold β, we consider the link to be congested. To resolve this link congestion, the node shifts some portion of F_l to an alternative path $p \in \mathcal{P}_{sd}^{(c)}$ where $\mathcal{P}_{sd}^{(c)}$ is a class-c multipath route, obtained by Algorithm 1. Again, the objective here is to decrease the link/path utilization to a more acceptable level by shifting a portion of the traffic to the alternative paths. To determine how many flows need to be shifted to alternative paths, F_l is roughly decomposed into two components, F_l' and F_l'', where F_l' is the amount of traffic traversing the shortest path p_{sd}, while F_l'' is the amount of traffic shifted to traverse the alternative paths, $\mathcal{P}_{sd}^{(c)} \backslash \{p\}$. We refer to the portion of traffic F_l'' as *detour* traffic. Then, we have

$$F_l = F_l' + F_l'' = \sum_d X_l^d\prime + \sum_d X_l^d\prime\prime = \sum_d X_l^d \tag{4.8}$$

Due to traffic shifting, the hash space for X_l^d is roughly divided into two spaces: one for the shortest path, and another for the alternative paths in the multipath route. The space for alternative paths is further divided into several spaces and each space is assigned to its associated alternative path.

By employing the traffic distribution function, every node may maintain the utilizations of all local links with a targeted level η or below. We design the traffic distribution function to maximize the utilization of the shortest path and maintain it just below the congested threshold β, under assumption that the shortest path is always better unless congestion occurs on the shortest path. The node detecting link congestion manages outgoing flows F_l on link l such that

$$F_l' = \eta C_l \text{ and } F_l'' = F_l - \eta C_l. \tag{4.9}$$

There might exist the case where the set of alternative paths, $\mathcal{P}_{sd}^{(c)} \backslash \{p\}$, does not have enough available bandwidth to accept the detour traffic F_l''. If this happens, the node signals ECI to its neighbor nodes. The traffic distribution function is designed in the inverse proportion to the link/path utilizations. The link/path with higher utilization is assigned a lower amount of detour traffic and that with lower utilization is assigned to a larger amount of detour traffic. For the traffic distribution function, let $p \in \mathcal{P}_{sd}^{(c)}$ be the shortest path containing link l, and let $\tilde{p} \in \mathcal{P}_{sd}^{(c)} \backslash \{p\}$ be an alternative path. The traffic distribution function is defined as a mapping $\phi : p \rightarrow [0, 1]$. The splitting ratio to path p is denoted by $\phi(p)$, and the proportion of traffic assigned to path $\tilde{p} \in \mathcal{P}_{sd}^{(c)}$ is defined as $\phi(\tilde{p})$. This splitting ratio should satisfy

$$\phi(p) + \sum_{\tilde{p} \in \mathcal{P}_{sd}^{(c)} \backslash \{p\}} \phi(\tilde{p}) = 1. \tag{4.10}$$

This constraint ensures that all flows are forwarded to class-c routes. From this equation, the proportion of traffic assigned to alternative paths is given by

$$\sum_{\tilde{p} \in \mathcal{P}_{sd}^{(c)} \backslash \{p\}} \phi(\tilde{p}) = 1 - \phi(p). \tag{4.11}$$

A fraction $\phi(p)$ of the hash space of size M is assigned to F_l' while F_l'' is mapped to the rest of the hash space. The rest of the hash space is further partitioned for each alternative path $\tilde{p}$ in proportion to $\phi(\tilde{p})$. The traffic splitting ratios, $\phi(p)$ and $\phi(\tilde{p})$, are initialized to 1 and 0, respectively. That is, all flows F_l initially traverse the shortest path p so, there is no detour traffic on any alternative path $\tilde{p}$.

CTMP introduces two phases for the traffic splitting: (1) traffic shift phase and (2) traffic withdrawal phase. Traffic shift phase is designed to shift some amount of traffic to the alternative paths. This phase is implemented when the link/path is congested. The

traffic withdrawal phase withdraws some or all of detour traffic from the alternative paths. This phase is executed when link congestion has been resolved.

4.4.1 Traffic shift phase

Before implementing the traffic shift phase, the node detecting congestion evaluates the alternative paths to see if they have enough capacity to carry detour traffic F_l'', where F_l'' is given by Eq. (4.9). The available bandwidth of all alternative paths, $\mathcal{P}_{sd}^{(c)}\backslash\{p\}$, is given by

$$\sum_{q\in\mathcal{P}_{sd}^{(c)}\backslash\{p\}} PC^t(q)[\gamma - PU^t(q)]^+, \tag{4.12}$$

where $[x]^+ \triangleq \max\{0, x\}$. If F_l'' is less than or equal to the available bandwidth of $\mathcal{P}_{sd}^{(c)}\backslash\{p\}$, the node executes the traffic shift phase. Otherwise, the node signals ECI=1 to its neighbors and initiates the traffic shift phase. By signaling ECI to neighbors, the node can reduce the incoming traffic from neighbors by performing load-balancing. The neighbor node also can shift traffic to its alternative paths as much as those paths can accept.

The distribution ratio of $\phi(p)$ is updated by adjusting from 1 to a certain level which is less than 1, such that the link utilization is reduced to $\max\{\eta LC(l), LC(l)LU^t(l) - \sum_{q\in\mathcal{P}_{sd}^{(c)}\backslash\{p\}} PC(q)[\gamma - PU^t(q)]^+\}$. Let t denote the (discrete-time) update time. Define

$$MAX = \max\{\eta LC(l), LC(l)LU^t(l) - \sum_{q\in\mathcal{P}_{sd}^{(c)}\backslash\{p\}} PC(q)[\gamma - PU^t(q)]^+\}.$$

Then, we have

$$\frac{LC(l)LU^t(l)}{\phi^t(p)} = \frac{MAX}{\phi^{(t+1)}(p)}, \tag{4.13}$$

where $LU^t(l)$ and $LC(l)$ are the current utilization of link l and the link capacity, respectively, $\phi^t(p)$ is the present splitting ratio for p, and $\phi^{(t+1)}(p)$ is the new splitting ratio for

p. This is rewritten as, in terms of $\phi^{(t+1)}(p)$,

$$\phi^{(t+1)}(p) = \phi^t(p)\frac{MAX}{LC(l)LU^t(l)}. \tag{4.14}$$

From Eqs. (4.11) and (4.14), the splitting ratio to alternative paths is obtained as:

$$\sum_{\tilde{p}\in\mathcal{P}_{sd}^{(c)}\setminus\{p\}} \phi^{(t+1)}(\tilde{p}) = 1 - \phi^t(p)\frac{MAX}{LC(l)LU^t(l)}. \tag{4.15}$$

From now on, we seek $\phi(\tilde{p})$ for each $\tilde{p} \in \mathcal{P}_{sd}^{(c)}\setminus\{p\}$. Each $\phi(\tilde{p})$ increases with the proportion of available bandwidth in alternative paths. The initial value of each $\phi(\tilde{p})$ is defined to be 0 because there is no detour traffic under no congestion conditions. $\phi(\tilde{p})$ increases by the function, the decrement of $\phi(p)$ multiplied by the proportion of available bandwidth on $\tilde{p}$ to available bandwidth of all alternative paths. Each $\phi(\tilde{p})$ is formulated for each $\tilde{p} \in \mathcal{P}_{sd}^{(c)}\setminus\{p\}$ as:

$$\begin{aligned}\phi^{(t+1)}(\tilde{p}) &= \phi^t(\tilde{p}) \\ &+ \frac{PC^t(\tilde{p})[\gamma - PU^t(\tilde{p})]^+}{\sum_{q\in\mathcal{P}_{sd}^{(c)}\setminus\{p\}} PC^t(q)[\gamma - PU^t(q)]^+} \\ &\times (\phi^t(p) - \phi^{(t+1)}(p)),\end{aligned} \tag{4.16}$$

where $PC^t(\tilde{p})[\gamma - PU^t(\tilde{p})]^+$ is the available bandwidth of path $\tilde{p}$, $\sum_{q\in\mathcal{P}_{sd}^{(c)}\setminus\{p\}} PC^t(q)[\gamma - PU^t(q)]^+$ is the available bandwidth of all alternative paths, and $(\phi^t(p) - \phi^{(t+1)}(p))$ represents the decrement of $\phi(p)$.

The neighbor nodes responding to congestion also have their own traffic distribution function. The expression $\phi(\tilde{p})$ for a neighbor node is the same as Eq. (4.16), but $\phi(p)$ is the same as Eq. (4.14) except that the neighbors are concerned with the utilization of the path

containing the congested link l. Let

$$MAX = \max\{\eta PC(l), PC(l)PU^t(l) - \sum_{q \in \mathcal{P}_{sd}^{(c)} \setminus \{p\}} PC(q)[\gamma - PU^t(q)]^+\},$$

where $PC(l)$ is the capacity of path containing link l and $PU^t(l)$ is the utilization of path containing link l at time t. We then have

$$\phi^{(t+1)}(p) = \phi^t(p)\frac{MAX}{PC(l)PU^t(l)}. \tag{4.17}$$

Therefore, the node detecting congestion assumes the traffic distribution function of Eqs. (4.14) and (4.16), and the neighbor nodes receiving ECI implement the functions given in Eqs. (4.17) and (4.16).

4.4.2 Traffic withdrawal phase

When congestion is resolved by load-balancing over class-c routes, some or all of detour traffic are withdrawn. This phase is executed at a node satisfying the following two conditions:

1. ECI is set to 0 and MPI is set to 1,
2. $LU(l)$ at the node detecting congestion or $PU(l)$ at its neighbors is less than γ.

When ECI=0 and MPI=1, this node establishes class-c routes to respond to congestion and does not need assistance from its neighbors to resolve congestion. If ECI is set to 1 with MPI=1, the node needs to have its neighbors distribute traffic over multipath routes. In this case, the traffic withdrawal is not initiated. The condition $LU(l) < \gamma$ or $PU(l) < \gamma$ indicates that the link or path has enough available bandwidth to accept the traffic from alternative paths that was supposed to traverse the shortest path.

The value $\phi(p)$ is updated by adjusting $LC(l)LU^t(l)$ (or, $PC(l)PU(l)$) to have a targeted utilization $\eta LC(l)$ or $\eta PC(l)$, so that the link utilization is increased to η. This is formulated as:

$$\frac{LC(l)LU^t(l)}{\phi^t(p)} = \frac{\eta LC(l)}{\phi^{(t+1)}(p)}. \tag{4.18}$$

From this equation, the traffic distribution function for the node detecting congestion becomes

$$\phi^{(t+1)}(p) = \phi^t(p)\frac{\eta}{LU^t(l)}, \tag{4.19}$$

while the distribution function for its neighbors is given by

$$\phi^{(t+1)}(p) = \phi^t(p)\frac{\eta}{PU^t(l)}. \tag{4.20}$$

The value $\phi(\tilde{p})$ should be updated as $\phi(p)$ increases. The sum of $\phi(\tilde{p})$ over the alternative paths $\tilde{p}$ decreases as $\phi(p)$ increases. The decreasing in $\phi(\tilde{p})$ is formulated in terms of the current distribution function $\phi^t(p)$ and the new distribution function $\phi^{(t+1)}(p)$. The total ratio of alternative paths, $1-\phi^t(p)$, decreases to $1-\phi^{(t+1)}(p)$. Thus, the new ratio of each alternative paths $\phi^{(t+1)}(\tilde{p})$ is given by

$$\phi^{(t+1)}(\tilde{p}) = \phi^t(\tilde{p}) \times \frac{1-\phi^{(t+1)}(p)}{1-\phi^t(p)}, \tag{4.21}$$

where $\frac{1-\phi^{(t+1)}(p)}{1-\phi^t(p)}$ represents the ratio to decrease for each $\phi(\tilde{p})$, and also for $\sum_{\tilde{p}\in\mathcal{P}_{sd}^{(c)}\setminus\{p\}}\phi(\tilde{p})$. Therefore, the node detecting congestion has the traffic distribution function of Eqs. (4.19) and (4.21), and its neighbor nodes receiving ECI implement the function of Eqs. (4.20) and (4.21).

Based on the value obtained by $\phi(\cdot)$, the hash space for each path is decided by Algorithm 2. This algorithm takes $\phi(\cdot)$ and $HashMax$ as inputs where $HashMax$ is the maximum hash table size supported by the hash function. The algorithm produces a hash table H_{sd} for the associated destination as output. It first computes the bin corresponding to the shortest path p from (lines 4 to 6). The hash space of path p always starts from the initial value of 0 and ends at the value of $HashMax * \phi(p)$. It then finds the bin for the alternative paths $\tilde{p}$. This hash table is used whenever the packet is forwarded to class-c routes. The hash value obtained by hashing over 5-tuple from packet header is compared with each bin in hash table. If this falls into a certain range from $val(i)$ to $val(i+1)$ then, the packet is forwarded through $val(i+2)$, the path associated with the bin.

Algorithm 2 Hash range for each path in $\mathcal{P}_{sd}^{(c)}$

1: **Input:** $\phi(p)$, $\phi(\tilde{p})$, $HashMax$; **Output:** H_{sd}
2: $H_{sd} \leftarrow \emptyset$
3: $i \leftarrow 0$
4: $val(i) \leftarrow 0$
5: $val(i+1) \leftarrow HashMax * \phi(p)$
6: $val(i+2) \leftarrow p$
7: $H_{sd} \leftarrow H_{sd} \cup \{val(i), val(i+1), val(i+2)\}$
8: $i \leftarrow i+3$
9: **for** each $\tilde{p} \in \mathcal{P}_{sd}^{(c)}$ **do**
10: $\quad val(i) \leftarrow val(i-3)$
11: $\quad val(i+1) \leftarrow val(i) + HashMax * \phi(\tilde{p})$
12: $\quad val(i+2) \leftarrow \tilde{p}$
13: $\quad H_{sd} \leftarrow H_{sd} \cup \{val(i), val(i+1), val(i+2)\}$
14: $\quad i \leftarrow i+3$
15: **end for**

It is important to define the network parameters, which must be maintained in order to keep the network stable. The three parameters β, η, and γ are related as follows:

$$0 < \gamma < \eta < \beta < 1. \tag{4.22}$$

The availability threshold γ should be assigned to be less than both η and β in order to avoid unwanted link/path congestion by shifting some portion of traffic from the original path to the alternative path. The targeted threshold η should be less than or equal to

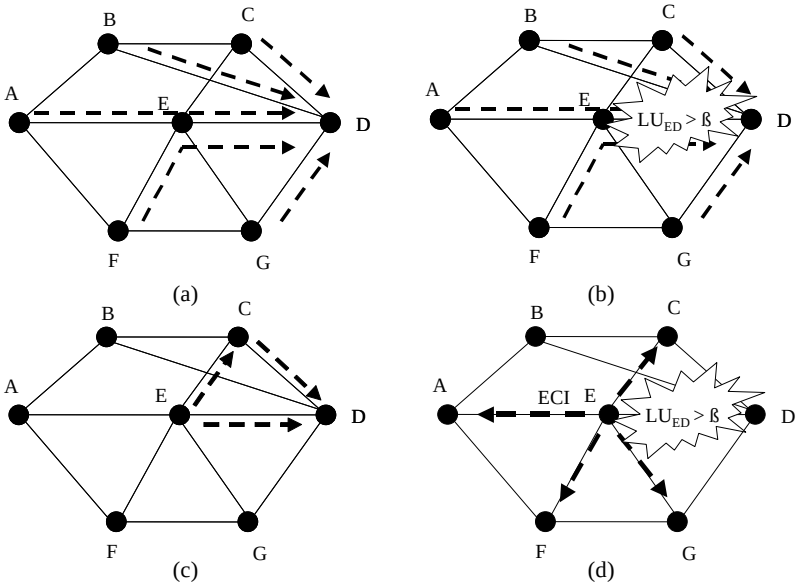


Figure 4.3: Example of network congestion alleviation.

the congestion threshold β. We investigate the detail of the relationships among the three parameters through the simulation results presented in Section 4.6. In summary, β can be set greater than or equal to η. That is, whenever the utilization of the link/path is greater than or equal to the targeted utilization, CTMP is triggered to balance the load over the network. Also, γ should be less than η so that the alternative path does not suffer from network congestion due to the detour traffic.

Figure 4.3 illustrates how the CTMP scheme resolves congestion for a simple network scenario. The dotted arrows in Figure 4.3(a) represent the shortest path from each node to node D. As shown in Figure 4.3(b), node E detects local congestion on link (E, D). Thus, node E computes the class-c routes (in this case $c = 0$) to node D to alleviate congestion on link (E, D). Figure 4.3(c) shows the class-0 routes with dotted arrows from node E to node D. The incoming traffic to node E for the destination node D is distributed according to the traffic distribution function $\phi(\cdot)$ given in Eqs. (4.14) and (4.16). In Figure 4.3(d), the rate of incoming traffic from node E to node D is too high for congestion to be resolved by multipath traffic distribution at node E. Thus, node E sends ECI signals to its neighbors A

and F. Nodes A and F set up class-c routes to node D. Node A establishes two paths, $A-E-D$, and $A-B-D$. Node F has two paths $F-E-D$ and $F-G-D$. The two nodes A and F distribute the traffic according to their traffic distribution functions. If their action does not resolve congestion on link (E, D), node E starts dropping incoming packets by the packet drop procedure described in Section 4.5. It might happen that link congestion is resolved by the assistance of neighbors (i.e., neighbors distribute traffic over the class-c routes), but congestion occurs again on link (E, D). To respond to this congestion, nodes E, A, and F adjust the traffic distribution ratio. If this adjustment is not effective, nodes A and F drop incoming packets. Once multipath routes from nodes A and F are no longer necessary (i.e., there is no detour traffic over alternative paths), those two nodes withdraw the multipath routes and signal ECI=0 to node E. Node E then sets ECI to 0. By this ECI exchange, the network condition returns to that of Figure 4.3(c). Also, node E can withdraw its class-c routes if there is no detour traffic over alternative paths.

4.5 Packet drop procedure

Network congestion may not be resolved even if every node associated with network congestion balances the load over multipath routes. The packet drop procedure is implemented as a last resort when there are no multipath routes available for the incoming traffic in the network. In CTMP, packets are dropped at an edge node if and only if the node is an edge node, and both ECI and MPI are set to 1. The packet drop procedure reduces the volume of incoming traffic to the entire network while the provisioning of multipath routes increases network throughput. This procedure is executed at all edge nodes whose paths to the destination are associated with network congestion and whose paths have congestion indication information (i.e., ECI = 1). Those edge nodes proportionally drop incoming packets incoming according to the path utilization. The packet drop procedure is placed between the hash table and the interface, as illustrated in Figure 4.4. Packets are dropped at the link associated with the congested path.

The process of packet drop proposed here is similar to RED (Random Early Drop) [58]

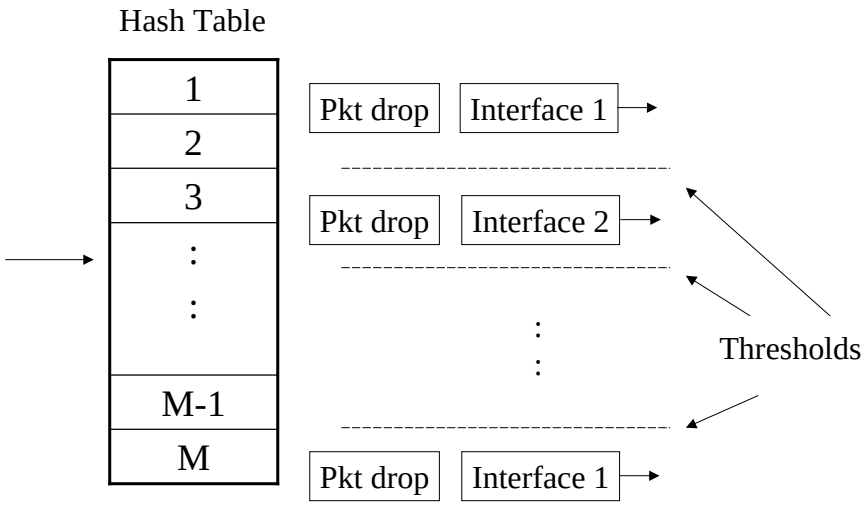


Figure 4.4: Packet drop procedure.

except it employs its own packet drop rate. The packet drop rate is estimated by exploiting the path utilization that every node maintains. Thus, this procedure enables the packet drop rate function to be resilient to changes of the link/path utilization. The packet drop rate function is similar to the slow start phase in TCP congestion control algorithm [22]. The packet drop rate is set to 0 by default and 1 for the initial drop when the edge node needs to drop packets. This rate increases and decreases exponentially every utilization update interval time. The drop rate exponentially increases when the path utilization is greater than the congestion threshold β, and exponentially decreases when the path utilization is less than β. When the packet drop rate is set back to 0 (i.e., initial value), the edge node adjusts the traffic distribution ratio.

4.6 Simulation results

We performed computer simulations to evaluate the CTMP scheme. The performance of CTMP is compared with that of three existing routing schemes: conventional PV routing, ECMP, and OMP. Specifically, we discuss the network performance of CTMP in terms of link utilization and the number of packet drops. In addition, we estimate TCP performance

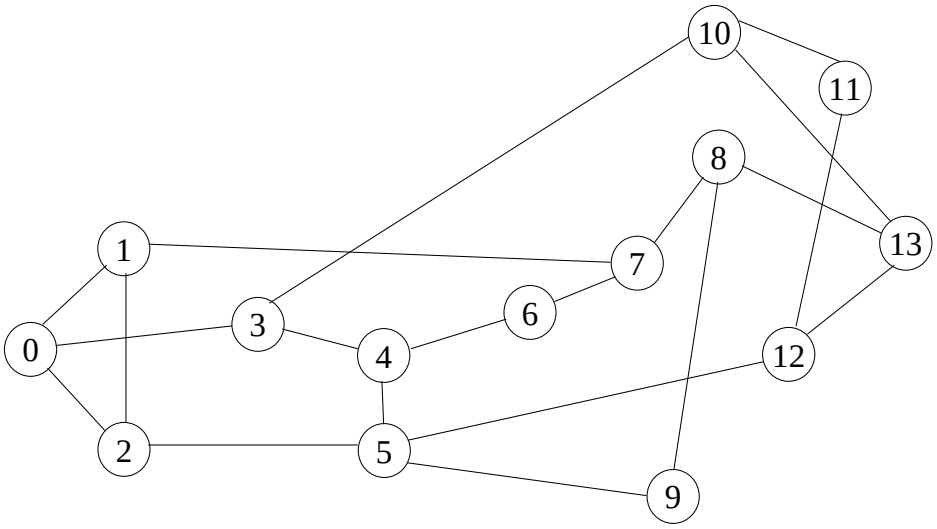


Figure 4.5: The network topology of NSFNET.

as well as network performance with a combination of TCP and UDP traffic. Further, we evaluate CTMP performance by varying class-c routes so as to estimate the effectiveness of class-c routes. Simulations are executed using class-0, class-1, and class-2 routes, respectively. Also, we ran simulations to evaluate the impact of three network parameters (i.e. β, γ, and η), the key parameters of CTMP.

