THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

A Family of QoS-Aware Traffic Control Protocols

By

YE LEI

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

November 2009

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signature)

_____(Name of Student)

ABSTRACT

As the Internet has evolved into a global commercial infrastructure, there has been a growing demand for Internet to support various real-time applications, such as voice over IP, IPTV and multimedia streaming. These applications call for various real-time services including Hard Real-time (HRT), Real-time Delay Adaptive (RDA) and Real-time Rate Adaptive (RRA) services. However, the existing transport layer protocols, including TCP and User Datagram Protocol (UDP), cannot provide such services. In today's Internet, the QoS features can only be enabled through the help of the core network nodes using technologies such as Resource ReSerVation Protocol (RSVP), DiffServ or Multiprotocol Label Switching (MPLS). These approaches, however, require significant involvement of the core network nodes and therefore have difficulty to scale to the global Internet. Hence it is necessary to design better distributed end-to-end congestion control mechanism to enable QoS feature for real-time applications.

In this thesis, we design a family of end-to-end traffic control solutions for both the reliable and the unreliable services.

Firstly, we propose the end-to-end traffic control design methodology, known as TCP-Elastic Real-time SErvice (TERSE). TERSE provides a unified end-to-end traffic control protocol that enables the Non-Real-time Elastic (NRE) service (i.e., the same as the one under the TCP control), Real-time Delay Adaptive (RDA) service, and Real-time Rate Adaptive (RRA) service. A specific service is enabled by properly setting a single parameter in the protocol only. TERSE is underpinned by a sound design methodology. While having its

root in the well-known utility-based optimization approach, this methodology successfully addresses the limitations of the utility-based optimization approach. It leads to the successful design of the unified traffic control protocol, which enjoys some provable properties, including fairness, convergence, and stability. Other than the traditional utility-based approach which expresses the QoS features implicitly in the utility function, TERSE methodology defines the utility functions of all the QoS services the same as TCP utility function and expresses the QoS features explicitly as boundary conditions of the optimization problem.

TERSE methodology requires the utility function of the prevailing TCP of Internet to design the end-to-end QoS aware protocols. So we derive a global utility function and the corresponding optimal control law, known as TCP control law, which maximizes the global utility. The TCP control law captures the essential behaviors of TCP, including slow start, congestion avoidance, and the binary nature of congestion feedback in TCP. We find that the utility function of TCP is linear in the slow start phase and is proportional to the additive increase rate and approaches the well-known logarithm function as the data rate becomes large in the congestion avoidance phase. We also show the fact that understanding the slow start phase with a fixed threshold is critical to the design of new transport layer control protocols to enable quality of service features.

For reliable services, based on TERSE and TCP utility function, we derive a family of optimal, distributed, QoS-aware, end-to-end congestion control laws. It enables a set of class of services (CoSs) including Assured Forwarding Service (AF), Minimum Rate Guaranteed Service (MRG), Upper Bounded Rate Service (UBR), and Minimum Rate Guaranteed and Upper Bounded Rate Service (MRGUBR). Also we use the fluid model to study the MRG control law under what conditions that it can achieve the target rate in competing with the TCP flows. These control laws are implemented as the unified window-based congestion

control protocols, similar to the window-based TCP congestion control protocol. The performance of these protocols is tested based on NS-2 simulation. The results indicate that the protocols are indeed TCP friendly and can provide end-to-end service assurance as long as the percentage of network bandwidth consumed by the flows using these protocols is moderately small. Both analytical and simulation studies show that they can provide required soft minimum-rate guarantee for both RRA and RDA traffic flows. The MRG protocol is also implemented in LINUX-based systems. The cross-pacific testing of this protocol with soft minimum rate guarantee shows that it can achieve more than 1.2Mbps throughput performance, 150% higher than TCP-reno and 50% higher than TCP cubic, which is considered to be the fastest variation of TCP.

For unreliable services, we first redefine the congestion indicator by the help of the Single Trip Time (STT). Then, we design an end-to-end Congestion Control Identifier (CCID) for Datagram Congestion Control Protocol (DCCP) to support RRA and RDA applications. The theoretical upper bound of guaranteed minimum rate is derived and verified through simulations. We also compare the theoretical drop ratio of proposed DCCP-QoS mechanism with that of the DCCP-TCP-like mechanism and verifies the result by NS-2 simulation. The experimental results show that the proposed mechanism can provide minimum rate guarantee for real-time applications and maintain a lower packet drop ratio.