The simulations were carried out using the ns-2 network simulator on the network topology shown in Figure 4.5. The topology consists of 14 different nodes and 21 identical links, each with a capacity of 1 Gbps. All links are bi-directional but links (i, j) and (j, i) are considered different links, for which separate measurements are taken to estimate the link utilization and the number of packets dropped on each link.

The traffic generation process is designed to cause congestion on the network. We explain how four different routing algorithms react to the network congestion with this scenario. The simulation procedure for ns-2 simulator is organized as follows:

1. Time 0: Begin computations for routing table.

2. Time 1: Generate traffic such that every node launches 20 TCP sessions and 10 UDP sessions to every other nodes except its neighbors.

3. Time 10: Start OMP and CTMP functions.

4. Time 100: Finish the simulation.

Traffic is launched after the routing tables of all nodes have stabilized. In order to evaluate the impact of routing, we do not generate self-destined or neighbor-destined traffic. The traffic distribution procedures of OMP and CTMP are initiated at time 10, after network condition stabilizes.

We evaluate both network performance and TCP performance in every simulation run. We estimate the average link utilization and the average packet drop rate for the network performance. For all links, link utilization is estimated every second and averaged over a window of one-second intervals. The average packet drop rate is estimated in a similar way as the average link utilization. We also evaluate goodput and badput for TCP performance. TCP goodput is the rate at which packets are successfully delivered, whereas TCP badput is the packet drop rate. The simulation scenarios described above are applied to obtain all of the results presented in this Chapter.

4.6.1 Comparison with four different routing algorithms

We perform simulations to evaluate four different routing algorithms under congested network conditions. Specifically, the performance of CTMP is compared with that of two existing Internet routing schemes, i.e., conventional PV routing and ECMP, and OMP which is an IETF draft proposal. We demonstrate that CTMP provides the best performance among these four different algorithms by effectively alleviating network congestion.

Network Performance

Figure 4.6 shows the average link utilization on the network for each routing algorithm. CTMP shows the best performance with 72.6 % on average link utilization. OMP estimate the average link utilization of 71.8 %, PV of 71.4 %, and ECMP of 70.3 %. As discussed in [70], ECMP performs the worst among the four algorithms, even worse than the shortest

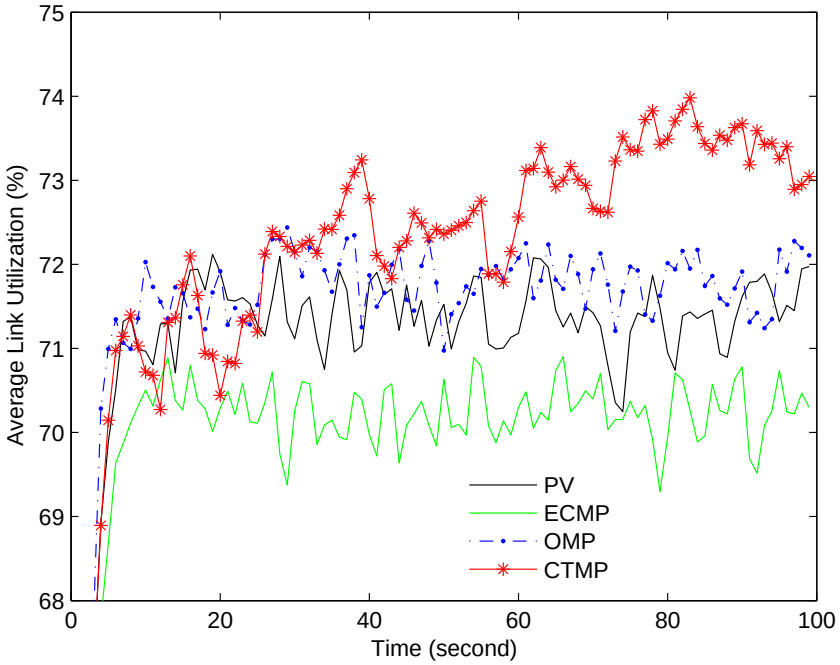


Figure 4.6: Average link utilization for four routing algorithms.

path routing algorithm. ECMP employs multipath routing but distributes traffic packet-by-packet over a multipath route, without taking into account the link conditions. This traffic distribution results in a poor traffic distribution, which causes other links to be congested. Nonetheless, OMP allows unequal traffic distribution over multipath by taking into account the link utilization. OMP traffic distribution overcomes shortcomings caused by SP and ECMP as observed in simulation results. However, OMP might cause network congestion on other paths because multiple nodes respond to congested paths, as described in detail in Chapter 2. CTMP resolves this problem by performing congestion detection only on local link and then triggering the traffic distribution over a multipath route when congestion is detected. From the simulation results, we see that CTMP effectively utilizes links by performing load balancing on the network.

The average packet drop rate for each algorithm is depicted in Figure 4.7. CTMP has the best packet drop rate of 0.173 % (i.e., 216 Kbytes/sec) on average. ECMP has the average packet drop rate of 0.191 % (i.e., 238 Kbytes/sec), SP has a drop rate of 0.191 %

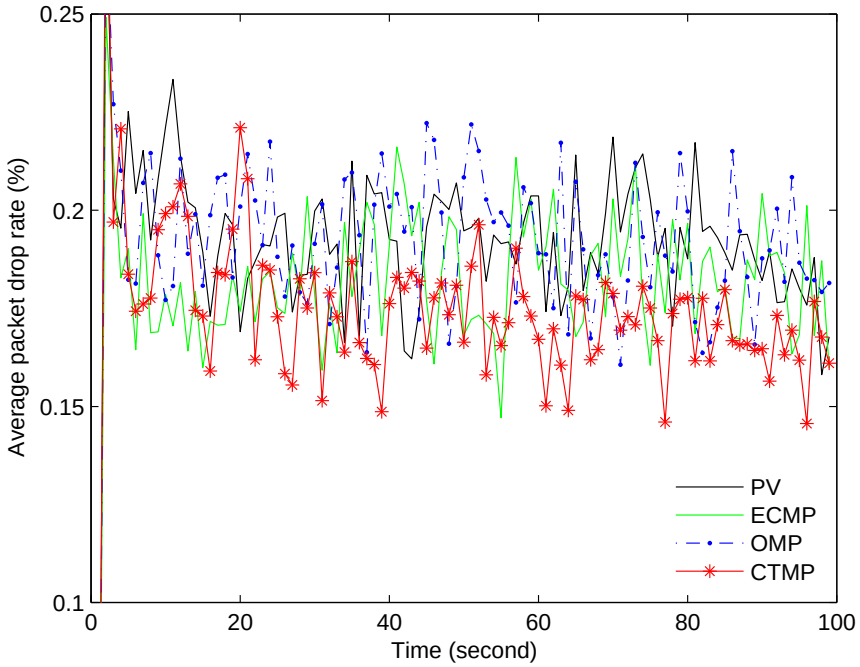


Figure 4.7: Average packet drop rate for four routing algorithms.

(i.e., 239 Kbytes/sec), and OMP has a drop rate of 0.192 % (i.e., 240 Kbytes/sec). The three algorithms, SP, ECMP, and OMP, show similar rates of packet drops. Although OMP and ECMP provide mechanisms to distribute traffic over multipath in order to avoid congestion on the original path, other paths become congested as an unwanted side effect. These mechanisms do not lead to substantial performance improvement in terms of packet drop rate. However, as validated by the simulation results, CTMP reduces the packet drop rate, using the multipath traffic distribution mechanism triggered by local link utilization rather than path utilization applied to OMP.

TCP performance

The performance of TCP goodput and badput are shown in Figure 4.8 and 4.9, respectively. ECMP has the worst performance of 796.3 Mbytes/sec on average in terms of TCP goodput and the highest TCP badput at 6.5 Mbytes/sec among the four routing algorithms. CTMP has a TCP goodput of 827.2 Mbytes/sec on average and a TCP badput of 7.3 Mbytes/sec on

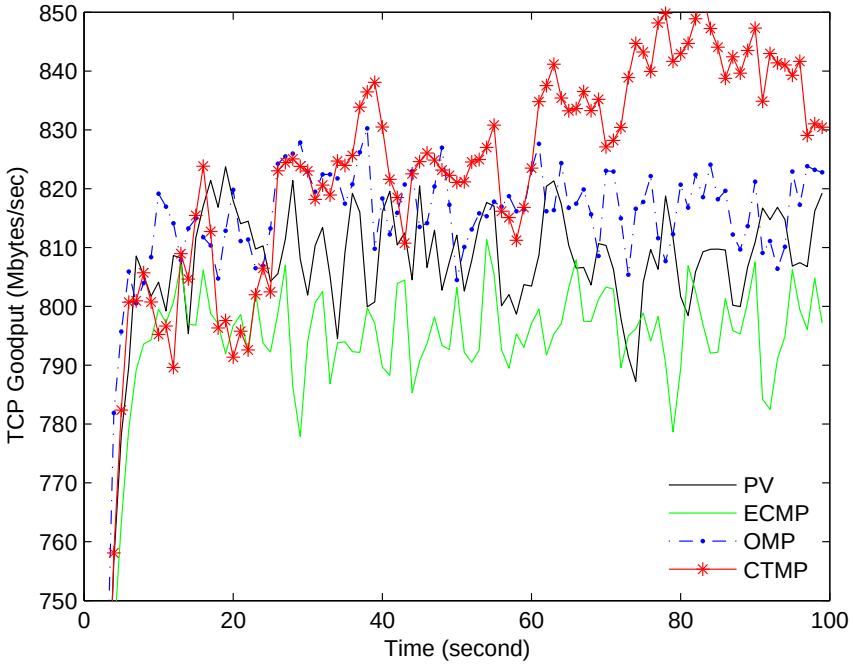


Figure 4.8: TCP goodput for four routing algorithms.

average. OMP and SP have TCP goodputs of 816.3 and 809.3 Mbytes/sec, respectively, and TCP badputs of 7.5 and 7.9 Mbytes/sec, respectively. In ECMP, all packets are forwarded packet by packet over the multipath route such that packets might arrive out-of-order within a flow at the destination. Out-of-order packet arrivals negatively impacts the TCP congestion windows of senders. From the ECMP simulation results, we see that TCP congestion window is not increased due to packet out-of-order delivery which results in poor TCP goodput. ECMP has the lowest TCP badput, but this results in the relatively low TCP goodput, compared with the other algorithms. Unlike ECMP, OMP has a mechanism, which guarantees that packets belonging to a flow are always forwarded on the same path unless a new multipath route has been established or the traffic distribution parameters have been changed. By alleviating the packet out-of-order delivery problem in ECMP, OMP significantly improves TCP goodput compared to ECMP and SP. However, CTMP has the better TCP goodput and badput than OMP, by eliminating the unwanted network congestion caused by load balancing in OMP and increasing the number of multipath.

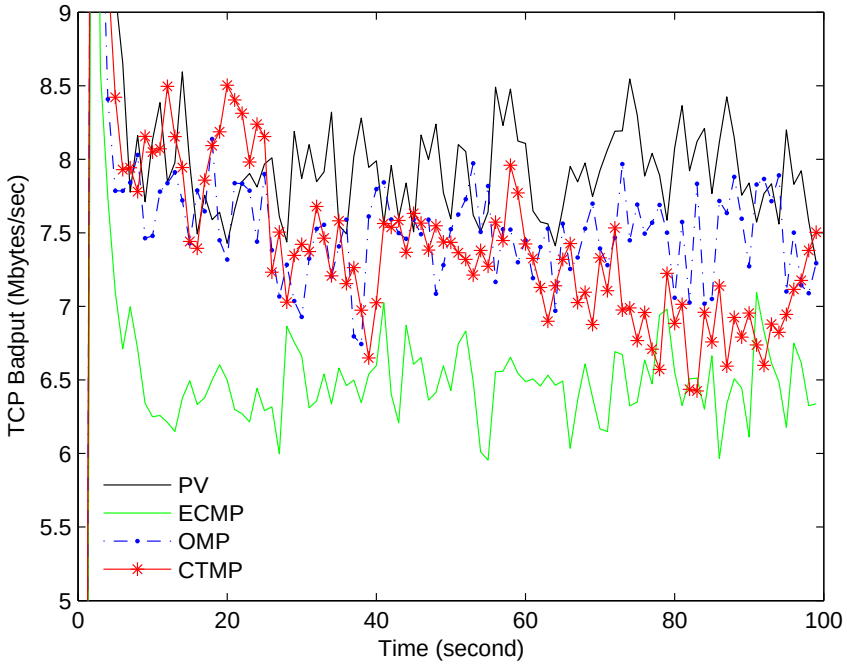


Figure 4.9: TCP badput for four routing algorithms.

From the simulation results for ECMP, we see that the simple packet-by-packet distribution over a multipath route in ECMP can degrade both the network and TCP performance. OMP eliminates this out-of-order packet delivery problem by applying hash-based packet distribution. As a result, OMP shows much better performance than ECMP and SP. However, we also see that OMP introduces the unwanted side effect that other paths can be congested by the load balancing mechanism of OMP. CTMP eliminates this side effect by responding to local congestion and employing an advanced multipath setup and traffic distribution scheme. As can be seen from the simulation results, CTMP has the best network and TCP performance among the four different routing algorithms. Our simulation results demonstrate that CTMP can respond to the network congestion more effectively than existing routing algorithms, resulting in network and TCP performance.

4.6.2 Evaluation with class-c routes

We conduct a simulation study to evaluate the impact of using class-c routes in the CTMP algorithm. We focus on class-0, class-1, and class-2 multipath routes, defined as follows (see Chapter 3):

1. class-0: No common links between a pair of alternative paths.
2. class-1: At most 1 common link between a pair of alternative paths.
3. class-2: At most 2 common links between a pair of alternative paths.

The NSFNET topology shown in Figure 4.5 has 177 multipath routes in class-0, 209 routes in class-1, and 218 routes in class-2. On average, each node establishes 1.89 class-0, 2.12 class-1, and 2.23 class-2 routes to each destination, as described in Chapter 3.

Network performance

Figure 4.10 depicts average link utilization in the network when CTMP is used with each of the three classes of multipath route. Class-1 and class-2 routes show the best performance with an average link utilization of 72.6 % while class-0 routes have a link utilization of 72.4 % on average. Thus, the use of class-1 and class-2 routes results in the better performance than that of class-0 routes. Class-1 and class-2 routes provide more alternative paths than class-0 routes, i.e., 32 paths and 41 paths, respectively. CTMP is designed to alleviate the local link congestion by redistributing the load on the link to a set of alternative paths. If a node finds more alternative paths, it generally has better chance to reduce the load on the congested link. Therefore, the use of class-1 and class-2 routes would tend to alleviate network congestion more effectively than class-0 routes, as demonstrated by our simulation results.

The average packet drop rate corresponding to each multipath class is shown in Figure 4.11. Class-1 and class-2 routes provide the best packet drop rate of 0.173 % (i.e., 216 Kbytes/sec) on average. Class-0 routes results in an average packet drop rate of 0.174 %

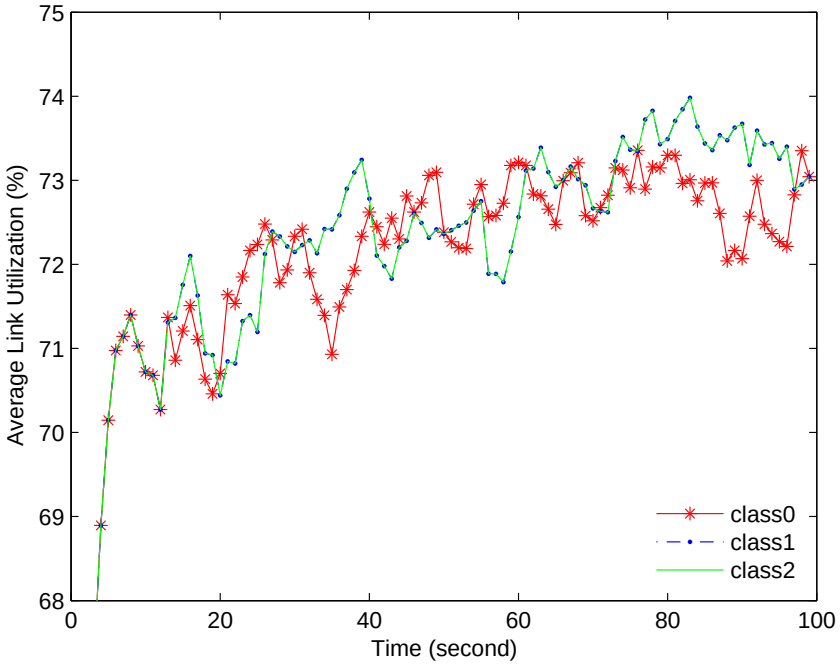


Figure 4.10: Average link utilization for class-c routes.

(i.e., 217 Kbytes/sec). Class-1 and class-2 routes result in sightly better average packet drop rates than class-0 routes. This performance difference is caused by the number of alternative paths corresponding to class-1 and class-2 routes, respectively.

TCP performance

The performance with respect to TCP goodput and badput are illustrated in Figures 4.12 and 4.13, respectively. Class-1 and class-2 routes achieve the best performance of 827.2 Mbytes/sec in terms of TCP goodput and the best performance of 7.32 Mbytes/sec in terms of TCP badput among three classes. Class-0 routes result in a TCP goodput of 823.9 Mbytes/sec and the TCP badput of 7.39 Mbytes/sec on average. Here, we see that the TCP performance is impacted by the number of alternative paths in a multipath route. As class-1 and class-2 routes establishes more alternative paths than class-0 routes, they lead to more efficient utilization of the network resources than class-0 routes. This in turn leads to an improvement in TCP performance.

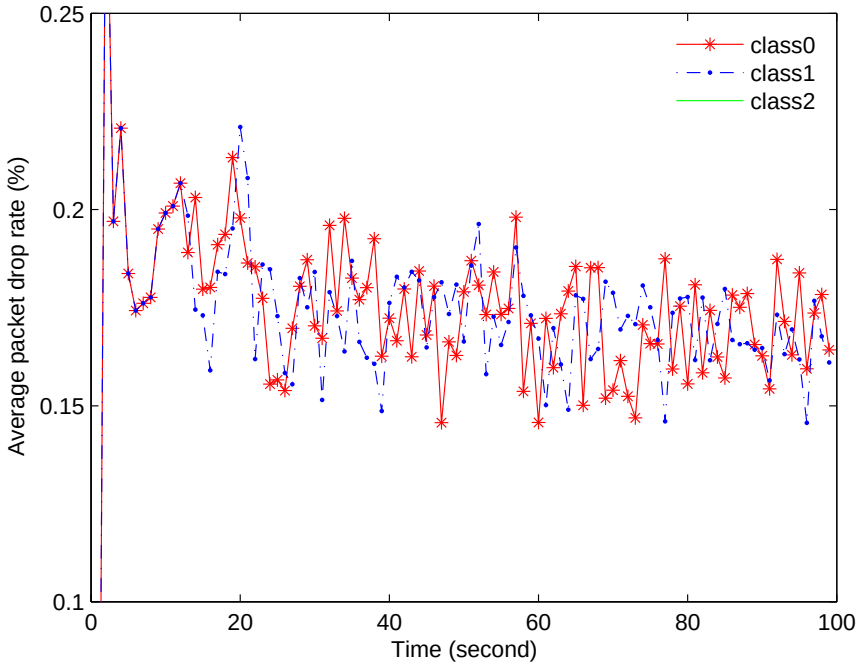


Figure 4.11: Average packet drop rate for class-c routes.

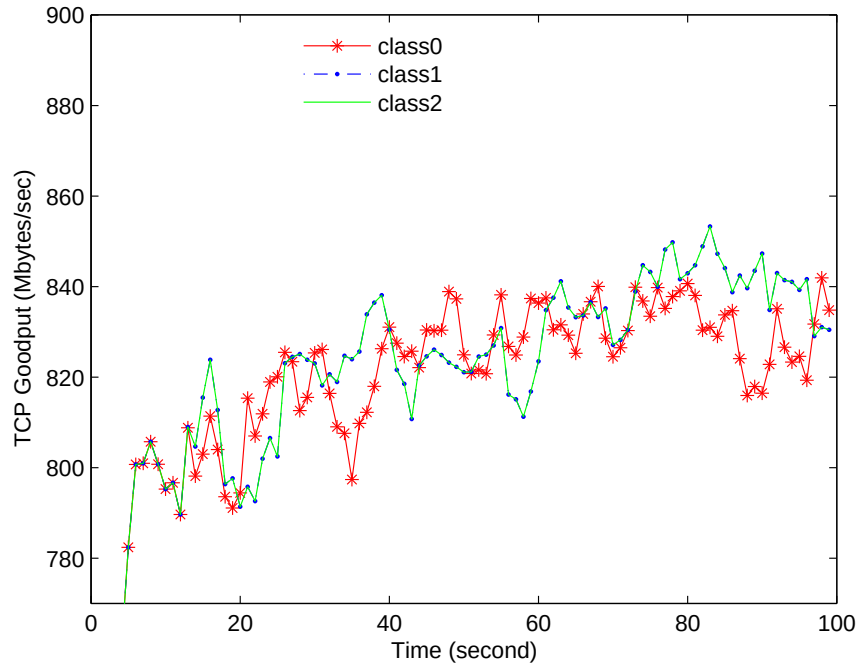


Figure 4.12: TCP goodput for class-c routes.

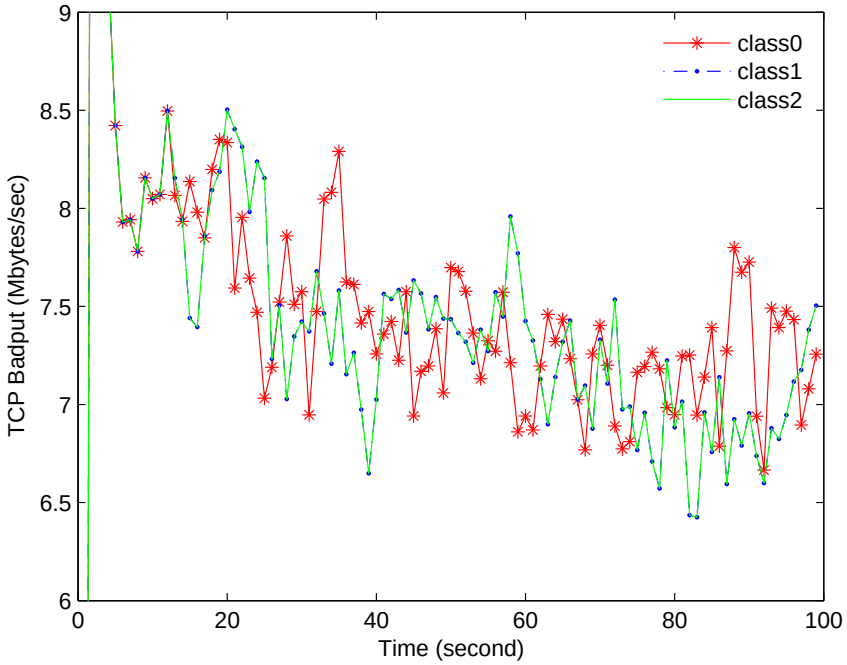


Figure 4.13: TCP badput for class-c routes.

From the simulation results, it is observed that class-1 and class-2 routes achieve better network and TCP performance than class-0 routes. However, notice that class-1 and class-2 routes do not show a remarkable difference in terms of network and TCP performance. Even in simulations, the use of class-3 and class-4 routes shows similar results compared to the use of class-1 and class-2 routes. This result comes from the number of alternative paths associated with each class of multipath route. The numbers of class-1 and class-2 routes are quite similar to each other, and to those of class-3 and class-4 routes. Any shortest path in NSFNET consists of a maximum of four hops from a source to a destination node so that class-1 or higher multipath routes have a similar number of alternative paths. Nonetheless, class-0 routes and class-c ($c > 0$) routes provide quite different numbers of alternative paths. This difference influences on the network and TCP performance. We can conclude that class-1 or higher multipath routes performs better than class-0 routes. For the NSFNET topology, we do not observe a substantial improvement in performance associated with class-c routes as c increases beyond 0.

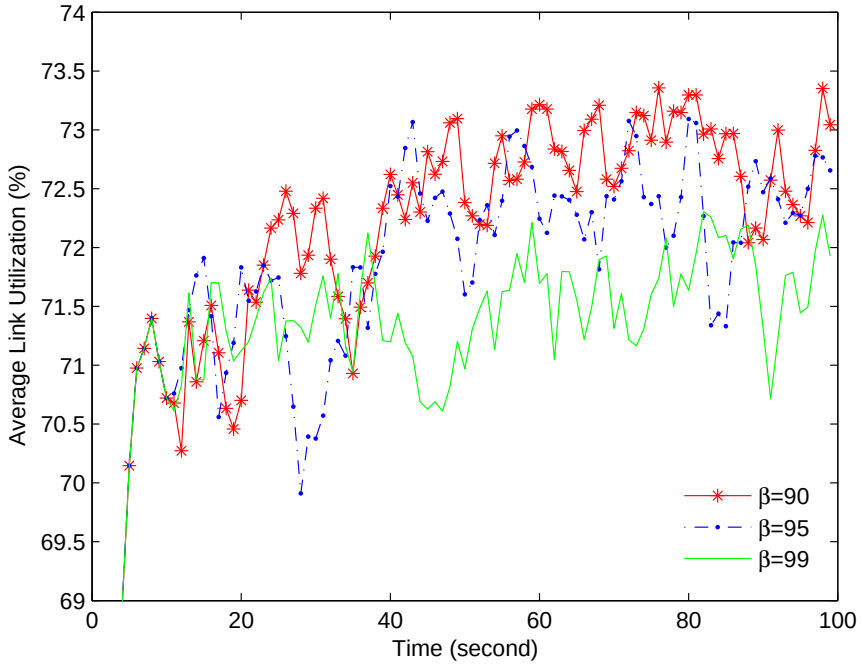


Figure 4.14: Average link utilization for various β.

4.6.3 Estimation of network parameters

Next, we study the effect of the CTMP parameter, the congestion threshold *beta*. We vary the congestion threshold β to determine its best value and characteristics. The congestion threshold β is configured to be 90 %, 95 %, and 99 % in different simulation runs. The other parameters, i.e., the availability threshold γ and the targeted utilization η are set to 85 % and 90 %, respectively.

Network performance

Figure 4.14 shows the average link utilization on the network for different values of β. The setting $\beta = 90$ % shows the best result, with an average link utilization of 72.4 %. When $\beta = 95$ %, the average link utilization is 72.0 % while $\beta = 99$ % results in an average link utilization of 71.5 %.

The performance of average packet drop rate is shown in Figure 4.15. The best packet drop rate of 0.173 % (i.e., 217.1 Kbytes/sec) is achieved when $\beta = 90$ %. When $\beta = 95$ %,

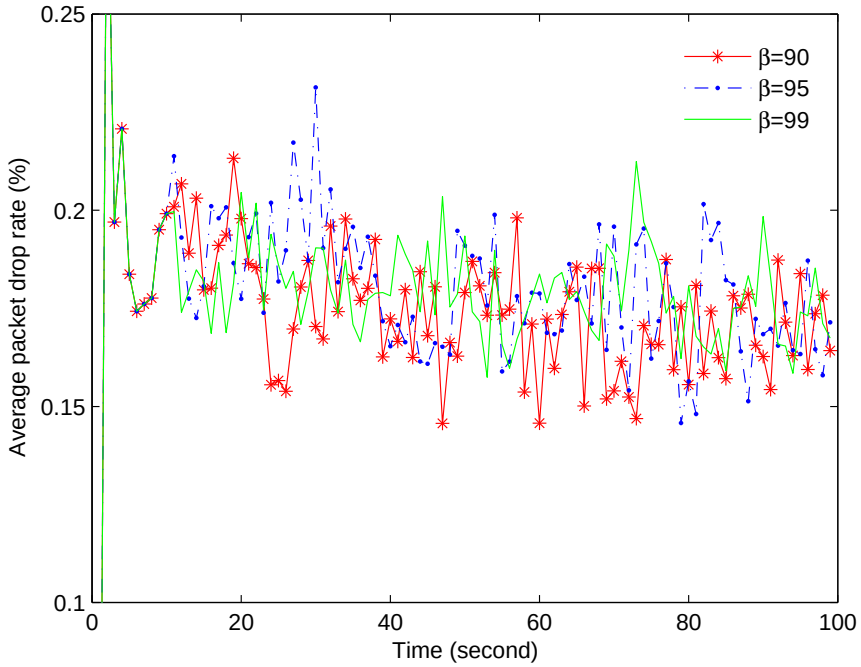


Figure 4.15: Average packet drop rate on various β.

the average packet drop rate is 0.179 % (i.e., 224.3 Kbytes/sec), while $\beta = 99$ % results in an average packet drop rate of 0.179 % (i.e., 223.8 Kbytes/sec). In terms of the average link utilization and the average packet drop rate, CTMP with $\beta = 90$ % performs the best among three different value of β.

TCP performance

The performance in terms of TCP goodput and badput are shown in Figure 4.16 and 4.17, respectively. β of 90 % achieves the best TCP performance with 823.96 Mbytes/sec on TCP goodput and 7.39 Mbytes/sec on TCP badput, on average. β of 95 % and 95 % carry out the TCP goodput of 816.75 Mbytes/sec and 807.98 on average, respectively and the TCP badput of 7.61 and 7.75 Mbytes/sec on average, respectively. As seen in network performance, when β is set to be 90 %, CTMP performs the best.

From overall simulation results obtained by varying β, we observe that the configuration with β of 90 % produces the best results in both network and TCP performance, and that

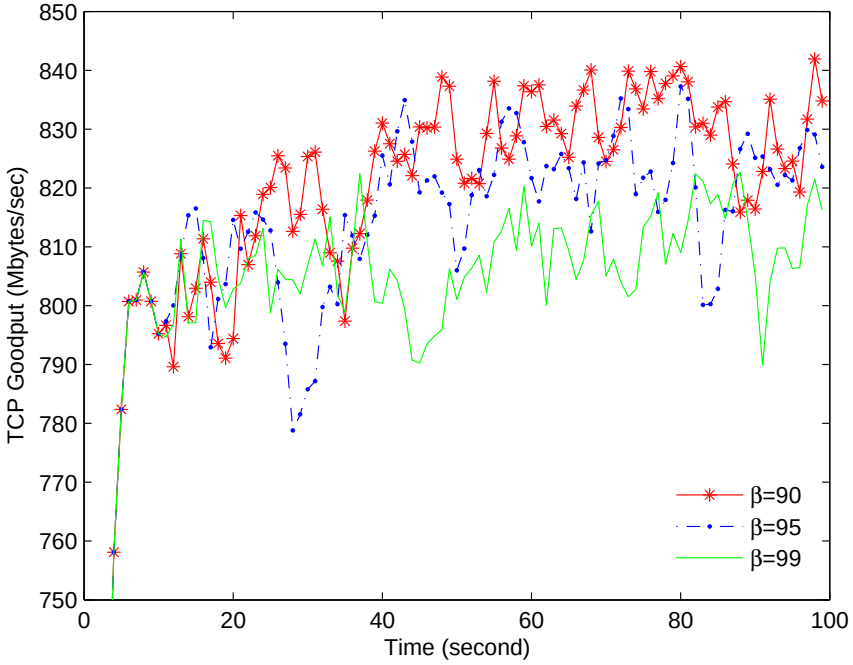


Figure 4.16: TCP goodput for various β.

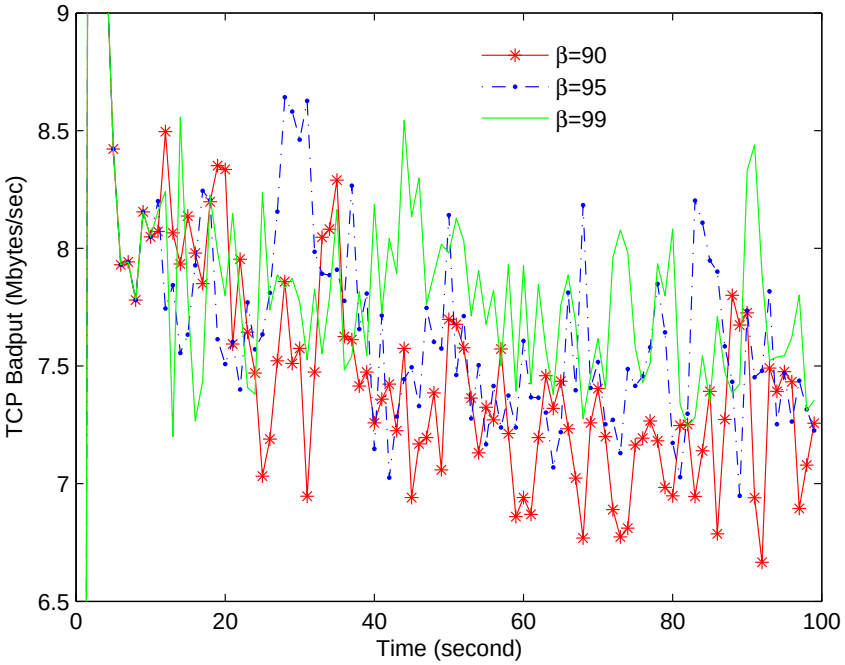


Figure 4.17: TCP badput for various β.

with β of 99 % introduces the worst. According to this observation, we conclude that as β becomes closer to targeted utilization η, CTMP achieves better response to the network congestion. In the network that flows contend each other to get assigned more network resources, it is important to provide network resources to each flow as much as possible. Therefore, β of 90 % close to η demonstrates the best simulation results.

4.7 Conclusion

We proposed a congestion-triggered multipath routing scheme to distribute traffic over a multipath route to avoid network congestion. A link is identified as being congested if the average link utilization exceeds a given threshold. When a node detects an outgoing link as being congested, it applies an algorithm to determine a set of alternative paths, thus forming a multipath route. Traffic is then moved away from the congested link onto the alternative paths according to a traffic splitting function. Our simulation results demonstrate the ability of the proposed multipath routing scheme to relieve network congestion and improve overall network utilization.

Chapter 5: Explicit Rate Signaling and Congestion Control

In this Chapter, we investigate the performance of an explicit rate transport protocol. Here, we adapt the explicit rate signaling protocol of the TIA-1039 [1], which employs in-band signaling standard. According to TIA-1039, the information to compute the explicit rate is carried in the data packet (i.e., in-band) so that this protocol does not require additional out-of-band signaling that can impose additional traffic on the network. We develop and implement an explicit rate transport protocol based on the UDT (UDP-based Data Transfer) protocol [74]. Therefore, we call this new protocol, ER-UDT (Explicit Rate UDP-based Data Transfer).

5.1 Background

We provide some background on the TIA-1039 protocol and the UDT transport protocol.

5.1.1 TIA-1039

TIA-1039 [1] is an in-band QoS signaling protocol, designed by Anagran [75] and standardized by TIA (Telecommunication Industry Association). The protocol can allocate resource on routers for flows via in-band signaling as the flows traverse the network. One difference from existing algorithms such as XCP, RCP, and SABUL/UDT is that TIA-1039 supports Quality-of-Service (QoS) in an IP network, in the form of Guaranteed Rate (GR), Maximum Rate (MR), Variable Rate (VR), and Available Rate (AR) services.

When a flow is initiated by the TIA-1039 protocol, a QoS structure or header is inserted in the header of the first packet. As the packet traverses routers along a path, if the available resources on a particular router do not meet the requirement of this flow, the flow will fail to be established or the values in the QoS header will be modified. After the initial packet

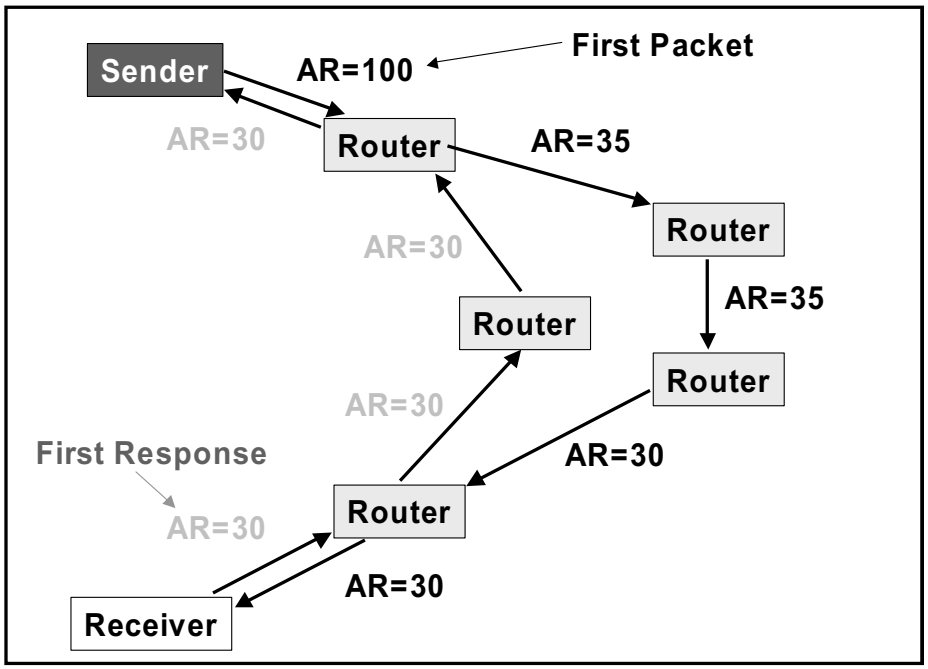


Figure 5.1: Example of TIA-1039 signaling [1].

reaches the destination, feedback in the form of a response packet is sent to the source. The source then adjusts its local QoS settings according to the feedback. Thus, the QoS flow is established in one round-trip time.

Figure 5.1 shows an example of TIA-1039 signaling in which an AR value is signaling from sender to receiver and then back to the sender. The source of the flow sends a first packet with AR=100. When the first packet reaches the first router, its AR value will be reset to 35, because there are only 35 bandwidth units available at this router. When the packet is processed by the third router, its AR values reset to 30. So when the first packet arrives to the receiver, its AR value is 30. Then the receiver sends feedback to the sender, informing it of the final AR setting of this flow. The sender then adjusts its sending rate to 30. We remark that the feedback packet can follow a different path from the forward path. Once a QoS flow is established, not all packets in the flow need to carry the QoS header. A QoS structure needs to be resent by the sender to update the QoS information only after a certain interval of packet arrivals, which is typically 128 packets in the case of TIA-1039.

5.1.2 UDP-base Data Transfer (UDT)

The primary goal of UDT [74] is to utilize the abundant bandwidth over current long haul networks for applications such as computational grids. The fairness issue is also taken into account in the design of UDT. In particular, two of the major fairness objectives are to be independent of round trip time (RTT) and to be TCP-friendly [76, 77]. UDT utilizes UDP for both data and control information.

The sender sends out a data packet every inter-packet interval, which is updated by the rate controller. However, the sender cannot send out the packet if the number of unacknowledged packets exceeds a threshold, which is updated by the flow control algorithm. A retransmission packet has higher priority than a first time packet. The receiver receives and reorders data packets. If the receiver detects packet loss, NAK (Negative Acknowledgement) packet will be sent back reporting the loss. Selective acknowledgement is used in the protocol, which generates an ACK (Acknowledgement)packet after a constant time interval if there are any packet to acknowledge. The receiver also measures the packet arrival rate and the link capacity, which will be sent back together with the ACK packet. The sender sends back an ACK2 packet for each received ACK packet, which is used by the receiver to measure RTT, as well as to decide the next ACK value (i.e., it must be greater than the last received ACK2). The receiver also checks the RTT variance to check if there is a delay-increasing trend. If so, it sends back a delay increasing warning packet to the sender.

UDT consists of two parts: a UDT protocol framework and a UDT congestion control algorithm. We modified the UDT protocol framework in order to implement explicit rate signaling. In addition, we replaced the UDT congestion control algorithm with an explicit rate congestion control algorithm. We chose UDT for the implementation of an explicit rate signaling for two main reasons: (1) To reduce traffic and processing time caused by ACKs. UDT responds to delivered packets by acknowledgement once in a given interval whereas TCP reacts to every delivered packet. (2) UDT provides more information on lost packets than TCP-SACK [56], which has been considered the best mechanism for multiple packet losses. Routers compute the sending rate and feed it back to the sender so that the sender

transmits data at the maximum rate, while avoiding congestion.

5.2 Explicit rate signaling protocol

We developed ER-UDT (Explicit Rate UDP-based Data Transfer), which estimates the accurate sending rate with support of network equipment (e.g., router, switch). ER-UDT achieves higher throughput compared to legacy TCP, over impaired links. For accurate rate estimation, ER-UDT utilizes the QoS signaling mechanisms of TIA-1039. QoS signaling helps to eliminate the slow rate increase in UDT so that sender can reach the maximum sending rate after one round trip time, in order to avoid network congestion. For higher throughput, we improve the retransmission algorithm in UDT. UDT has three transmission phases: data transmission, retransmission, and idle. When the sender buffer becomes full, the sender does not transmit any packets (i.e., idle phase). This happens due to the loss of retransmission packet(s) over an impaired link. We utilize this idle phase to transmit lost packets again (second retransmission) without waiting for the next NAK (Negative Acknowledgement). Second retransmission can reduce the retransmission buffer quickly so that ER-UDT can achieve higher throughput.

5.2.1 Connection establishment

The proposed architecture establishes a connection between source and destination via a three-way handshake (i.e., SYN, SYN/ACK, and ACK), similar to TCP connection establishment. This three-way handshake carries the connection information as well as the explicit rate information so that the sender can obtain the maximum sending rate to avoid network congestion as well as to establish a connection to a destination.

The sender generates and transmits a handshake packet (i.e., SYN) to request a connection establishment to a destination, and to obtain the sending rate computed by routers along a path to destination. Note that this header format is different from that of TIA-1039. The header of the handshake packet includes an ER field as well as information in

TCP header. The sender can set the ER field to its desired rate and the routers along the path to the destination update the ER field. After a handshake packet is delivered to the destination, the destination establishes a connection between the sender and destination pair, and then it transmits a response packet for the handshake packet (i.e., SYN/ACK). This response packet includes the ER field carried by the handshake packet.

TIA-1039 utilizes handshake and response packets to obtain the explicit rate. However, the proposed protocol employs one more step to confirm the ER and round trip time (RTT) values with routers. After the sender receives the response packet from the destination, it sets its sending rate to the ER recorded in the response packet, and then computes a round trip time (RTT). We then form a new ACK packet, called ACK2, to carry the ER and RTT as well as the information recorded in the ACK of legacy TCP. By transmitting the ACK2 packet, the ER and RTT information are signaled to the routers along the path. By confirming the ER information, routers can manage the information on ER more accurately so that they can assign available bandwidth to flows.

For example, suppose that the network consists of a sender, two routers (routers A and B), and a destination, as illustrated in Figure 5.2. All links are bi-directional and have a capacity of 100 Mbps. The sender requests an explicit rate to the network. Assume that router A assigns 100 Mbps to the flow while another router B assigns 50 Mbps for the flow. As a result, the sender receives the ER of 50 Mbps and the sender sets its sending rate to 50 Mbps. This ER value of 50 Mbps is informed to the network by an ACK2 packet. Router A changes the ER for the flow from 100 Mbps to 50 Mbps. As a result, router A can save the difference of these two rates, which can be assigned to another flows. Using the RTT information, the routers can deal with the delay information between routers and the sender so that they can support more accurate and stable rate computation (see Section 5.3).

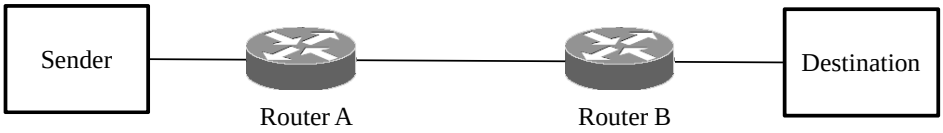


Figure 5.2: Network topology.