PUBLICATIONS

Journal Papers

1. **L. Ye**, Z. Wang, H. Che, H.C.B. Chan, and C.M. Lagoa, "Utility Function of TCP", *Computer Communications*, 32(5), pp. 800-805, 2009.

2. **L. Ye**, Z. Wang, H. Che, and C.M. Lagoa, "TERSE: A Unified End-to-End Control Mechanism to Enable Elastic, Delay Adaptive, and Rate", submitted to *IEEE/ACM Transactions on Networking*.

3. **L. Ye**, Z. Wang, and H.C.B. Chan, "A QoS-Aware DCCP Congestion Control for Multimedia Streaming", submitted to *IEEE Transation on Multimedia*.

Conference Papers

1. **L. Ye**, and Z. Wang, "A QoS-Aware Congestion Control Mechanism for DCCP", *IEEE Symposium on Computers and Communications*, 2009

2. **L. Ye**, Z. Wang, and H.C. B. Chan, "A Family of QoS Aware Congestion Control Protocols", *IEEE International Conference on Communications*, pp. 5897-5901, 2008.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Dr. Zhijun Wang, for his rigorous supervision of my research. I thank him for his support, patience and encouragement during my Ph.D. study. He unremittingly trained me to be a good researcher. He taught me how to find research issues and how to solve problems. Time after time, he showed me how to express my ideas and write academic papers. His vision, passion, and attitude towards the research deeply affected me. Without his help and support, this body of work would not have been possible. What I have learned and experienced during the time I spent in his research group will benefit me much in the future.

Next, I thank my co-supervisor, Dr. Henry Chan, for his great support and generous help during my Ph.D. study. He set a good example for me as accuracy and strict method. Also, I thank Dr. Hao Che for his great help and wise advices during my Ph.D. study. He made a lot of contributions to this research work. I wish to acknowledge my appreciation to Dr. Yuan Zheng, Dr. Meng Wang, Jin Xie, Yang Liu, Guobin Liu and Qin Li, who shared with me the pleasure of the Ph.D. study at the Hong Kong Polytechnic University. Furthermore, I would like to thank all my teachers from whom I learned so much in my long journey of formal education. They are Dr. Zhili Shao, Dr. Bin Xiao, Dr. Alvin Chan at the Hong Kong Polytechnic University, and many others.

Finally, but most significantly, I thank my wife and my parents for their continuous love, support, trust, and encouragement through the whole trip of my life. Without them, none of this would have happened.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The congestion control mechanism used in the Transmission Control Protocol (TCP) at the transport layer is a fully distributed, end-to-end traffic control mechanism. It relies solely on single-bit binary information feedback as input for the control, i.e., whether the forwarding path is congested or not. This binary information is acquired on the basis of source inferable information only, such as repetitive acknowledgements (ACKs) of the same segment, measured round-trip time, and/or ACK timeout, without the assistance of the network nodes for the control and regardless of the link technology and queuing mechanism used in the network nodes. This has made the proliferation of the Internet applications at global scale possible. An excellent example is the swift, ubiquitous adoption of World Wide Web due to its use of TCP as its underlying transport.

However, as Internet has evolved into a global commercial infrastructure, there has been a growing demand for Internet to support various real-time applications, such as voice over IP, IPTV (e.g., BBC iPlayer [50]) and multimedia streaming. These applications call for various real-time services, e.g., as defined in [106], including Hard Real-time (HRT), Real-time Delay Adaptive (RDA) and Real-time Rate Adaptive (RRA) services. Many of these real-time applications are built on Real-time Transport Protocol (RTP [102]) and RTP Control Protocol (RTCP [102]). RTP was designed to cooperate with RTCP to carry data with real-time characteristics such as video conference and voice over IP. RTCP is a sister protocol of RTP and it is designed to provide the control information for RTP. But RTP and RTCP do not have the ability to guarantee the Quality of Service (QoS). They still rely on the

routers to provide QoS, using technologies such as Integrated Services (IntServ [11]) with Resource ReserVation Protocol (RSVP [10]) support and Differentiated Services (DiffServ [9]) with Multiprotocol Label Switching (MPLS [98]) support. These approaches, however, require significant involvement of the network nodes and therefore have difficulty to scale to the global Internet. On the other hand, RTP itself does not have a standard TCP or UDP port on which it communicates. It relies on TCP or UDP as a bearer to transfer real-time data. While it is difficult to enable HRT solely based on an end-to-end control mechanism, we shall demonstrate that the unified end-to-end protocols can be developed to support RDA and RRA, which require soft minimum-rate guarantee, as well as the Non-Real-time Elastic (NRE) service [106].