5.2.2 Data transfer

The data transfer phase of the proposed protocol is similar to that of UDT. The sender implements a rate-based transmission mechanism which tunes the inter-packet interval whenever the sending rate changes. The sender transmits packets according to rate-based traffic shaping (i.e., a packet is transmitted once every inter-packet interval). If a packet is successfully delivered to the destination, the destination responds to it with an acknowledgement as in legacy TCP.

Retransmission due to a single packet loss

If a single packet loss is detected by the destination, it transmits a duplicate ACK as in TCP. Once the sender receives a duplicate ACK, fast retransmission is implemented. However, the congestion window of the sender is not changed because the sending rate is determined by the network, whereas TCP-Tahoe sets the congestion window size to one and TCP-Reno reduces the window size by half. This is the main difference between legacy TCP and an explicit rate transport protocol.

Retransmission due to multiple packet losses

If multiple packet losses (i.e., two or more packet losses in a flow control window) are detected by the destination, a negative acknowledgement (NAK) packet is utilized in order to explicitly inform the sender of multiple packet losses. The NAK packet carries the list of sequence numbers of lost packets in increasing order. Whenever the sender receives duplicate NAKs, it retransmits the lost packet. By informing the sender of explicit loss information, this mechanism is similar to TCP-SACK [56], but the NAK packet can provide

more information than the TCP SACK field.

In UDT, the NAK packet consists of an array of 32 bit integers. If the sequence numbers of the array are in increasing order, it is considered that all sequence numbers of packets except the last sequence number in the array have been lost. In our proposed protocol, we improve the usage of the last sequence number of the array to acknowledge the corresponding packet, rather than indicating a lost packet as in UDT. This NAK can specify the range of packets that are consecutively lost, as in TCP-SACK. To record sequence numbers of consecutive lost packets, the first bit of the sequence number of the first lost packet is flipped. This modified sequence number is recorded, followed by the last sequence number of consecutively lost packets. For example, the following information might be carried by a NAK packet:

$$00000010, 00000100, 10000110, 00001001, 00001100.$$

The first two numbers indicate that packets 2 and 4 have been lost. In the third number, the first bit is flipped, indicating that packets 6 to 9 (indicated in the third number) have been consecutively lost. The last number indicates that packet 12 has been successfully delivered. According to the information contained in the NAK packet, the sender should retransmit packets 2, 4, 6, 7, 8, and 9.

In order to achieve higher throughput over impaired link, we improve the retransmission algorithm in UDT. In UDT, if retransmitted packet is not successfully delivered to destination, a sender has to stop sending packets and wait for RTO (Retransmission Time Out) time unit, which is so called the idle phase. We utilize the idle phase of UDT to implement the second retransmission, in which the sender transmits all lost packets again during the idle phase. For the second retransmission, the references in [78–80] proposed a Lost Retransmission Detection (LRD) algorithm which can detect the retransmitted packets lost, transmit the lost packets again, and decrease the window size. However, in our algorithm, we do not need to implement the LRD algorithm to identify lost retransmission packet, since our algorithm based on UDT enters the idle phase when the retransmitted packet is lost. During the idle phase, the sender transmits lost packets again but does not change

the window size, since the window size is adjusted by ER control, which is provides as feedback from the routers. This second retransmission phase helps improve the end-to-end throughput as can seen in the simulation results.

5.2.3 Simulation results

We present simulation results that evaluate the throughput performance of four different protocols: Legacy TCP, TCP-ER, UDT, and the proposed protocol ER-UDT under various delay and loss rate conditions. This simulation was carried out on the Emulab testbed [81], on a simple network topology consisting of a sender, two routers (routers A and B), and a destination, as illustrated in Figure 5.2. All links are bi-directional and have a capacity of 100 Mbps. Various losses and delays were assigned to the link between the two routers. However, the other two links are loss-free and have zero delay. Thus, the throughput for each of the four different protocols is measured under various loss rate and delay conditions. For this simulation study, the explicit rate is set to 100 Mbps to fully utilize the link capacity for TCP-ER and the proposed protocol. The simulation results are given in Figure 5.3.

The throughputs of all four protocols become worse as the delay becomes longer and/or the loss rate becomes higher. The proposed protocol (ER-UDT) has the best performance among the four different protocols for various loss and delay settings, while legacy TCP shows the worst performance. Legacy TCP and UDT have lower performance than TCP-ER and the proposed protocol as the loss rate and/or delay increases. This low performance is caused by the implicit rate feedback that the sender has to compute its own sending rate according to the feedback of ACK/NAK from the destination. However, TCP-ER and the proposed protocol employ explicit rate feedback signaling so that the sender can keep transmitting data at the rate assigned by routers. Considering the various delay and loss rate conditions, the best throughput is obtained by the proposed ER-UDT protocol, the second by TCP-ER, the third by UDT, and the worst throughput achieved by legacy TCP.

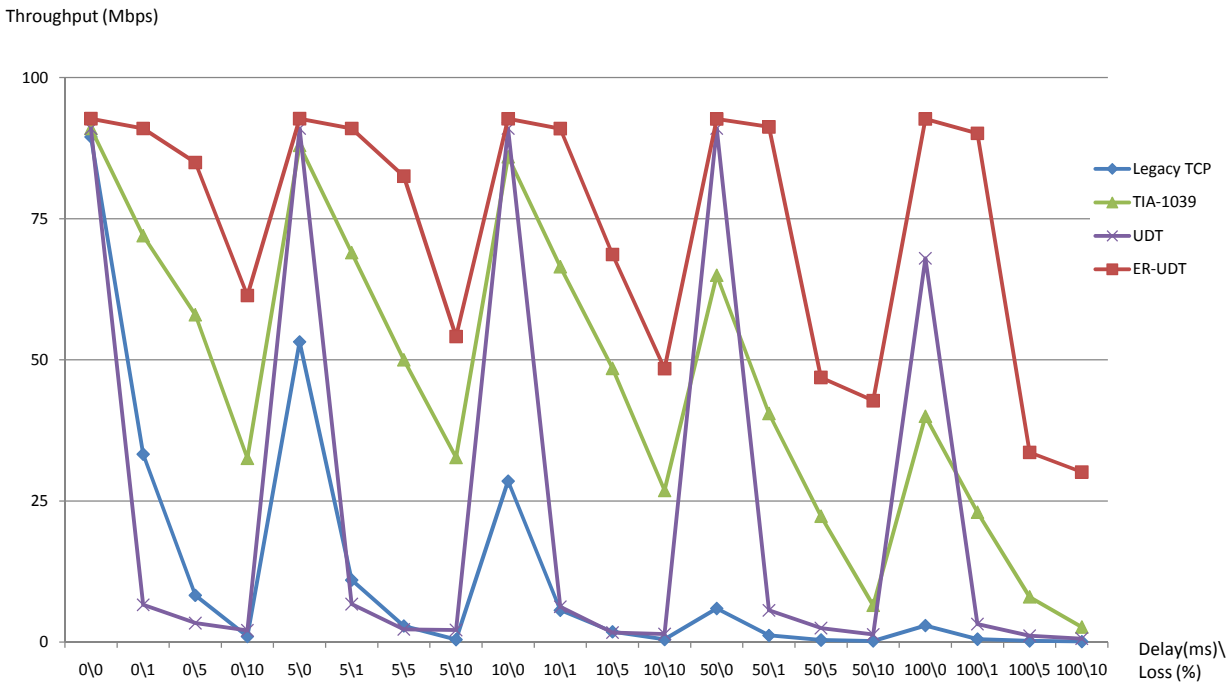


Figure 5.3: Throughput comparison under various delay and loss rate conditions.

5.3 Interoperation with Multipath routing

We discuss an approach to interoperate explicit rate signaling protocol (i.e., ER-UDT) with multipath routing. This is designed to overcome the limitations of current explicit rate congestion control and single path routing protocols. In shortest path routing, when a link/path is congested, the sender can only decrease the rate of the flow. The proposed architecture provides an alternative solution to this problem, in which a new flow is assigned to the best path among multiple paths in terms of achievable throughput. In contrast to other proposed multipath algorithms, the multipath algorithm assigns each flow to a given path. The best path is determined by collecting information from the multipath route and then computing a rate based on the information.

5.3.1 Framework

A multipath route is maintained by the ingress router for flows traversing the routes by using the class-c algorithm discussed in Chapter 3. Whenever a new flow arrives to the ingress router, the router assigns the flow to a "best" path. We define the best path as the path which provides the greatest sending rate among the multipath routes set to a given destination. This best path is determined based on the expected flow throughput (i.e., ER) that would be provided to the new flow and is discussed in detail in Chapter 6.

In order to determine the multipath routes to destination, the routers employs the class-c algorithm, which utilizes the path vector describing the list of intermediate routers along a path. In addition to the path vector in routing table, we introduce one more field, ER, in order to decide which path provides the largest ER in the class-c routes. Whenever a routing table exchange occurs, the value of ER is also exchanged. In order to implement these requirements, we modify the routing table to carry the ER along with Path Vector information. The modified routing table is periodically updated as in conventional routing table update, and based on the updated information, the best path is determined. Whenever the new flow arrives at the router, the new flow is assigned to the best path while existing flows takes the pre-defined path.

For the routing table update function, the path vector is updated by a concatenation function. For example, if node A receives the vector of BD then, it updates to ABD. However, the ER, $r(p)$, is updated using the MIN function as follows:

$$r(p) = \min\{r(e), r(p')\}, \tag{5.1}$$

where e is the new link and p' is the path in advertised routing table. From the example, link e would be AB and path p' be BD.

Figure 5.4 illustrates the operation of the proposed architecture. Suppose that node A is the ingress node, node D is the egress node, and the rest of the nodes are intermediate nodes. The goal is to find the best path from node A to destination D. Each node advertises the path it prefers to get to destination D. For example, destination D advertises its presence in the network with the information of (D, inf); the first information is path vector which the second is explicit rate (ER) of the path which is initially infinity. When intermediate nodes, B and C, receive this information, they update the information and advertise it to their neighbors. Node B updates the path vector to BD and the ER to 20 (say, ER is 20 on the path BD). Node C also updates the path vector to CD and the ER to 10 (assuming that ER is 10 on CD). Finally, node A receives the information from node B and C and then, has the routing information of (ABD, 20) and (ACD, 10), with assumption that paths AB and AC both have an ER of 100. In this case, the best path from node A to destination D would be ABD since ER of path ABD is the greatest among the class-c routes.

5.4 Conclusion

An explicit rate transport protocol, so-called ER-UDT, is proposed to take into account feedback transmission delay and to improve the throughput on impaired link. The proposed protocol is implemented on top of UDT [74] with relatively minor changes. An explicit rate signaling protocol is proposed that confirms ER and RTT information along a path to allow more accurate and stable rate computation. The performance of the proposed explicit rate

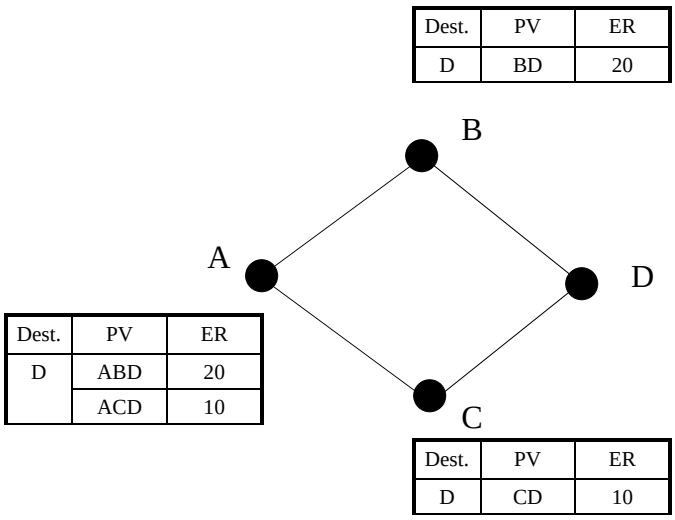


Figure 5.4: Incorporation of ER-UDT with multipath routing.

transport protocol was evaluated on the Emulab network emulator platform [81]. In our simulation, it was demonstrated that the proposed protocol performs much better than existing transport protocols under various delay and loss rate conditions.

Chapter 6: Explicit Rate Computation

The Transmission Control Protocol (TCP) [22] has been the dominant protocol for end-to-end congestion control in the Internet (see [23]). However, TCP has often been found to be inefficient over links with large delays and/or high packet loss rates [25–28]. To solve the problems of TCP, protocols based on explicit congestion control have been proposed, such as eXplicit Control Protocol (XCP) [25, 26], Rate Control Protocol (RCP) [27], and TCP-Explicit Rate (TCP-ER) [28]. In these protocols, congestion control is based on the feedback of explicit congestion information determined by the network elements (e.g., routers or switches) along the path from source to destination. Thus, the sender transmits data at a rate signaled explicitly by the network. If congestion occurs in the network, the network signals a reduced explicit rate to the sender in order to alleviate the congestion.

By contrast, TCP is based on the so-called AIMD (Additive Increase Multiplicative Decrease) algorithm, which increases the transmission rate additively until a timeout indicating that packet loss occurs. In this case, the transmission rate is decreased by a multiplicative factor. AIMD-based congestion control then leads to large oscillations in the sender rate. The explicit congestion protocols may underestimate the effect of the feedback (i.e., the rate) propagation delay between the network and sender or do not take it into account. This in turn may lead to congestion collapse due to outdated rate information. Dynamic rate controllers in [32–35] take into account this propagation delay in the explicit rate computation. These controllers are designed based on queue dynamics at the network elements, such that their congestion avoidance mechanisms are optimized for relatively heavy traffic conditions, but not for light traffic conditions.

In this Chapter, we propose a new explicit rate controller that adjusts the sender's transmission rate according to the network status (e.g., network congestion at the bottleneck). In

contrast to other proposals explicit rate control, the proposed controller takes into account various traffic conditions such as light/heavy traffic on a link as well as propagation delay in the computation of an explicit rate. The proposed controller is designed on basis of control theory. In particular, a Smith Predictor is employed to implement effective compensation for the time delay in the feedback loop. It is also designed to be adaptive to all traffic conditions at a link by taking into account both the volume of traffic and queue dynamics, whereas existing proposals deal with one of those. Further, the proposed controller takes into account the marginally unstable process (i.e., integrating process) that disrupts the convergence of the traffic rate to a target when a load disturbance is present. We introduce several controllers in conjunction with the Smith Predictor in order to stabilize the proposed rate controller. We analyze the stability of the proposed controller and determine the effective control parameters of the proposed controller. The proposed controller can adaptively respond to various traffic conditions, compute the appropriate sending rate to avoid network congestion, and achieve the maximum network utilization.

6.1 Related Work

Explicit rate control algorithms have been intensively studied in the context of ATM networks in order to control ABR (Available Bit Rate) traffic. A simple traffic control scheme, so-called binary feedback scheme, was proposed in [82–86]. This scheme utilizes a one-bit feedback indicating the presence or absence of congestion, which may lead to the rate oscillations, unstable network condition, and even severe network congestion. To overcome this problem, explicit rate algorithms were proposed that compute an explicit sending rate for each flow. A number of studies, including [34, 87–89], have been done to compute a rate for Virtual Circuit (VC) in ATM so, flow is individually maintained at ATM switch and its rate is individually computed.

Explicit rate control algorithm have been proposed for IP networks in order to alleviate network congestion and provide better network throughput in [32, 33, 35, 90–93]. The proposed rate controllers utilize queue dynamics or outgoing link utilization to compute a

sending rate. The rate controller proposed by Benmohamed and Meerkov in [32] was the first attempt to develop an analytic model for the closed-loop congestion controller in order to provide stability and fairness, using a single Proportional and Derivative (PD) controller. However, this controller does not provide good transient response nor good steady-state response [34, 88, 89]. The Benmohamed-Meerkov controller is based on queue dynamics. The algorithm performs well under heavy traffic condition (i.e., the queue at a router is occupied) but responds slowly under light traffic conditions (i.e., the queue is mostly empty). Alternatively, rate-based algorithm could compute the rate according to link utilization. If there is available bandwidth, the rate is increased. However, rate control algorithms based on link utilization do not respond well under heavy traffic conditions since they do not take into account the degree of congestion.

In order to eliminate the effect of time delay on the network, The Smith Predictor has been employed in the design of rate controller in the literature. The Smith predictor is a well known effective compensator for the time delay in feedback control systems and can provide stable performance. It includes a model that can process the time delay in the feedback controller.

Some feedback controllers have a time delay so, they can respond after the time delay. In computer networks, a flow between a source and destination pair also has a propagation delay so, explicit rate controllers work as feedback controllers with time delay. The Smith Predictor includes a feedback structure that can effectively take the delay outside of the feedback loop.

Fig. 6.1 depicts the standard Smith Predictor. It includes a controller $C(s)$, a plant $G(s)$, and a delay process e^{-ds} where d is the time delay. $R(s)$ is the reference (i.e., the target value), $M(s)$ is the measured value as a output, and $Y(s)$ is the error (i.e., the difference between $R(s)$ and the feedback). In order to compensate for the feedback delay, a model of a plant $G_M(s)$, and a delay process $e^{-d_M s}$ is introduced where d_M is the model of delay. If the plant $G(s)$ and the delay d are known, we can design the model of $G_M(s)$ and $e^{-d_M s}$. $G(s)$ and $G_M(s)$ are typically designed to be identical. Thus, the transfer function of the

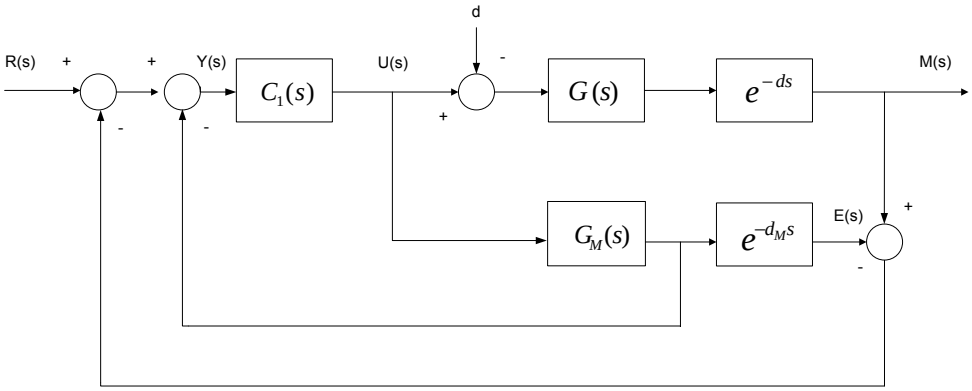


Figure 6.1: Standard smith predictor.

system is given by

$$T(s) = \frac{M(s)}{R(s)} = \frac{G(s)C(s)}{1 + G(s)C(s)} e^{-ds}. \tag{6.1}$$

This model enables to predict the non-delayed output of the plant and move the delay out of the control loop. Therefore, this predictor works as if there were no delay in the system and the resulting closed-loop system would show the behavior of a finite closed-loop system with no delay. The concept of moving the delay out of the control loop applies to the new rate controller. This makes the design of the rate controller much simpler and allows it to work more effectively.

The Smith Predictor has been employed to compute a sending rate in [88,89,93–95]. The feedback control algorithms of Mascolo *et al.* in [88,94] exploit the Smith Predictor for ATM congestion control in which source rates are adjusted according to VC (Virtual Channel) queue lengths at intermediate nodes along the path. However, they have a disadvantage that more computational and informational load is placed on the ATM switch due to per-VC FIFO queue maintenance [89]. Mascolo in [89] extends these algorithms to TCP. It can determine the adequate sending window size at router for each TCP flow, not at the end system. Cavendish in [95] describes the details of theoretical approach for the algorithm in [94]. However, these algorithms are developed to compute per-flow based sending rate.

Mascolo in [93] proposed an algorithm that can take care of multiple flows in a link. It also modified the Smith Predictor to consider the feedback delay. However, these algorithms focus on the queue dynamics to control the rate so that they may not be able to quickly respond to low utilization (i.e., queue length is zero and link utilization is less than 100 %). Furthermore, they employ an integrating process as a plant, and a proportional controller to deal with error (i.e., difference between reference value and measured value). The integrating process disrupts the stability of the controller when load disturbance is present [96–100]. Thus, the computed rate may not converge into a constant rate.

We propose a rate controller that can respond quickly to both heavy and low traffic condition by taking into account both queue dynamics and link utilization. The proposed controller employs the Smith Predictor in order to take into account multiple delays rather than a single flow. We modify the Smith Predictor to stabilize the proposed rate controller even if load disturbance is present, by introducing several controllers in addition to the Smith Predictor. The proposed controller is stable, has a fast transient response, and eliminates steady-state error.

6.2 Network Model

A flow between a source and destination pair traverses one or more network elements (e.g., routers, switches) in the network. The proposed rate controller resides in each router, which calculates a desirable rate based on both the outgoing link utilization and queue length. In TCP-ER, this rate is signaled to the source by including a QoS header in one every 128 packets.

Figure 6.2 depicts a bottleneck queue in a router that can identify incoming flows and compute the sending rate, such as SAFIRE (Software Adaptive Flow-Intelligent Router) router [101–103] and Anagran flow router of FR-1000 [104]. There are N different incoming flows identified as $f_1, f_2, \ldots, f_N$. These flows are relayed through the router at the maximum transmission rate of C, which is the capacity of the outgoing link. However, if the volume

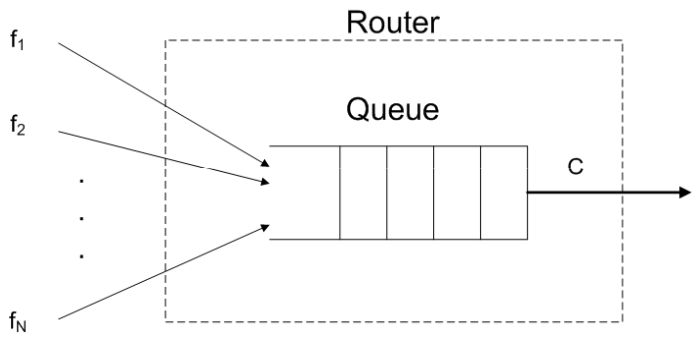


Figure 6.2: Bottleneck queue of a router with N incoming flows.

of incoming flows is greater than C, the excess volume of traffic is stored in the queue.

Let $q(t)$ denote the length of the bottleneck queue at time t and let Q denote its maximum value, due to the finite size of the buffer. The service discipline of the queue is First-Input First-Output (FIFO). Moreover, the sending rate of source i is defined by $r_i(t)$. Each flow f_i may have different time delay (i.e., round trip time) between the associated source and destination pair, which we denote by d_i.

The dynamics of the queue in the case of N flows with varying delays is, in general, given by

$$q(t) = Sat_Q \left\{ \sum_{i=1}^{N} r_i(t - d_i) - C \right\}, \tag{6.2}$$

where

$$Sat_a(z) = \begin{cases} 0, & \text{if } z < 0, \\ a, & \text{if } z > a, \\ z, & \text{otherwise.} \end{cases}$$

Rate controllers based on the queue dynamics, e.g., [32–34], utilize this equation. They are designed to determine the rate such that the queue length is kept close to a target queue length. Since such controllers focus on queue dynamics so, they may not perform well when the volume of incoming traffic is less than the capacity C of the outgoing link

(i.e., under light traffic conditions). Thus, we consider the utilization of outgoing link in addition to queue dynamics so that the controller can effectively respond to both heavy and light traffic conditions. More precisely, we define a heavy traffic condition as occurring when the incoming traffic is greater than the capacity of the outgoing link. In this case, excess traffic will be stored in the queue. A light traffic condition occurs when the volume of incoming traffic is less than or equal to the capacity of the outgoing link. In this case, the queue does not grow.

We slightly modify Eq. (6.2) in order to take care of both queue dynamics and outgoing link utilization at a time. The proposed system model is given by

$$T(t) = Sat_{C+Q}\left\{q(t) + \sum_{i=1}^{N} r_i(t - d_i)\right\}, \tag{6.3}$$

where $T(t)$ is the total traffic available for processing at the outgoing link at time t and upperbounds by $C + Q$. The objective of the controller is to determine $r(t)$ such that the total traffic at time t satisfies

$$T(t) = C + q^0, \tag{6.4}$$

where $q^0 < Q$ is the target queue length in the controller, which can be set at the discretion of network administrator. Based on this system model, we shall design new rate controller to manage the queue occupancy and the traffic volume. This results in an efficient rate control mechanism that prevents from the queue overflow and avoids low link utilization.

6.3 Rate Controller

We propose a new rate controller based on the Smith Predictor, which takes into account the rate feedback delay. The proposed controller can achieve a near zero queue deviation, and prevent from buffer overflow and low utilization of the outgoing link. Every source is provided with the same sending rate such that the unfairness of TCP to far away sources

is avoided.

The main advantage of Smith predictor method is to eliminate time delay from the characteristic equation of the closed loop so that the controller design becomes much easier and provides stabilized performance. However, the Smith Predictor is not adequate for the system which includes an unstable or integrating process (i.e., marginally stable process). Therefore, we modify the Smith Predictor to stabilize the system, especially the integrating process.

6.3.1 Controller structure

The proposed modified Smith Predictor structure is depicted in Fig. 6.3. This new controller works as an effective time delay compensator for the integrating process which disrupts the stability of the proposed controller. The new structure is a relatively simple modification of classical Smith Predictor.

Here, $R(s)$ is the traffic (i.e., $C+q^0$) expected to be processed at the controller and $Y(s)$ is the error (i.e., the difference between $R(s)$ and the feedback). The measured volume of traffic on the outgoing link is modeled by $M(s)$, which is described by theoretical processes of $G(s)$ and $\sum_{i=1}^{N} e^{-d_i s}$ to estimate the volume of traffic at the outgoing link as described in Eq. (6.3). Then, $M(s)$ is measured at the outgoing link directly, and utilized as feedback in the proposed controller so as to compute the sending rate that can match to targeted volume of traffic, $R(s)$. If the value of $M(s)$ is the same as the target volume of traffic $R(s)$, this feedback is canceled by $E(s)$ which is the volume of traffic modeled by the Smith Predictor. The quantity of $E(s)$ requires the value of delay of each flow for accurate modeling. We assume that this delay information is carried by the in-band signaling protocol.

The disturbance caused by the bottleneck link is represented by $D(s)$. Even if the present router determines a high sending rate for a given flow, the next router with a bottleneck link may set smaller rate. Consequently, the present router obtains flow with a smaller rate after a round trip time. This disrupts the process of accurate rate computation. Thus, $D(s)$ models the difference of those rates as a disturbance. However, we assume that it is

impossible to measure $D(s)$ as in [89]. Thus, it is important to show that $D(s)$ does not affect the steady-state behavior of the system, which will be shown by stability analysis in Section 6.3.3.

The proposed rate controller introduces the multiple delays because multiple flows arrive to the router. Thus, it is important, as in the standard Smith Predictor, to model a process of $G_M(s)$ and a delay process of $\sum_{i=1}^{N} e^{-d_i^M s}$ in order to minimize the error between $M(s)$ and $E(s)$, where $M(s)$ is the traffic measured at the outgoing link and $E(s)$ is the traffic estimated by the model of $G_M(s)$ and $\sum_{i=1}^{N} e^{-d_i^M s}$. Typically, $G_M(s)$ is set to be identical to $G(s)$ in the standard Smith Predictor. The output of $G(s)$ is $F(s)$, the Laplace transform of the sending rate $f(t)$, while the output of $G_M(s)$ is $F_M(s)$, a model of the sending rate. For the model of multiple delays given by $\sum_{i=1}^{N} e^{-d_i^M s}$, the signaling packet transmitted by the sender is used to inform the router of each sender's round trip time (RTT). The ER-UDT (Explicit Rate UDP-based Data Transfer) protocol proposed in Chapter 5 implements signaling that carries these delay information.

We also introduce three different controllers represented by $C_1(s)$, $C_2(s)$, and $C_3(s)$, respectively, which are the main components of the proposed rate controller in addition to the Smith Predictor. The controller $C_1(s)$ is designed to regulate the feedback error, while $C_2(s)$ is used to stabilize the integrating process $G(s)$, and $C_3(s)$ rejects the disturbance $D(s)$. If $C_2(s)$ and $C_3(s)$ are removed from the proposed controller, the standard Smith Predictor is obtained.

6.3.2 Design of controller

Under the assumption that the model parameters are exactly matched to the process, the set point (or so-called reference or target) and disturbance responses are given by

$$M(s) = T_R(s)R(s) + T_D(s)D(s), \tag{6.5}$$

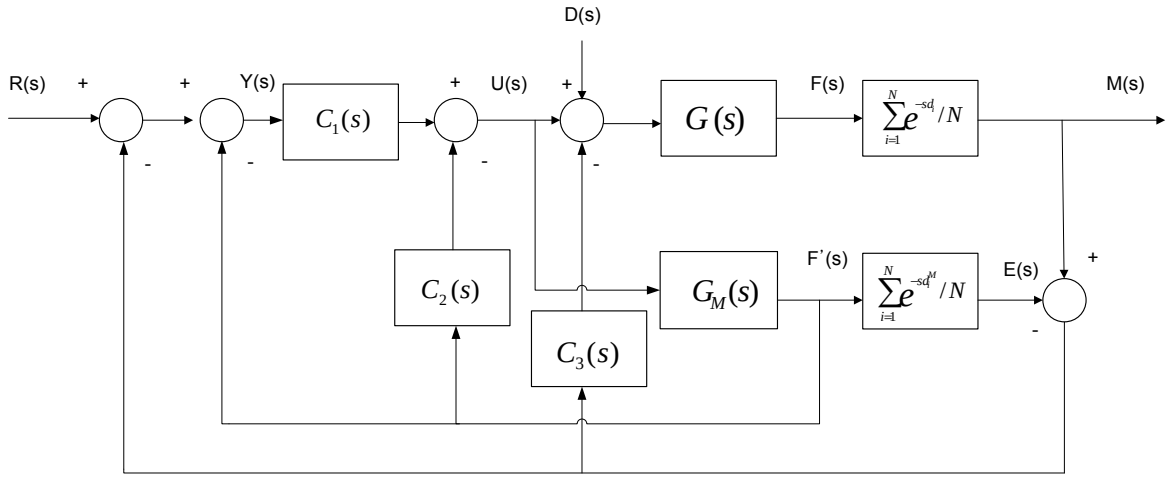


Figure 6.3: The proposed rate controller.

where

$$T_R(s) = \frac{G_M(s)C_1(s)}{1+G_M(s)(C_1(s)+C_2(s))}\sum_{i=1}^{N} e^{-d_i^M s}/N, \tag{6.6}$$

$$T_D(s) = \frac{G_M(s)[1+G_M(s)(C_1(s)+C_2(s)-C_1(s)\sum_{i=1}^{N} e^{-d_i^M s}/N)]}{[1+G_M(s)(C_1(s)+C_2(s))][1+C_3(s)G_M(s)\sum_{i=1}^{N} e^{-d_i^M s}/N]}$$

$$\times \sum_{i=1}^{N} e^{-d_i^M s}/N. \tag{6.7}$$

The transfer function for the set point response in Eq. (6.6) shows that the set point response is affected by only the parameters of the controllers $C_1(s)$ and $C_2(s)$. Those parameters may be obtained by utilizing a model of the delay-free part of the process. The transfer function of the disturbance response in Eq. (6.7) indicates that the controller $C_3(s)$ influences the disturbance response but does not affect the set point response.

The controllers $G_1(s)$ and $G_2(s)$, and a load disturbance rejection controller $G_3(s)$ have

independent responsibilities for the system performance. Thus, the set point and disturbance responses are decoupled, which is a dominant advantage of the proposed controller. That is, both the set point and load disturbance response can be independently optimized. The proposed controller includes an integrating process, which disrupts the stability of the controller when load disturbance is present. Therefore, it is important to analyze the stability of the proposed controller. We first define each process. The integrating process with multiple dead-times (i.e., time delay) can be given by

$$P(s) = G(s)\sum_{i=1}^{N} e^{-d_i s}/N \tag{6.8}$$

$$= \frac{1}{s}\sum_{i=1}^{N} e^{-d_i s}/N, \tag{6.9}$$

where N is the number of flows, d_i is the dead time of flow i, and $G(s)$ is the delay-free part of the process $P(s)$.

The process $G_M(s)$ is set to be identical with $G(s)$. Thus, it is described as

$$G_M(s) = G(s) = \frac{1}{s}. \tag{6.10}$$

The controllers $C_1(s)$, $C_2(s)$, and $C_3(s)$ are a Proportional and Integral (PI) controller, a Proportional (P) controller, and a Proportional and Derivative (PD) controller, respectively. They are given by

$$C_1(s) = K_P + \frac{K_I}{s} = K_P(1 + 1/sT_I), \tag{6.11}$$

$$C_2(s) = K_Q, \tag{6.12}$$

$$C_3(s) = K_R + sK_D = K_P(1 + sT_D), \tag{6.13}$$

where K_P, K_Q, and K_R are the proportional parameters in each controller that are used to control system response time, K_I is the integral parameter in $C_1(s)$ that enables to eliminate steady-state error, and K_D is the derivative parameter in $C_3(s)$ that helps fast convergence to steady-state.

For the set point response, substituting Eq. (6.6) from Eqs. (6.11), (6.12), and (6.13), the set point transfer function is given by

$$T_R(s) = \frac{(K_P s + K_I)}{s^2 + (K_P + K_Q)s + K_I} \sum_{i=1}^{N} e^{-d_i^M s}/N \tag{6.14}$$

$$= \frac{K_P(T_I s + 1)}{T_I s^2 + (K_P + K_Q)T_I s + K_P} \sum_{i=1}^{N} e^{-d_i^M s}/N. \tag{6.15}$$

Substituting Eq. (6.7) and from Eqs. (6.11), (6.12), and (6.13), we can obtain the load disturbance transfer function as

$$T_D(s) = \frac{s^2 + (K_P + K_Q)s + K_I - (K_P s + K_I)\sum_{i=1}^{N} e^{-d_i^M s}/N}{(s^2 + (K_P + K_Q)s + K_I)\left(s + (K_R + K_D s)\sum_{i=1}^{N} e^{-d_i^M s}/N\right)}$$

$$\times \sum_{i=1}^{N} e^{-d_i^M s}/N, \tag{6.16}$$

$$= \frac{s^2 + (K_P + K_Q)s + K_P/T_I - K_P(s + 1/T_I)\sum_{i=1}^{N} e^{-d_i^M s}/N}{(s^2 + (K_P + K_Q)s + K_P/T_I)\left(s + K_R(1 + sT_D)\sum_{i=1}^{N} e^{-d_i^M s}/N\right)}$$

$$\times \sum_{i=1}^{N} e^{-d_i^M s}/N. \tag{6.17}$$

However, for the load disturbance, reference [98] shows that even if the process is an integrator, the following relations is always true:

$$\lim_{t\to\infty} d(t) = d. \tag{6.18}$$

Thus, the signal $d(t)$ is an estimate of the constant input load disturbance d. However, we assume that it is impossible to measure d. It is important to model a controller that can completely reject any $d(t)$. In the proposed controller, $T_D(s)$ enables the load disturbance rejection.

6.3.3 Analysis of Controller Stability

Classical control theory enables us to design new controllers whose performance is analytically evaluated rather than rely on simulations. In order to analyze the performance of the controller, the Laplace transform is commonly utilized so this approach is applied to the analysis of the proposed controller.

In order to ensure the stability of the controller, the controller should guarantee that the output of traffic is always equal to the target rate in steady-state, which is given by

$$m(t) = C + q^0, \tag{6.19}$$

where $m(t)$ is the time domain form of $M(s)$. We analyze the stability of the controller under this condition. However, the stability condition of rate controllers in [89, 93] suggests that the output is less than or equal to the target rate as follows:

$$m(t) \leq q^0, \text{ for } t > 0, \tag{6.20}$$

where q^0 is the bottleneck queue capacity, which guarantees that this queue is always bounded (i.e., no packet loss). This condition provides the upper bound for the queue occupancy but does not guarantee an error-free condition in steady-state. We propose a new controller that can guarantee the steady-state error-free condition by rejecting load disturbance using the PD controller of $C_3(s)$.

We analyze the proposed controller in terms of the set point and disturbance responses

at steady-state. The Laplace transform of $m(t)$ is given from Eq. (6.5) as

$$\begin{aligned} M(s) &= T_R(s)R(s) + T_D(s)D(s) \\ &= M_R(s) + M_D(s). \end{aligned} \tag{6.21}$$

Transforming these equations to the time domain, we have

$$\begin{aligned} m(t) &= t_r(t)r(t) + t_d(t)d(t) \\ &= m_r(t) + m_d(t). \end{aligned} \tag{6.22}$$

We first analyze the set point response $T_R(s)$ in steady-state. The set point response $T_R(s)$ is given in Eq. (6.14) as

$$T_R(s) = \frac{(K_P s + K_I)}{s^2 + (K_P + K_Q)s + K_I} \sum_{i=1}^{N} e^{-d_i^M s}/N. \tag{6.23}$$

Utilizing this equation, we obtain the set point response at steady-state which is given by

$$\lim_{s \to 0} T_R(s) = 1. \tag{6.24}$$

Thus, we obtain $m_r(t)$ as follows

$$m_r(t) = C + q^0. \tag{6.25}$$

We also analyze the load disturbance response at steady-state as analyzed in the set point response. In this analysis, we show that the load disturbance is completely rejected which is assumed to be impossible to measure. The load disturbance response $T_D(s)$ is

described in Eq. (6.16) as

$$\begin{aligned} T_D(s) &= \frac{s^2 + (K_P + K_Q)s + K_I - (K_P s + K_I)\sum_{i=1}^{N} e^{-d_i^M s}/N}{(s^2 + (K_P + K_Q)s + K_I)\left(s + K_R(1 + sT_D)\sum_{i=1}^{N} e^{-d_i^M s}/N\right)} \\ &\times \sum_{i=1}^{N} e^{-d_i^M s}/N. \end{aligned} \tag{6.26}$$

From this equation, we find the load disturbance response at steady-state, which can be given by

$$\lim_{s \to 0} T_D(s) = 0. \tag{6.27}$$

Therefore, the load disturbance at time domain, $m_d(t)$, becomes

$$m_d(t) = 0. \tag{6.28}$$

This property of the proposed controller rejects the load disturbance which is not modeled in the controller so that it is possible to stabilize the proposed controller which includes the integrating process.

From Eqs. (6.25) and (6.28), we can conclude that

$$\begin{aligned} m(t) &= m_r(t) + m_d(t) \\ &= C + q^0, \end{aligned} \tag{6.29}$$

which shows that the proposed controller always matches the output of traffic to the set point (i.e., targeted traffic) in steady-state. Thus, it is shown that the proposed controller can eliminate steady-state error that can occur in the controller, which includes the integrator process when the load disturbance is introduced.

6.3.4 Controller Parameters

We analyze the transient response of the proposed controller in order to obtain optimal parameters of the controllers, $C_1(s)$, $C_2(s)$, and $C_3(s)$. These parameters determine the behavior of the transient response in the proposed controller, that is, they can help improve the speed of the system response and provide fast convergence to the steady-state, and enable to eliminate the error (i.e., mismatching the output traffic to the set point). It is important for the proposed controller to approach to the steady-state in short time without making network congestion, in addition to ensure the stability of the controller at steady-state.

The parameters of the main controllers, $C_1(s)$ and $C_2(s)$, may be determined using a model of the delay-free part of the plant, which is the beneficial feature of the Smith Predictor. We define the delay-free part of transfer function $T_R(s)$

$$T_{RF}(s) = \frac{(K_P s + K_I)}{s^2 + (K_P + K_Q)s + K_I}. \tag{6.30}$$

The transfer function $T_{RF}(s)$ is a second-order system which can be modeled in one of following two general forms:

$$T(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}, \tag{6.31}$$

and

$$T(s) = \frac{\omega_n s/\alpha\xi + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}, \tag{6.32}$$

where the parameter ξ is the damping ratio that can control the rate oscillation, and the parameter ω_n is the undamped natural frequency that helps to control the response time of the system. Each of these forms may show different types of the system response according to the poles of the system (i.e., real and unequal, real and equal, complex, or purely imaginary).

We consider only the response of stable systems whose poles have the negative real parts.

In addition, we focus on the general form of Eq. (6.32) since our system has one zero and two poles. In this system, the zero is placed at $s = -\alpha\xi\omega_n$. The parameter α controls the rate of rise (i.e., the system response time) or decay of the system response, that is, the parameter α controls the damping of the system, which is called the damping factor. If the parameter α is large, the zero is placed far to the left of the poles. Thus, the zero will have little effect on the system response because the coefficient of the s term is so small. Reference [105] shows that the zero will have little effect on the system if the parameter α is set to greater than 3, but as the parameter α decreases below 3, it has an increasing effect, especially when the parameter α is less than 1. Thus, we choose the parameter α as 3 in the proposed controller. This property of the proposed controller enables a fast system response. However, when the parameter ξ is equal to 1, the oscillations disappear and the system is said to be critically damped. Under this condition $\alpha = \xi\omega_n$. We can set the parameter ξ to be

$$\xi = \frac{\alpha}{\omega_n}. \tag{6.33}$$

When the parameter ξ is less than 1, the system becomes underdamped and when the parameter ξ is greater than 1, the system is overdamped. We choose the parameter ξ as 1 to minimize the oscillations on the system. This property of the proposed controller prevents from the queue overflow.

Comparing Eq. (6.30) with the generalized form of second-order system defined in Eq. (6.32) with the parameters, $\alpha = 3$ and $\xi = 1$, we obtain

$$\omega_n^2 = K_I, \tag{6.34}$$

$$2\omega_n = K_P + K_Q, \tag{6.35}$$

$$\frac{\omega_n}{3} = K_P. \tag{6.36}$$

It is well known that a large value of K_P increases the initial control effort therefore, K_P is constrained to unity. From Eqs. (6.34), (6.35), and (6.36), we finally obtain the optimal

parameters of the proposed controller as follows:

$$K_I = 9K_P^2, \tag{6.37}$$

$$K_Q = 5K_P. \tag{6.38}$$

The parameters of the controller, $C_3(s)$, may be determined using a model of the delay-free part of the plant as seen in the controllers of $C_1(s)$ and $C_2(s)$. The delay-free part of the transfer function $T_D(s)$ is given by

$$T_{DF}(s) = \frac{s^2 + (K_P + K_Q)s + K_P/T_I - K_P(s + 1/T_I)\sum_{i=1}^{N} e^{-d_i^M s}/N}{(s^2 + (K_P + K_Q)s + K_P/T_I)\left(s + K_R(1 + sT_D)\sum_{i=1}^{N} e^{-d_i^M s}/N\right)}. \tag{6.39}$$

The stability of the controller $C_3(s)$ depends on the roots of the characteristic equation

$$\left(s^2 + (K_P + K_Q)s + K_P/T_I\right)\left(s + K_R(1 + sT_D)\sum_{i=1}^{N} e^{-d_i^M s}/N\right) = 0. \tag{6.40}$$

Note that we already obtained the control parameters in the left term in Eq. (6.40). Thus, we focus on the new characteristic equation which is the second term inside the large parameters of Eq. (6.40) as follows:

$$1 + F(s) = 0, \tag{6.41}$$

where

$$F(s) = \frac{K_R(1 + sT_D)}{s}\sum_{i=1}^{N} e^{-d_i^M s}/N. \tag{6.42}$$

This form of characteristic equation is analyzed in [99], which we utilize to analyze the optimal parameters of $C_3(s)$. We introduce the parameter Φ_{PM} which is the phase margin

of Eq. (6.41). The phase margin is a key parameter for the stability of the controller $C_3(s)$ since the inaccurate setting of phase margin results in system oscillations. The phase margin is given by

$$\Phi_{PM} = \pi + \arg\{F(jw_1)\}. \tag{6.43}$$

From Eqs. (6.41) and (6.42), $F(jw_1)$ is constrained by

$$|F(jw_1)| = 1. \tag{6.44}$$

We set the parameter T_d to be proportional to the rate sampling interval τ_s, which is given by

$$T_d = \alpha\tau_s,\ 0 \leq \alpha < 1. \tag{6.45}$$

Substituting into Eq. (6.43) from Eqs. (6.42) and (6.45), we obtain

$$w_1 = \frac{\pi/2 - \Phi_{PM}}{(1-\alpha)\tau_s},\ 0 < \Phi_{PM} < \pi/2. \tag{6.46}$$

From Eq. (6.44), we obtain

$$w_1 = \frac{K_R}{\sqrt{1-(K_R\alpha\tau_s)^2}},\ 0 \leq K_R\alpha\tau_s < 1. \tag{6.47}$$

From Eqs. (6.46) and (6.47), we obtain

$$K_R = \frac{\pi/2 - \Phi_{PM}}{\tau_s\sqrt{(1-\alpha)^2 + (\pi/2 - \Phi_{PM})^2\alpha^2}}. \tag{6.48}$$

It is shown in [99] that throughout numerous simulations the parameters of α and Φ_{PM} have optimal values of 0.4 and 64^o, respectively. These optimal parameters of K_P, K_Q, K_R, and T_D are used in our simulation experiments in order to verify the transient response of

the proposed controller.