A key challenge in designing end-to-end, traffic control protocols, in addition to TCP, is how to achieve desired performance in terms of user satisfaction, minimal impact on the existing TCP flows, and globally stable control, in the face of a highly interactive, end-to-end controlled mixture of traffic flows. The existing end-to-end transport layer traffic control mechanisms, such as TCP, variants of TCP (TCP Reno [107], TCP Vegas [12], Fast TCP [126], TCP New Reno [32], TCP Hybla [14], and TCP CUBIC [39]), and TCP-friendly protocols, were largely designed empirically. A consequence of such a design approach is that it is left unknown or difficult to reason whether or not the network will ever converge to a stable state and what kind of fairness/friendliness these protocols bring to themselves as well as TCP. Although the Internet has so far been more or less stable with the majority of applications running over TCP, things can easily get out of control if more and more empirically designed, end-to-end protocols are introduced into Internet without theoretically proved fairness/friendliness to TCP.

A promising approach to address the above problem, called utility-based optimization approach in this thesis, is to formulate the above problem as a distributed optimization problem aiming at achieving a given utility-related design objective, such as the maximiza-

tion of the sum of individual user utilities (utility-maximization for short, e.g., [106]) or user utility max-min, e.g., [15]. The targeted solution is a set of end-to-end control laws governing the behaviors of individual user flows that drive the network to an operational state where the design objective is achieved.

Although optimal results exist, there are some fundamental issues concerning the above utility-based optimization approach, which make it difficult to apply the existing results to the design of end-to-end protocols to support real-time applications. First, the utility-based optimization approach attempts to encode all the desired effects, such as QoS features and fairness, implicitly in the utility function. This makes the design of utility functions a difficult task. Second, attempting to achieve a global design objective means that the achievable user utility is, in general, a complex function of the traffic load and pattern, making it difficult to quantify the QoS features of a given service, especially a real-time service. Third, as a transport layer protocol providing the NRE service, TCP has played a dominant role in supporting major application layer protocols, such as HTTP, FTP, and TELNET. As a result, to be practically useful, a new protocol designed based on the utility-based optimization approach must demonstrate convergence and optimality in the presence of TCP controlled flows. As observed later in this thesis, the effective utility function of TCP is time-varying, i.e., a function of the current traffic load and pattern. This implies that in general, the utility function corresponding to other services must also be time-varying to be TCP aware/friendly. This requirement, however, is in conflict with the traditional definition of utility function that characterizes user satisfaction of a service and therefore should not change over time. Finally, since the actual price charged to the user for using a given service will have an impact on the user satisfaction of the service received, how the utility function design should be related to the pricing structure is still an open issue. All these issues have made the practical application of the utility-based optimization approach difficult.

We put forward a new design methodology to tackle the above problems. Although

having its root in the traditional utility-based optimization approach, the methodology gives entirely different meaning to the utility function than the traditional one. More specifically, in this methodology, the utility function only captures the elastic part of the user utility. To be fair to TCP and other services, the utility functions for all the services are defined to be identical and equal to the TCP utility function. Other than being encoded in the utility function, the QoS features are expressed explicitly as boundary conditions in the utility optimization problem. This methodology sets the foundation for the proposed TCP-Elastic Real-time SErvice (TERSE). TERSE enables NRE, RDA, and RRA services under a unified end-to-end congestion control protocol. Different services are turned on by properly setting a single parameter in the congestion control law only. This congestion control law is a generalization of the end-to-end TCP congestion control mechanism and degenerates to TCP when applied to NRE flows. The three services enabled by the protocol are fair/friendly to one another in the sense that they compete for network resources fairly, provided that the minimum guaranteed rates for RDA and RRA services are achieved. In the meantime, the control protocol is also resilient to resource shortages, caused by, e.g., link/node failures or over commitment of network resources. In other words, it has the ability to reduce its flow rate to the level that is lower than its soft minimum guaranteed rate, ensuring that Internet will not experience partial (such as starving TCP flows) or complete shutdown in the presence of resource shortages.