6.3.5 Rate Computation

The goal of rate controller is to find a rate such that traffic at outgoing link matches to the target traffic $C + q^0$. We utilize the derivative of traffic $U(s)$ as seen in Fig. 6.3 to find the rate, which is given by

$$U(s) = \frac{C_1(s)\,(R(s) - M(s) + E(s))}{1 + (C_1(s) + C_2(s))\,G_M(s)}. \tag{6.49}$$

By inverse-transforming $U(s)$ to time domain, we obtain

$$u(t) = K_p y(t) + K_I \int y(t)\,dt - K_Q f_M(t), \tag{6.50}$$

where

$$y(t) = C + q^0 - (m(t) - e(t)) - f_M(t),$$

$f_M(t)$ is the sending rate at time t obtained by $G_M(s)$, and $m(t)$ and $e(t)$ are the measured traffic at the outgoing link and the traffic estimated by the Smith Predictor, respectively.

Based on $u(t)$, the new rate is given by

$$f(t) = \int u(t)\,dt. \tag{6.51}$$

We use this equation to compute the rate that can control network congestion. This will be verified by simulations experiments.

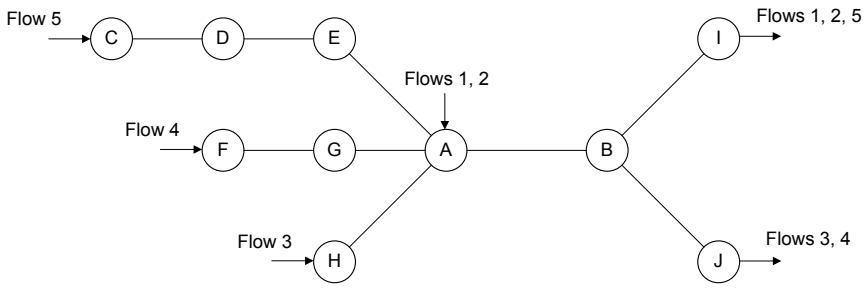


Figure 6.4: The network topology.

6.4 Simulation Results

We present results of simulations for the proposed rate controller. The results are compared with other existing controllers proposed in [32, 88] which are the Benmohamed-Meerkov (BM) controller and the Mascolo-Cavendish (MC) controller, respectively. The simulations were carried out using the NS-2 network simulation platform [106], which is a discrete event simulator. The network topology is shown in Figure 6.4, which is the same as that used in [32, 88] in order to compare the performance of our proposed controller with that of the other existing controllers. The topology consists of 10 different nodes and 9 different links. Every link has identical properties, such that each link has the capacity of 45 Mbps and the propagation delay of 2 ms, and every link is assumed to be bi-directional.

Traffic is generated so as to cause congestion on the network, especially on the link between nodes A and B. Nodes A, C, F, and H generate traffic using ER-UDT (see Chapter 5), which can adjust its sending rate according to the feedback information sent by the routers. Flows 1, 2, and 5 are destined for the right upper node I in Figure 6.4, while flows 3 and 4 are destined for the lower node H. Thus, the five different flows compete with each other for the bandwidth of the bottleneck link (A, B). All flows are generated according to the predefined schedule shown in Table 6.1.

The targeted buffer size Q is set to 30 packets (i.e., 45 Kbytes). The sampling interval is set to 30 ms at node A, which is identical to the inband signaling interval of ER-UDT. We

Flow	1	2	3	4	5
Start time	0	200	50	350	500
End time	700	1000	850	1000	1000

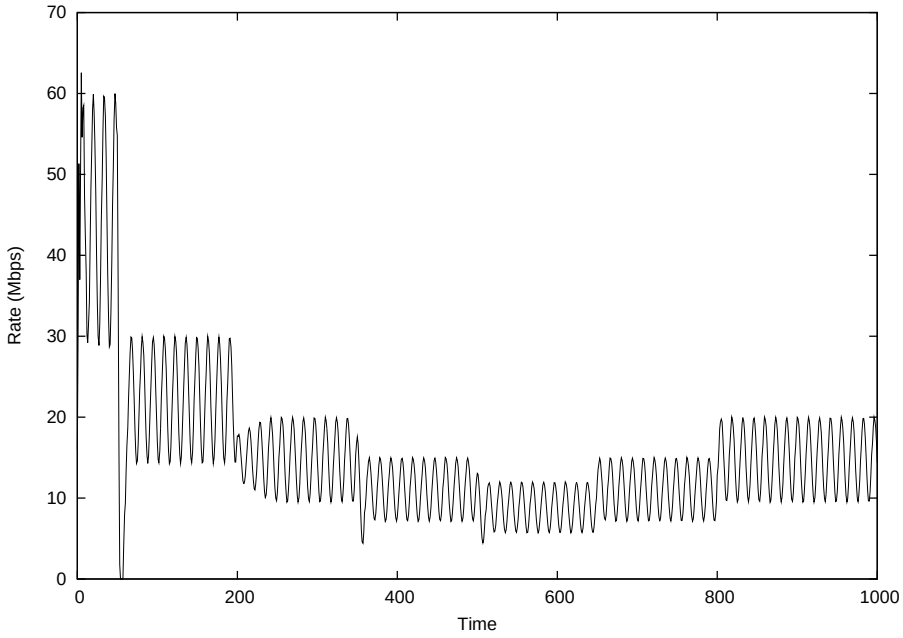
Table 6.1: Traffic generation.

implemented simulations over two different scenario: 1) short RTT and 2) large RTT. For the short RTT scenario, the propagation delay of link (A, B) is set to 2 ms, i.e., the same as the other links. The largest RTT between source and destination in the network is less than the sampling interval. In the large RTT scenario, the propagation delay of link (A, B) is set to 30 ms which makes the largest RTT greater than the sampling interval while the propagation delay on the other links remains at 2 ms. Based on these two scenarios, we investigate the performance of the proposed rate controller.

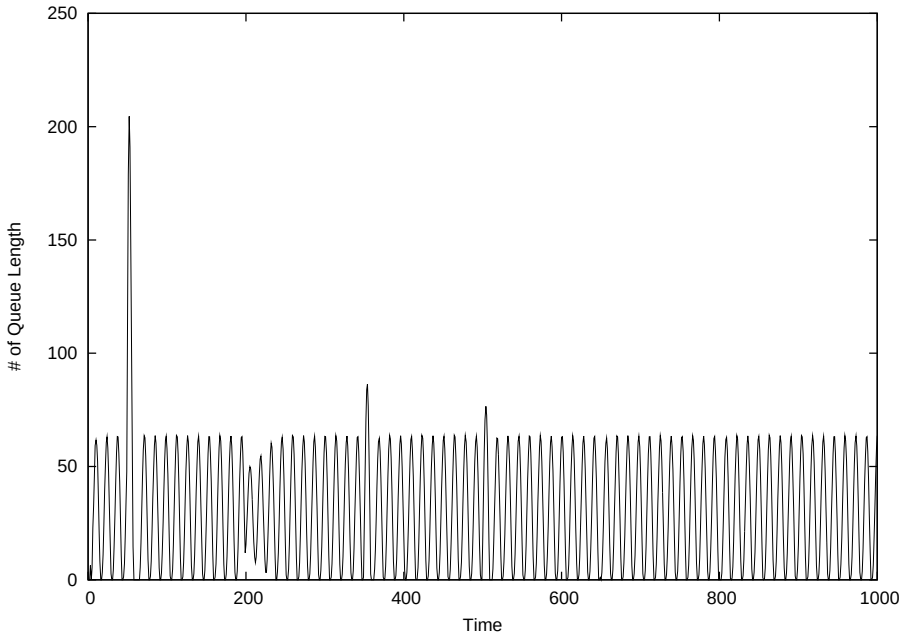
Figure 6.5 illustrates the BM controller in terms of the rate and queue length in the short RTT scenario. We observe that the rate fluctuates over time and does not converge to the desired rate. We also observe that the queue length oscillates and does not converge to the target length of 30 packets. Moreover, the queue occupancy increases dramatically whenever new flows are introduced. Both the rate and queue length fluctuate over time in steady-state because this controller is based on a PD controller. In our proposed controller, a PI controller is introduced to deal with such fluctuations.

Simulation results for the large RTT scenario for the BM controller are shown in Figure 6.6. We observe that the rate slowly changes over time, while the queue length changes drastically. We see that the BM controller does not perform reliably in the large RTT scenario. The BM controller assumes that every packet can carry the signal (i.e., rate feedback) information in its header, such that the rate feedback is updated within a single RTT on the sender side. However, ER-UDT delivers the rate feedback information every 128 packet so the sending rate is updated much more slowly than under the assumption of the BM controller. Thus, the BM controller performs poorly under large RTT when ER-UDT is used.

The performance of the MC controller is shown in Figure 6.7 and 6.8 for the two different

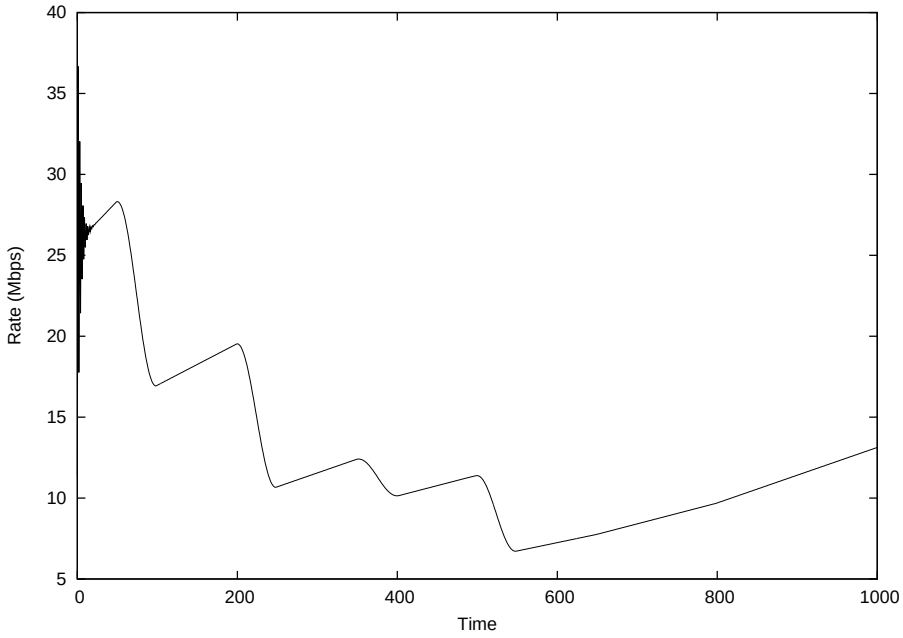


(a) Rate.

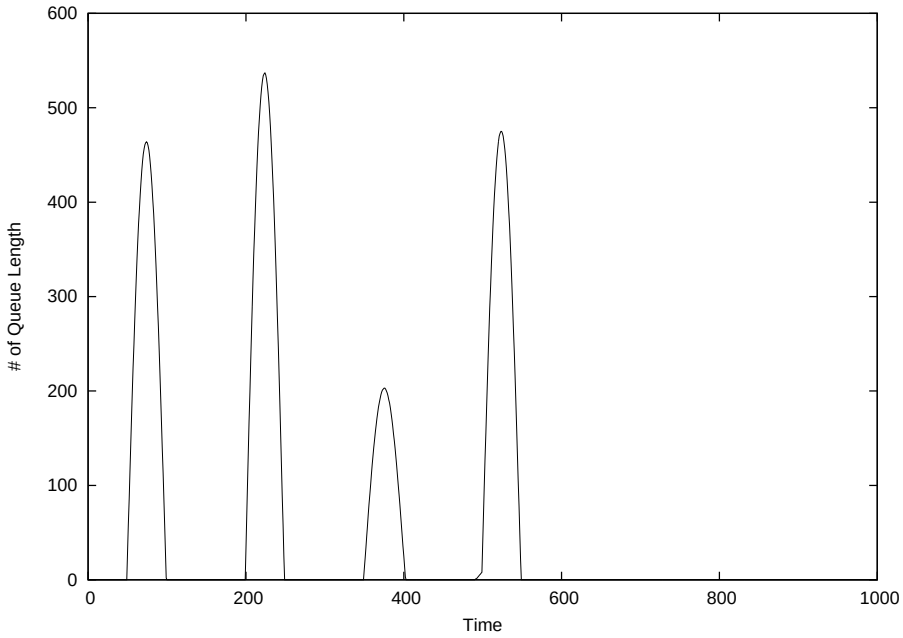


(b) Queue size.

Figure 6.5: Performance of the BM controller in the short RTT scenario.



(a) Rate.



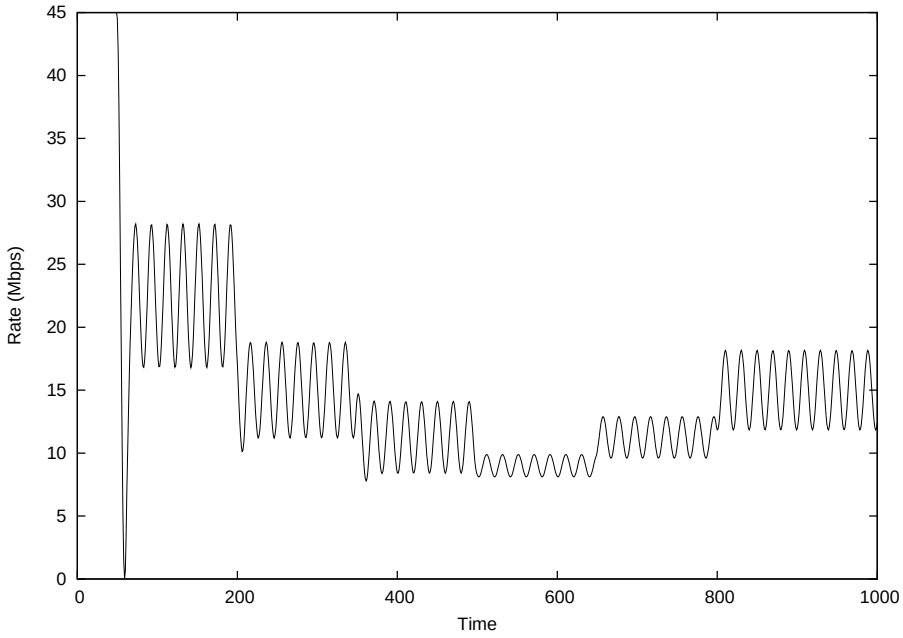
(b) Queue size.

Figure 6.6: Performance of the BM controller in the large RTT scenario.

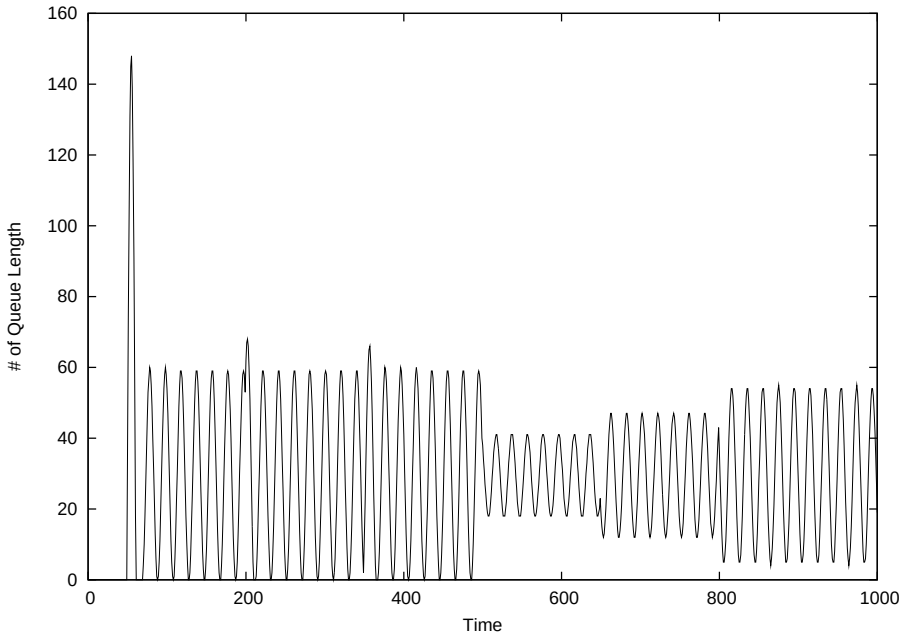
RTT scenarios. We see that the MC controller performs better than the BM controller in both scenarios. In the short RTT scenario, the queue occupancy is high during the first transient period, but settles to a level near the target during the middle of simulation run. In the large RTT scenario, the controller shows large queue occupancy during each transient period because it responds slowly to queue length changes compared with the short RTT scenario. However, the MC controller does not result in convergence of the queue length into the target queue length since it is based on P-PD controllers, that is, it does not employ a PI controller.

The performance of the proposed controller in the short RTT scenario is depicted in Figure 6.9. We see that the proposed controller has the best performance among three controllers. It responds to the changes in the queue length quickly and accurately, both in the transient period and in steady-state. During the transient periods, the proposed controller responds to changes in the queue length by quickly adjusting the sending rate. For the first transient period, the controller does not have enough information on the queue and traffic dynamics, which results in a slightly higher queue length (i.e., 38 packets) compared with the next transient periods. However, the proposed controller responds much better to changes of the queue length during the first transient period compared to the other controllers. In the subsequent transient periods, the queue length is less than 33 packets. In addition, the queue length converges to the target queue length in steady-state, which the other controllers cannot accomplish.

In the large RTT scenario, the queue length changes slowly in the first transient period compared with the short RTT scenario. The main difference between two scenarios occurs before the second flow is generated. In the short RTT scenario, the queue length is maintained at almost 0 until the second flow is generated, but in the long RTT scenario, the queue fills up right after the first flow is generated. The computed sending rates in the two scenarios are slightly different due the difference in the RTTs. However, after the second transient period, the queue length and rate computations are almost the same. Note that the queue length converges to the target length in steady-state, in both scenarios.

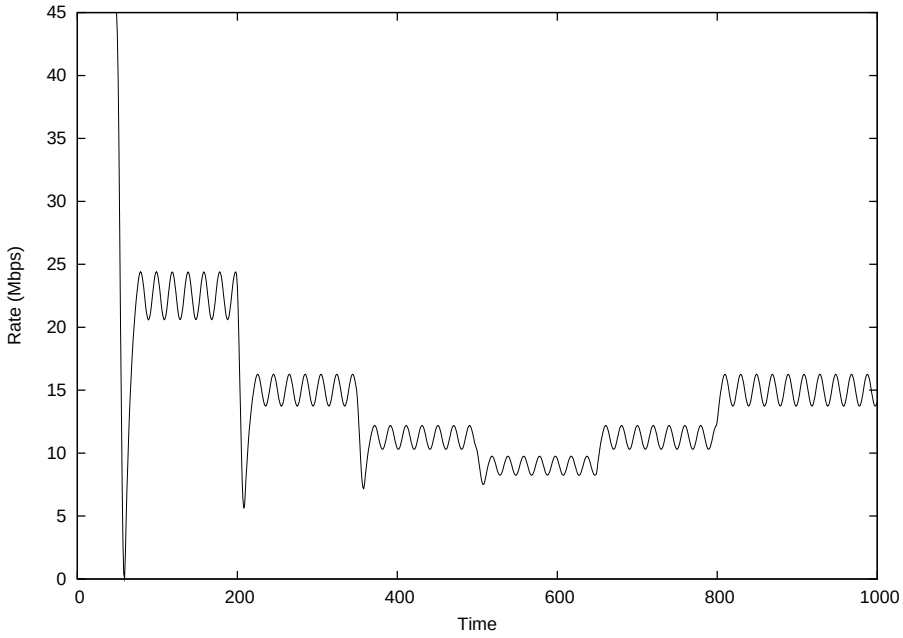


(a) Rate.

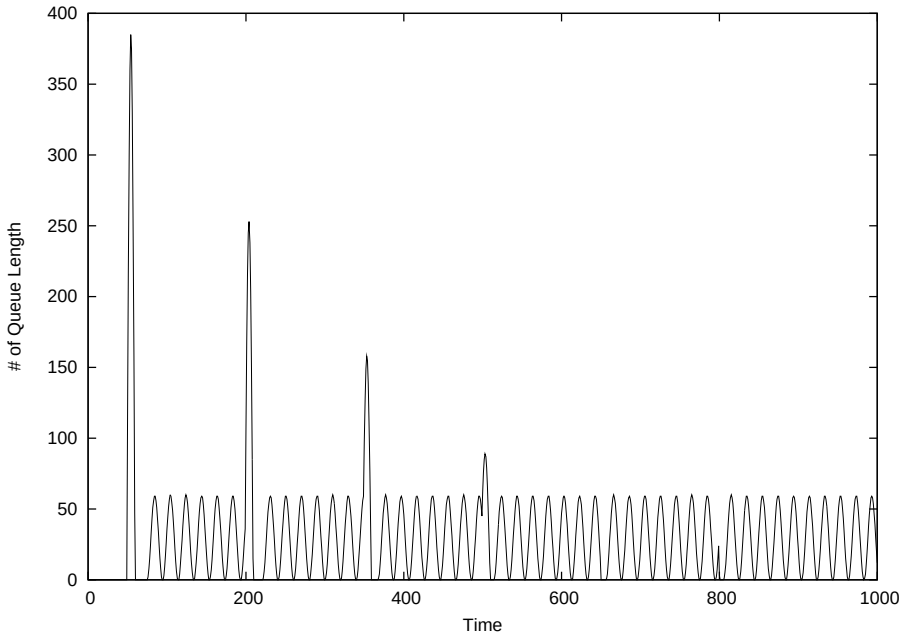


(b) Queue size.

Figure 6.7: Performance of the MC controller in the short RTT scenario.

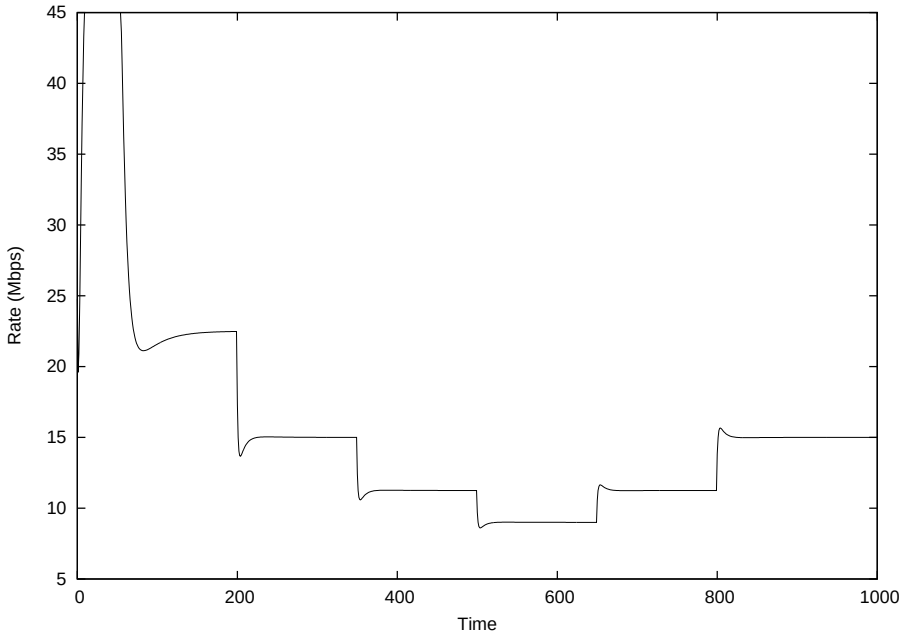


(a) Rate

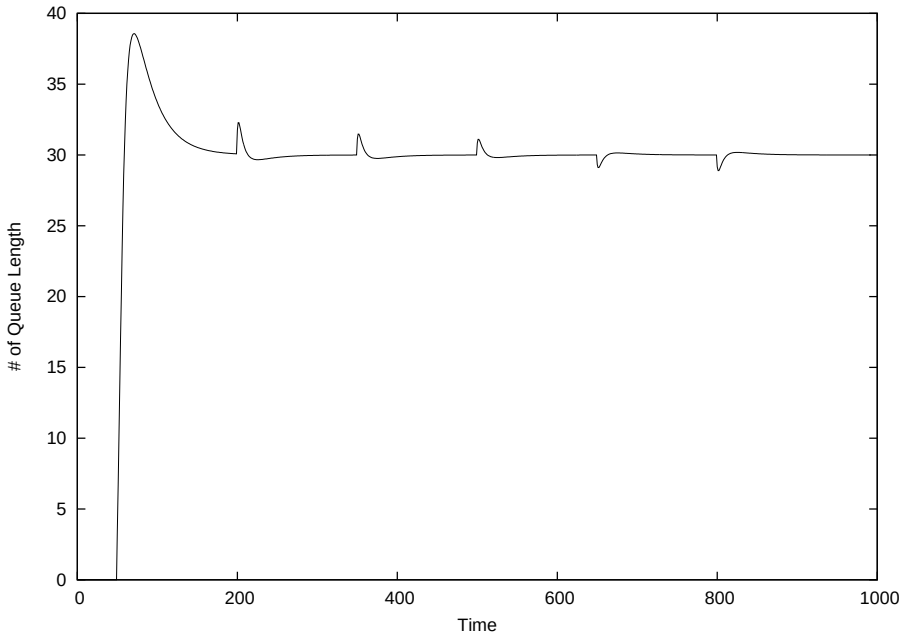


(b) Queue size

Figure 6.8: Performance of the MC controller in the large RTT scenario.

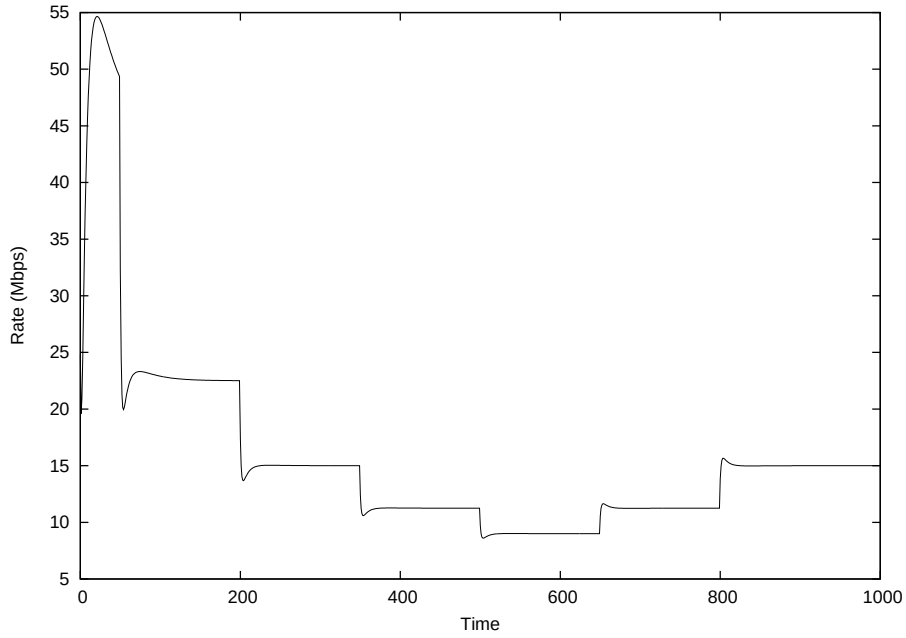


(a) Rate.

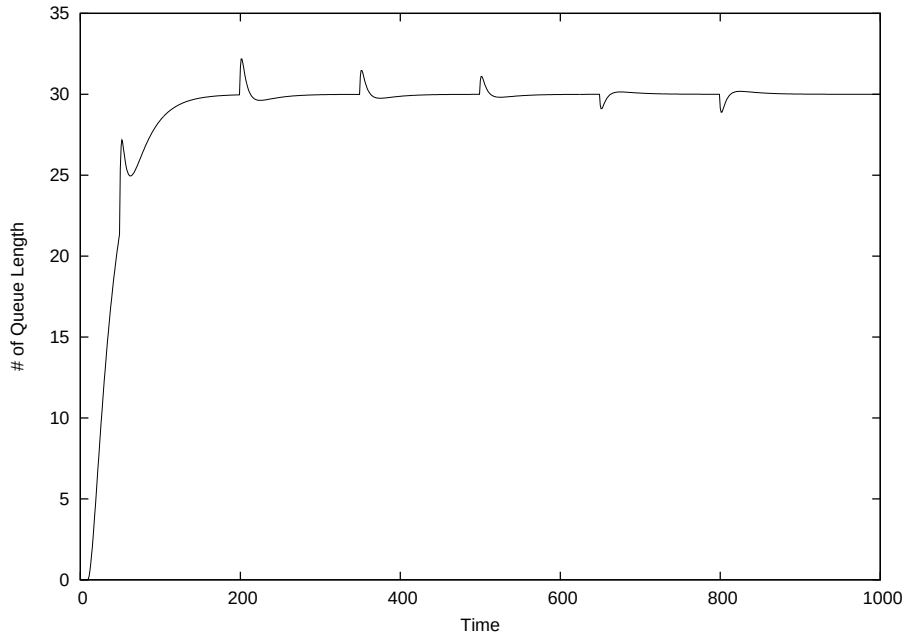


(b) Queue size.

Figure 6.9: Performance of the proposed controller in the short RTT scenario.



(a) Rate.



(b) Queue size.

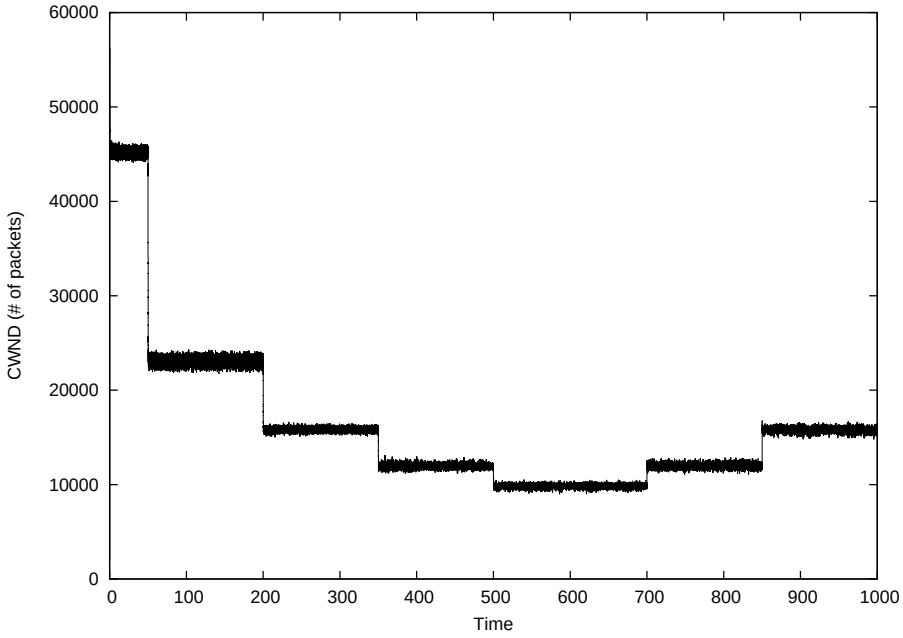
Figure 6.10: Performance of the proposed controller in the large RTT scenario.

Three different controllers show better performance in the short RTT scenario than in large RTT scenario. They respond quicker and maintain the queue length to be close to the target queue length. BC and MC controllers are based on the queue length only, that is, their controllers respond to the queue occupancy only. Thus, they do not respond quickly to the low link utilization, i.e., less than 100% of the link utilization. The proposed controller is designed to respond to both the queue length and the link utilization so that it can produce better performance than other controllers. Further, the proposed controller eliminates fluctuations in steady-state by employing a PI (Proportional-Intergral) controller, and provides the good transient response.

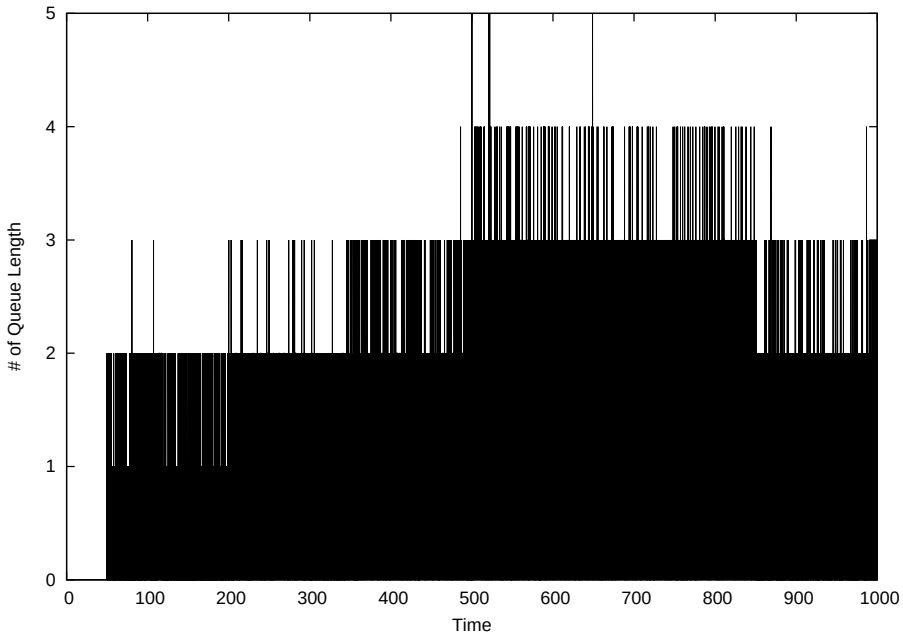
For the purpose of further comparison with existing rate controllers, we ran network simulations using a ns-2 implementation of XCP [25, 26]. XCP is designed to improve the unfairness among multiple flows in TCP. A router implementing XCP computes congestion window size for multiple flows in order to fully utilize the outgoing link capacity and maintain its queue length to be zero. Then, the congestion window size is fed back to each source. This mechanism is somewhat different from our proposed controller, which computes a common sending rate for multiple flows and matches the queue length to a target.

However, we can compare the performance of XCP with that of our proposed controller in terms of the congestion window dynamics and queue dynamics. The congestion window dynamics are related to the rate dynamics. Rather than explicitly comparing congestion window dynamics of XCP to rate dynamics of our proposed controller, we investigate how the congestion window responds to new incoming flows. Figure 6.11 illustrates the congestion window size and queue length of XCP in a short RTT scenario and Figure 6.12 depicts these parameters in the large RTT scenario.

During the transient period in both short and large RTT scenarios, the congestion window shows stable window size transition since XCP computes the window size for each XCP packet arriving to the router. However, the window size does not converge to the desired size since it uses a PD controller mechanism. With respect to the queue length in the short RTT scenario, the XCP router maintains 2 or 3 packets in its queue during

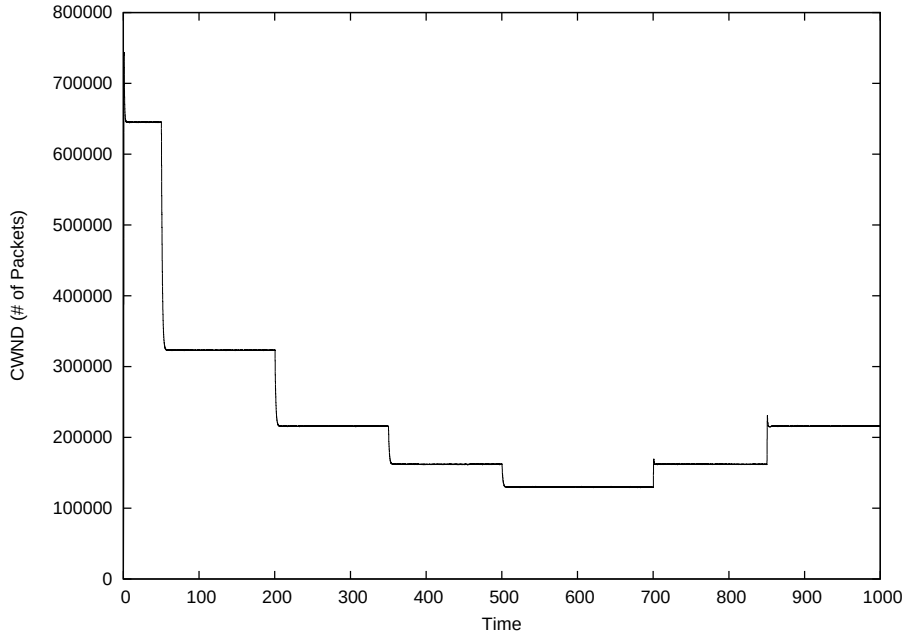


(a) Congestion window.

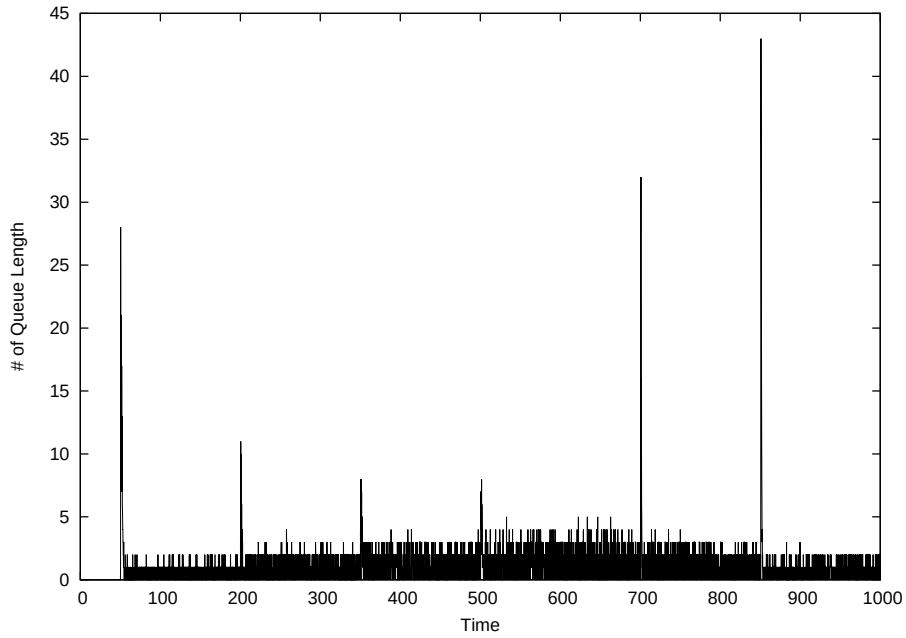


(b) Queue size.

Figure 6.11: Performance of XCP in the short RTT scenario.



(a) Congestion window.



(b) Queue size.

Figure 6.12: Performance of XCP in the large RTT scenario.

most of the simulation run, and has 5 packets at maximum when all flows are generated. However, in the large RTT scenario, the queue occupancy is very high during the transient periods. We observe that as RTT becomes large, the queue length at the XCP router can become very large, leading to packet drops which can cause packet retransmissions.

6.5 Conclusion

We have proposed a new rate controller based on the Smith Predictor that can deal with the feedback delay, which is an critical factor in computer networks. The proposed controller maintains both a stable queue length and link utilization in order to avoid network congestion. We have analyzed the proposed controller on using tools from control theory and demonstrated that the control parameters can guarantee system stability in steady-state and even in transient periods. We have compared the proposed controller with several state-of-the-art congestion control solutions in [32, 88]. Our simulations results show that the proposed controller performs much better than those solutions in terms of rate computation and queue management. The solutions proposed in [32, 88] are designed based on the queue length, whereas the proposed controller is developed based on both queue length and link utilization. This enables the router to compute a sending rate which does not cause network congestion in steady-state and even during transient periods.

Chapter 7: Discussion and Future Research Directions

7.1 Discussion

We proposed an algorithm to find a multipath routes in Chapter 3. The algorithm for determining multipath routes finds a set of disjoint or partially disjoint paths between a source-destination pair based on shortest path routing information. This algorithm does not require the pre-establishment of paths or source routing and can be implemented straightforwardly on top of path vector routing.

In Chapter 4, a congestion-triggered scheme is proposed to distribute traffic over a multipath route to avoid network congestion. In the proposed congestion-triggered multipath (CTMP) routing scheme, a link is identified as being congested if the average link utilization exceeds a given threshold. When a node detects an outgoing link as being congested, it applies an algorithm to determine a set of alternative paths, thus forming a multipath route. Traffic is then moved away from the congested link onto the alternative paths according to a traffic splitting function. Our ns-2 simulation results demonstrate the ability of the multipath routing scheme to relieve network congestion and improve overall network utilization.

In Chapter 5, we proposed an explicit rate congestion control protocol that takes into account the feedback of rate and propagation delay and improve the throughput over impaired links. The proposed protocol confirms the ER and round trip time (RTT) between source and destination pairs. This confirmation helps routers to effectively utilize the available bandwidth in their rate computation and to compute the optimal sending rate. Also, in terms of performance on impaired link, the simulation results show that the proposed protocol performs much better than existing protocols.

In Chapter 6, a new rate control algorithm (ER-UDT) is proposed to compute the

local explicit rate at a given router. The rate controller is designed to provide the sending rate adjustment for the multiple flows (i.e., aggregate flows) on outgoing link. This rate controller is designed based on the Smith Predictor, which can accurately compute the sending rate under conditions of different propagation delays. It also quickly responds to high/low link utilization conditions by using information about both the link utilization and the queue dynamics. This results in full utilization of the outgoing link capacity, preventing from packet loss and wasting available bandwidth. We discuss the analysis of the proposed controller on basis of control theory and demonstrate that the control parameters can guarantee the system stability in steady-state and even in transient period

7.2 Future research directions

The CTMP scheme proposed in Chapter 4 could be combined with the explicit rate control scheme developed in Chapters 5 and 6. The CTMP scheme identifies flow and routes it as a unit along a fixed path to avoid out-of-order packet delivery. It also exploits path redundancy by distributing excess traffic onto multiple alternative paths. Moreover, the explicit rate (ER) control limits source rate of a given flow on a given path based on explicit rate feedback and improves throughput of a flow along a given path. Thus, the combination of CTMP and ER control could achieve the benefits of both schemes.

7.2.1 Explicit rate based multipath routing

We have proposed an approach to interoperate explicit rate signaling protocol with multipath routing in Chapter 5. In this approach, the router maintains ER information associated with links in its routing table and periodically exchanges it with others. Using ER as an additional link metric stored in the routing table, the router can determine an optimal path among the alternative paths in the multipath route obtained by the class-c algorithm discussed in Chapter 3. By incorporating ER information into the routing decision, the router performs load-balancing by assigning new incoming flows to the current best path based on bandwidth availability.

However, the issue of incorporating ER information into routing requires further investigation. The responsiveness of the routing algorithm to changes in ER values is a critical parameter. If the routing table updates are too infrequent, the load-balancing would be based on old information, which would severely limit its effectiveness. On the on the other hand, frequent routing table updates would increase the signaling load on the network.

7.2.2 Explicit Overlay

Explicit rate signaling has to be supported by every router in the network in order for routers to feed back accurate information to the sender. However, explicit rate signaling may not be supported by every router in the network. In practice, only a small percentage of routers in the network may support explicit rate signaling. To facilitate the incremental deployment of explicit rate signaling, future work could investigate the design of an *overlay network* [107–110] that supports explicit rate signaling.

An overlay network is a virtual network built on top of the existing networks. The nodes (e.g., router, switch) in the overlay network, i.e., overlay nodes, work as routers to deliver data to their destinations. Between two overlay nodes, a virtual link is established, so-called an *overlay link*. The overlay link may include one or more conventional routers as intermediate nodes between two overlay nodes. Explicit rate control could be implemented on the overlay nodes. Since intermediate nodes in an overlay link may not support explicit rate signaling, a new rate computation algorithm must be designed to realize an explicit rate overlay network. Since queue length information in the overlay link is not generally available, one possible approach is to utilize the round trip time of overlay link as a measure of congestion in the overlay link.