By combining the congestion control law in [83], our TERSE can guarantee that the network would be driven into a stable optimal state. Because we are using TCP utility function for all the QoS flows, the fairness to TCP and between themselves would be achieved. The design of the proposed protocol also leads to the finding of a utility function corresponding to the end-to-end TCP congestion control mechanism, which captures major TCP behaviors, quantitatively.

Many researchers were motivated to look for a possible interpretation of TCP conges-

tion control from an optimization-based, distributed traffic control framework [56] [61] [74]. To us, the implication of the finding such an interpretation is significant for two reasons. First, such an interpretation will unveil whether or not the TCP congestion control leads to globally stable operation and if so, what is the underlying global design objective the TCP congestion control protocol strives to achieve. Second, such an interpretation will pave the way for the TERSER methodology, which jointly with TCP, achieve globally optimal and stable traffic control.

As we shall explain in the related works, some existing optimization-based control laws successfully capture important aspects of the TCP congestion control, especially some variations of TCP that involve network nodes. However, in the existing work, the aperiodic, binary nature of end-to-end TCP congestion control and the slow start phase of TCP control are not captured. On the basis of the family of control laws given in [83], we develop a more accurate utility function of TCP. Our proposed TCP control law captures the essential behaviors of TCP, including slow start, congestion avoidance, and the binary nature of congestion feedback in TCP. We find that the utility function of TCP is linear in the slow start phase and is proportional to the additive increase rate and approaches the well-known logarithm function as the data rate becomes large in the congestion avoidance phase. We also find that understanding the slow start phase with a fixed threshold is critical to the bandwidth allocation between network flows.

Based on TERSE methodology and the TCP utility function, we propose the unified end-to-end QoS aware congestion control protocols for reliable services. Our proposed unified end-to-end protocols can support Best Effort Service (BE, which is the NRE service), Assured Forwarding Service (AF), Minimum Rate Guaranteed Service (MRG), and Minimum Rate Guaranteed and Upper Bounded Rate Service (MRGUBR). It degenerates to TCP in the case of the BE service. We use the fluid model to analysis the capacity of our proposed MRG protocol. Also, the MRG protocol is implemented in Linux-based system and tested

by cross Pacific file transmission.

For some real-time applications, the lost and retransmitted packets may be too late to be useful to the applications. The unneeded retransmitted packets may occupy the critical link's bandwidth and decrease the useful packets' rate. So it is necessary to design unreliable QoS aware protocol to support real-time applications.

The unreliable UDP [93] lacks congestion control mechanism and hence it may lead Internet into congestion collapse [33] as the real-time application using UDP grows fast. Moreover, by using Differentiated Services (DiffServ) [9], the bandwidth would not be fairly shared among UDP and TCP flows when they are in same class [103]. Due to these reasons, UDP traffic is always filtered out by some networks [72]. The Datagram Congestion Control Protocol (DCCP) [59] is a framework designed to provide congestion control for unreliable real-time traffic. DCCP can support multiple congestion control mechanisms (Congestion Control Identifiers, named CCID) and allow an application to choose the most suitable one. There are two main CCIDs: TCP-like (CCID2) [34] and TCP-Friendly Rate Control (TFRC, CCID3) [35] [36] [58]. But both TCP-like and TFRC cannot provide QoS guarantee.

For unreliable services, we develop an end-to-end QoS aware CCID (called DCCP-QoS) for DCCP protocol. It can guarantee a soft minimum rate and maintain a lower packet drop ratio in the mean time.

1.1 Motivation and Objective

The motivations for designing a family of end-to-end QoS aware traffic control protocols are:

1. Provide End-to-End Quality of Service (QoS) guarantee:

    - Today's Internet, there has been the growing demand to support different types

of real-time applications, such as multimedia streaming. All the existing QoS solutions are heavily involving with the support of technologies from the Internet core nodes, such RSVP [10] and MPLS [98]. But many parts of the Internet core nodes can only support the best effort TCP and nowadays QoS solutions cannot been applied to the whole Internet. It is necessary to design end-to-end QoS solutions for Internet to support real-time applications.