Figure 7.1 illustrates an overlay network and two overlay links: the link defined by path (A, B, C, D) and the link defined by path (A, E, D). Nodes B, C, and E are conventional routers that do not have explicit rate signaling capability. Overlay node A estimates the base RTT and the observed RTT for the two overlay links to overlay node D, where the observed RTT is the present measure of RTT on the overlay link, and the base RTT is the

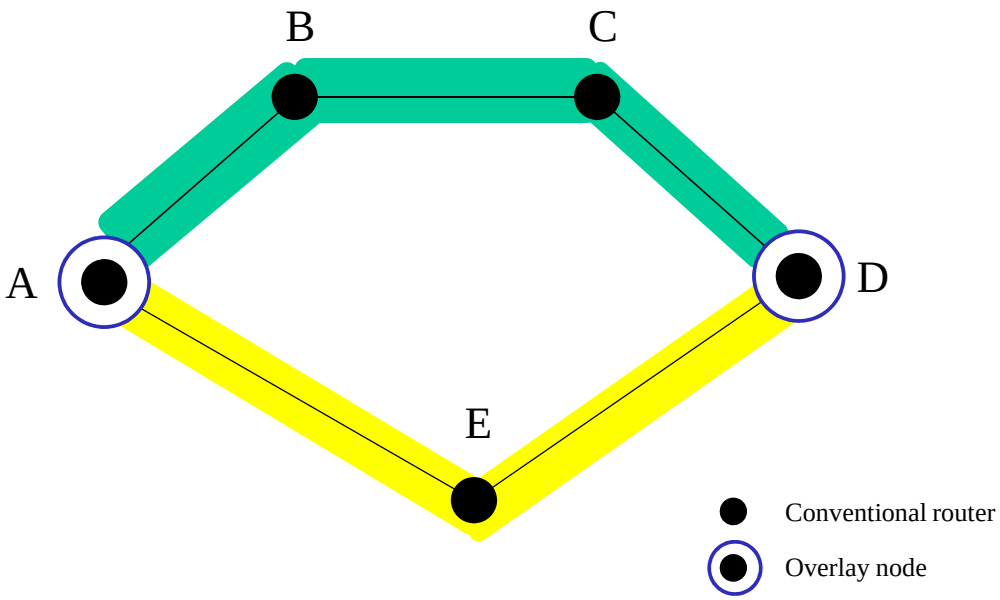


Figure 7.1: Overview of overlay network.

minimum observed RTT on the overlay link, similar to the base RTT and observed RTT in Fast TCP [111–116] and TCP Vegas [117–121]. In order to estimate these two parameters, node A encapsulates the incoming packet with an overlay packet header and then transmits the encapsulated packet to node D on the appropriate overly link. The overlay packet header includes a SN (Sequence Number) and Timestamp. Whenever node D receives an overlay packet, it creates a response packet and transmits it back to the node A. Thus, the node A will be able to estimate the round trip time, which can be utilized to compute the base RTT and the observed RTT.

The two RTTs and overlay packet loss are utilized to measure the degree of congestion in the overlay link. As the difference of two RTTs increases, the degree of congestion increases, vice versa. Packet loss statistics could also be used to indicate severe congestion. A new rate controller could be designed based on the base RTT, the observed RTT, and packet loss on the overlay link. Alternatively, one could attempt to derive an approximate formula

relating these three parameters to the queue length q on the overlay link:

$$q = F(\mathrm{RTT}_{\mathrm{base}}, \mathrm{RTT}_{\mathrm{obs}}, p_{\mathrm{loss}}), \tag{7.1}$$

where $\mathrm{RTT}_{\mathrm{base}}$ is the base RTT, $\mathrm{RTT}_{\mathrm{obs}}$ is the observed RTT, and p_{loss} is the packet loss statistics. In this case, the rate controller proposed in Chapter 6 could be employed without modification. The effectiveness of this approach would depend heavily on the approximate formula for queue length.

Bibliography

Bibliography

[1] L. G. Roberts, "QoS Signaling for IP QoS Support," *Telecommunications Industry Association*, May 2005.

[2] S. Floyd, "The NewReno Modification to TCP's Fast Recovery Algorithm," *IETF RFC 2592*, April 1999.

[3] G. Malkin, "RIP Version 2 Protocol Analysis," *IETF Internet RFC 1721*, November 1994.

[4] J. Moy, "OSPF Version 2," *IETF Internet RFC 2328*, 1998.

[5] P. Traina, "BGP-4 Protocol Analysis," *IETF RFC Internet 1774*, October 1995.

[6] H. Suzuki and F. A. Tobagi, "Fast bandwidth reservation scheme with multi-link and multi-path routing in ATM networks," in *Proc. IEEE INFOCOM*, 1992.

[7] R. Krishnan and J. Silvester, "Choice of allocation granularity in multi-path source routing schemes," in *Proc. IEEE INFOCOM*, vol. 1, 1993, pp. 322–329.

[8] S. Bahk and M. E. Zarki, "Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks," in *Proc. SIGCOMM*, 1992, pp. 53–64.

[9] P. Georgatsos and D. Griffin, "A management system for load balancing through adaptive routing in multiservice ATM networks," in *Proc. IEEE INFOCOM*, 1996, pp. 863–870.

[10] K. Ishida, Y. Kakuda, and T. Kikuno, "A routing protocol for finding two node-disjoint paths in computer networks," in *International Conference on Network Protocols*, 1992, pp. 340–347.

[11] J. de Azevedo, M. Costa, J. Madeira, and E. Marins, "An algorithm for the ranking of shortest paths," *European Journal of Operational Research*, pp. 97–106, 1993.

[12] A. Bako and P. Kas, "Determining the k-th shortest path by matrix method," *Szigma*, pp. 61–66, 1977.

[13] S. E. Dreyfus, "An appraisal of some shortest path algorithms," *Operations Research*, pp. 395–412, 1969.

[14] J. W. Suurballe, "Disjoint paths in a network," *Networks*, pp. 125–145, 1974.

[15] B. L. Fox, "k-th shortest paths and applications to the probabilistic networks," *ORSA/TIMS Joint National Mtg.*, p. B263, 1975.

[16] G. J. Horne, "Finding the K least cost paths in an acyclic activity network," *J. Operational Research Soc.*, pp. 443–448, 1980.

[17] N. Katoh, T. Ibaraki, and H. Mine, "An $O(Kn^2)$ algorithm for K shortest simple paths in an undirected graph with nonnegative arc length," *Trans. Inst. Electronics and Communication Engineers of Japan*, pp. 971–972, 1978.

[18] ——, "An efficient algorithm for K shortest simple paths," *Networks*, vol. 12, pp. 411–427, 1982.

[19] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Management Science*.

[20] B. Jabbari, S. Gong, and E. Oki, "On constraints for path computation in multi-layer switched networks," *IEICE Transactions on Communications*, vol. E90-B, no. 8, pp. 1922–1927, 2007.

[21] C. Villamizar, "OSPF Optimized Multipath," *IETF Internet draft*, February 1999.

[22] J. Postel, "Transmission Control Protocol," *IETF RFC 793*, September 1981.

[23] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Nework*, vol. 11, no. 6, pp. 10 – 23, January 1997.

[24] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *IETF RFC 2001*, January 1997.

[25] D. Katabi, M. Handley, and C. Rohrs, "Internet Congestion Control for High Bandwidth-Delay Product Networks," in *Proc. of ACM Sigcomm 2002*, August 2002.

[26] Y.Zhang and T. Henderson, "An implementation and experimental study of the eXplicit Control Protocol (XCP)," in *Proc. of IEEE INFOCOM*, 2005.

[27] N. Dukkipati, M.Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor Sharing Flows in the Internet," in *In Thirteenth International Workshop on Quality of Service (IWQoS)*, June 2005.

[28] L. G. Roberts, "Major Improvements in TCP Performance Over Satellite and Radio," in *Proc. MILCOM*, October 2006, pp. 1–7.

[29] J. Postel, "User Datagram Protocol," *IETF RFC 768*, August 1980.

[30] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, no. 4, pp. 314 – 329, August 1988.

[31] W. Feng, D. Kandlur, D. Saha, and K. Shin, "The Blue Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, August 2002.

[32] L. Benmohamed and S. M. Meerkov, "Feedback Control of Congestion in Packet Switching Networks: The Case of a Single Congested Node," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, pp. 693 – 707, December 1993.

[33] ——, "Feedback Control of Congestion in Packet Switching Networks: The Case of a Multiple Congested Node," *International Journal of Communications Systems*, vol. 10, no. 5, pp. 227 – 246, September 1997.

[34] A. Kolarov and G. Ramamurthy, "A Control-Theoretic Approach to the Design of an Explicit Rate Controller for ABR Service," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 741 – 753, October 1999.

[35] B. Kim, "Theoretic Framework for Feedback Control Models in High-Speed Networks," in *14th International Workshop on Database and Expert Systems Applications*, 2003, pp. 134–138.

[36] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)," *IETF Internet RFC 2205*, September 1997.

[37] IP/MPLS Forum. [Online]. Available: http://www.ipmplsforum.org/

[38] D. Johnson, Y. Hu, and D. Maltz, " The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," *IETF Internet RFC 4728*, February 2007.

[39] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," in *Proc. the conference on Communications architecture and protocols*, vol. 21, August 1991, pp. 885–896.

[40] R. Ogier, V. Rutenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Transactions on Information Theory*, vol. 39, pp. 443–455, March 1993.

[41] D. Eppstein, "Finding the k shortest paths," in *Proc. Foundations of Computer Science*, November 1994, pp. 154–165.

[42] R. Bhandari, "Optimal hysical diversity algorithms and survivable networks," in *Proc. Second IEEE Symposium on Computers and Communications*, Alexandria, Egypt, July 1997, pp. 433–441.

[43] D. Estrin, T. Li, Y. Rekhter, K. Varadhan, and D. Zappala, "Source Demand Routing: Packet Format and Forwarding Specification," *IETF Internet RFC 1940*, May 1996.

[44] J. Dunn and C. Martin, "Terminology for ATM Benchmarking," *IETF RFC 2761*, February 2000.

[45] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Lable Switching Architecture," *IETF RFC 3031*, January 2001.

[46] S. Murthy and J. Garcia-Luna-Aceves, "Congestion-Oriented Shortest Multipath Routing," in *Proc. IEEE INFOCOM*, March 1996.

[47] W. Zaumen and J. J. Garcia-Luna-Aceves, "Loop-Free Multipath Routing Using Generalized Diffusing Computations," in *Proc. IEEE INFOCOM*, March 1998.

[48] S. Vutukury and J. Garcia-Luna-Aceves, "An Algorithm for Multipath Computation using Distance-Vectors with Predecessor Information," in *Proc. IEEE IC3N*, Boston, Massachusetts, October 1999.

[49] P. Narvaez and K. Y. Siu, "Efficient Algorithms for Multi-Path Link State Routing," in *Proc. ISCOM*, Kaohsiung, Taiwan, 1999.

[50] S. Vutukury and J. Garcia-Luna-Aceves, "MDVA: A Distance-Vector Multipath Routing Protocol," in *Proc. IEEE INFOCOM*, Anchorage, Alaska, USA, April 2001, pp. 319–327.

[51] ——, "MPATH: a loop-free multipath routing algorithm," *Journal of Microprocessors and Microsystems*, pp. 319–327, 2001.

[52] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," *IETF RFC 2581*, April 1999.

[53] T. Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–91, 2003.

[54] S. Floyd, "HighSpeed TCP for Large Congestion Windows," *IETF Internet RFC 3649*, December 2003.

[55] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. (1996) TCP Selective Acknowledgment Options. [Online]. Available: citeseer.ist.psu.edu/mathis96tcp.html

[56] ——, "TCP Selective Acknowledgment Options," *IETF RFC 2018*, October 1996.

[57] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP," *IETF RFC 2883*, July 2000.

[58] S. Floyd and V.Jacobson, "Random Early Detection Gateways for congestion avoidance," *IEEE/ACM Transaction on Networking*, pp. 397 – 413, August 1993.

[59] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active Queue Management," *IEEE Network*, 2001.

[60] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. of IEEE INFOCOM*, April 2001.

[61] R. Gibbens and F. Kelly, "Distributed connection acceptance control for a connectionless network," in *Proc. of the 16th International Telegraffic Congress*, June 1999.

[62] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue," in *Proc. of ACM SIGCOMM*, 2001.

[63] K. Ramakrishnan, S. Floyd, and D. Black, "The addtion of explicit congestion notification (ECN) to IP," *IETF RFC 3168, Internet Proposal Standard*, 2001.

[64] S. H. Low, F. Paganini, J. Wang, S. Aldlakha, and J. C. Doyle, "Dynamics of TCP/AQM and a scalable control," in *Proc. IEEE INFOCOM*, June 2002.

[65] D. Katabi and C. Blake, "A note on the stability requirements of Adaptive Virtual Queue," 2002, MIT Technical Memo.

[66] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *IETF Internet RFC 2475*, December 1998.

[67] T. Lizambri, F. Duran, and S. Wakid, "Priority scheduling and buffer management for ATM traffic shaping," in *Proc. 7th IEEE Workshop on Future Trends of Distributed Computing Systems*, December 1999.

[68] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based Congestion Control for Unicast Applications," in *ACM SIGCOMM*, 2000.

[69] R. Wang, S. Horan, B. Gutha, B. Sun, and P. Reguri, "An Experimental Analysis of Rate-based and Window-based Transmission Mechanisms over Simulated Space-Internet Links," in *IEEE GLOBECOM*, 2005, pp. 281 – 285.

[70] S. Sohn, B. L. Mark, and J. Brassil, "Congestion-triggered Multipath Routing based on Shortest Path Information," in *Proc. ICCCN*, October 2006, pp. 191–196.

[71] R. K. Ahuja, T. L. Magnatti, and J. B. Orlin, *Network flows.* Prentice Hall, Englwood Cliffs, 1993.

[72] S. Sohn, B. L. Mark, and J. T. Brassil, "Congestion-triggered Multipath Routing based on Shortest Path Information," Network Architecture and Performance Laboratory, Dept. of Electrical and Computer Enginering, George Mason University, Fairfax, VA, Tech. Rep., July 2006.

[73] Z. Cao, Z. Wang, and E. W. Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," in *Proc. INFOCOM*, 2000, pp. 332–341.

[74] Y. Gu and R. L. Grossman, "UDT: UDP-based Data Transfer for High-Speed Wide Area Networks," *Computer Networks (Elsevier)*, vol. 51, May 2007.

[75] Anagran. [Online]. Available: http://www.anagran.com/

[76] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," *IETF Internet RFC 3448*, 2003.

[77] S. Jin, L. Guo, I. Matta, and A. Bestavros, "TCP-Friendly SIMD congestion control and its convergence behavior," in *Proc. International Conference on Network Protocols*, November 2001, pp. 156–164.

[78] B. Kim and J. Lee, "Retransmission Loss Recovery by Duplicate Acknowledgement Counting," *IEEE Comm. Letters*, vol. 8, no. 1, pp. 69 – 71, January 2004.

[79] B. Kim, D. Kim, and J. Lee, "Lost Retransmission Detection for TCP SACK," *IEEE Comm. Letters*, vol. 8, no. 9, pp. 600 – 602, September 2004.

[80] B. Kim, Y. Kim, M. Oh, and J. Choi, "Microscopic behaviors of TCP loss recovery using lost retransmission detection," in *Consumer Communications and Networking Conference*, 2005, pp. 296 – 301.

[81] Emulab-Network Emulation Testbed. [Online]. Available: http://www.emulab.net

[82] K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 158–181, 1990.

[83] K. W. Fendick, M. A. Rodrigues, and A. Weiss, "Analysis of a rate-based feedback control strategy for long haul data transport," *Performance Evaluation*, vol. 16, pp. 67–84, 1992.

[84] L. Roberts, "Enhanced PRCA (Proportional Rate-Control Algorithm)," 1994. [Online]. Available: http://www.ohub.net

[85] N. Yin and M. G. Hluchyj, "On closed-loop rate control for ATM cell relay networks," in *Proc. Infocomm 94*, vol. 1, 1994, pp. 99–108.

[86] F. Bonomi, D. Mitra, and J. B. Seery, "Adaptive algorithms for feedback-based flow control in high-speed, wide-area ATM networks," vol. 13, no. 7, 1995, pp. 1267–1283.

[87] B. L. Mark, R. Fan, and G. Ramamurthy, "Dynamic Rate Control Scheduling for ATM Switches," in *Proc. IEEE GlobeCom'98*, vol. 1, November 1998, pp. 439–446.

[88] S.Mascolo, D. Cavendish, and M. Gerla, "ATM Rate Based Congestion Control Using a Smith Predictor," *Performance Evaluation, Special Issue on ATM Traffic Control*, vol. 31, no. 1-2, November 1997.

[89] S.Mascolo, "Congestion Control in High-Speed Communication Networks Using the Smith Principle," *Special Issue on Control Methods for Communication Networks*, December 1999.

[90] J. Aweya, M. Ouellette, and D. Y. Montuno, "A simple, scalable, and provably stable explicit rate computation scheme for flow control in communication networks," *Int. J. commun. Syst.*, vol. 14, pp. 593 – 618, June 2001.

[91] S. Ryu and C. Cho, "PI-PD-controller for robust and adaptive queue management for supporting TCP Congstion Control," in *Proc. Annual Simulation Symposium*, 2004.

[92] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operational Research Society*, pp. 237 – 252, March 1998.

[93] S. Mascolo, "Smith's Principle for Congestion Control in High Speed Data Networks," *IEEE Transactions on Automatic Control*, February 2000.

[94] S. Mascolo, D. Cavenish, and M. Gerla, "ATM Rate Based Congestion Control Using a Smith Predictor: an EPRCA Implementation," in *Proc. INFOCOM'96*, vol. 2, March 1996, pp. 569–576.

[95] D. Cavendish, M. Gerla, and S. Mascolo, "A control theoretic approach to congestion control in packet networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 893 – 906.

[96] K. Watanabe and M. Ito, "A process model control for linear systems with delay," *IEEE Trans. Automat. Control*, vol. AC-26, no. 6, pp. 1261 – 1266, December 1981.

[97] K. J. Astrom, C. C. Hang, and B. C. Lim, "A new Smith predictor for controlling a process with an integrator and long dead-time," *IEEE Trans. Automat. Control*, vol. 39, no. 2, pp. 343–345, February 1994.

[98] M. R. Matausek and A. D. Micic, "A modified Smith predictor for controlling a process with an integrator and long dead-time," *IEEE Trans. Automat. Control*, vol. 41, pp. 1199 – 1203, August 1996.

[99] ——, "On the modified Smith predictor for controlling a process with an integrator and long dead-time," *IEEE Trans. Automat. Control*, vol. 44, no. 8, pp. 1603 – 1606, August 1999.

[100] I. Kaya, "A new Smith predictor and controller for control of processes with long dead time," *ISA Transactions*, vol. 42, pp. 101–110, 2003.

[101] B. Mark and S. Zhang, "A Multipath Flow Routing Approach for Increasing Throughput in the Internet," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, August 2007.

[102] B. Mark, S. Zhang, R. McGeer, J. Brassil, P. Sharma, and P. Yalagandula, "Performance of an Adaptive Routing Overlay under Dynamic Link Impairments," in *Proc. IEEE Military Communications Conference*, October 2007.

[103] J. Brassil, R. McGeer, R. Rajagopalan, P. Sharma, P. Yalagandula, S. Banerjee, D. Reed, S. J. Lee, A. Bavier, L. Peterson, L. Roberts, A. Henderson, B. Khorram, S. Zhang, B. M. S. Sohn, C. Heiter, J. Spies, and N. Watts, "The CHART System: A High-Performance, Fair Transport Architecture Based on Explicit-Rate Signaling," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, pp. 26–35, January 2009.

[104] Anagran FR-1000. [Online]. Available: http://anagran.com/products/fr1000

[105] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Prentice Hall, 2005.

[106] The Network Simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns

[107] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. the eighteenth ACM symposium on Operating systems principles*, vol. 35, October 2001, pp. 131– 145.

[108] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," University of California at Berkeley, Berkeley, CA, Tech. Rep. Tech. Rep. CSD-01-1141, 2001.

[109] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41– 53, January 2004.

[110] H. T. Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi, "BANANAS: an evolutionary framework for explicit and multipath routing in the internet," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 277–288, October 2003.

[111] J. Wang, A. Tang, and S. H. Low, "Local stability of FAST TCP, booktitle=."

[112] J. Wang, D. X. Wei, and S. H. Low, "Modeling and stability of FAST TCP," in *Proc. IEEE INFOCOM 2005*, 2005, pp. 938–948.

[113] ——, "Modeling and stability of FAST TCP," *IMA Volumes in Mathematics and its Applications*, vol. 143, pp. 331–356, 2006.

[114] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. IEEE Infocom*, 2004.

[115] C. Jin, D. X. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, H. N. O. Martin, F. Paganini, S. Ravot, and S. Singh, "FAST TCP: from theory to experiments," *IEEE Network*, vol. 19, no. 1, pp. 4–11, January 2005.

[116] L. Tan, C. Yuan, , and M. Zukerman, "FAST TCP: fairness and queuing issues," *IEEE Communication Lettet*, vol. 9, no. 8, pp. 762–764, August 2005.

[117] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM*, 1994, pp. 24–35.

[118] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, p. 1465ï¿½1480, October 1995.

[119] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, October 2000.

[120] S. H. Low, L. Peterson, and L. Wang, "Understanding Vegas: a duality model," *Journal of ACM*, vol. 49, no. 2, pp. 207–235, March 2002.

[121] C. Samios and M. Vernon, "Modeling the Throughput of TCP Vegas," in *Proc. ACM SIGMETRICS'03*, June 2003.

Curriculum Vitae

Soonyong Sohn received the B.S. (Bachelor of Science) in Telecommunication and Information Engineering in 1991 from Hankuk Aviation University, Republic of Korean. He then pursued graduate studies at George Mason University, Fairfax, VA, where he received the M.S. (Master of Science) in 2002. His main research interests lie in the design, modeling and performance evaluation of broadband wired network architectures and protocols. More generally, his interests lie in both theoretical and practical implementation aspects of information sciences and systems.