2. Global optimization concern:

   - The existing end-to-end transport layer traffic control mechanisms, such as UDP, TCP, variants of TCP and TCP-friendly protocols, were largely designed empirically. A consequence of such a design approach is that it is left unknown or difficult to reason whether or not the network will ever converge to a stable state and what kind of fairness/friendliness these protocols bring to themselves as well as TCP.
   - Traditional utility based optimization approaches have some fundamental issues which make it difficult to apply the existing results to the design of end to end protocols to support real-time applications.

To solve the above prbolems, our objective in this thesis is to design a family of end-to-end QoS aware traffic control protocols based on a well-defined methodology which possesses some desirable, provable properties, including fairness, stability, and optimality. Moreover, for all the control protocols in this family, the minimum non-local information required for the control is simply a discontinuous, aperiodic, single-bit binary indicator, indicating whether the forwarding path is congested or not. It means that the family of protocols is truely end-to-end and does not need the help from network nodes. This end-to-end feature makes the family of protocols easy to scale to the whole Internet.

1.2 Problem Formulation

The optimization problem of the flow rates in Internet can be expressed as below.

$$max \sum_{i=1}^{F} U_i(x_i) \tag{1.1}$$

subject to network constraints

$$\sum_{i:l\in Ł} x_i \leq B_l \qquad l \in Ł \tag{1.2}$$

with the QoS requirements: the Assured Forwarding Service (AF) requirements

$$x_i = \Lambda_i \qquad i \in AF \tag{1.3}$$

the Minimum Rate Guaranteed Service (MRG) requirements

$$x_i \geq \theta_i \qquad i \in MRG \tag{1.4}$$

the Upper Bounded Rate Service (UBR) requirements

$$x_i \leq \Theta_i \qquad i \in UBR \tag{1.5}$$

the Minimum Service Guarantee and an Upper Bounded Rate Service (MRGUBR) requirements

$$\theta_i \leq x_i \leq \Theta_i \qquad i \in MRGUBR \tag{1.6}$$

where $U_i(x_i)$ is the utility function for flow i; F is the total number of flows in the network; x_i is the rate of flow i; Ł is the set of links in the network; and B_l is the bandwidth of link l in Ł; Λ_i is the assured data rate of flow i; θ_i is the minimum guaranteed rate of flow i and Θ_i is the upper bounded rate of flow i.

Our aim is to find the traffic control law $\dot{x} = F(x, cg)$ for flow rate x, where $\dot{x}$ is the changing flow rate after the control, cg is a binary congestion indicator.

1.3 Contributions

Our contributions are:

1. We propose a new utility based protocol design methodology TERSE to design new transport protocols which prossess some desirable, provable properties, including fairness, stability, and optimality.

    - We demonstrate the fundamental issues and difficulties concerning the traditional utility-based protocol design approach, including (a) design utility function to enable desired service features; (b) quantify QoS features; (c) relate utility-based approach to the pricing structure; (d) deal with the time-varying TCP utility function.
    - TERSE methodology redefines the utility function which accounts for only the elastic part of user utility and expresses the inelastic part explicitly as flow-level constraints. It provides a unified end-to-end traffic control protocol framework that enables the Non-Real-time Elastic service, Real-time Delay Adaptive service, and Real-time Rate Adaptive service.

2. We derive a global utility function and the corresponding optimal control law, known as TCP control law, which maximizes the global utility.

    - The TCP control law captures the essential behaviors of TCP, including slow start, congestion avoidance, and the binary nature of congestion feedback in TCP.
    - The TCP utility function is linear in the slow start phase and approaches the well-known logarithm function in the congestion avoidance phase. We also find that the slow start phase with a fixed threshold is critical to the bandwidth allocation between the flows.

3. Based on TERSE methodology and the proposed TCP utility function, we derive the unified end-to-end QoS aware congestion control protocols for reliable service.

   - The unified end-to-end QoS aware congestion control protocols support a set of class of services, such as Assured Forwarding Service (AF), Minimum Rate Guaranteed Service (MRG), Upper Bounded Rate Service (UBR), and Minimum Rate Guaranteed and Upper Bounded Rate Service (MRGUBR). These services can be enabled only by setting a tunable function $r(x)$, and the protocols degenerate to the best effort TCP when $r(x)$ is set as 1.
   - We use the fluid model to analysis the capacity of MRG comparing to TCP protocol. The NS-2 simulations show that our MRG can significant improve the average target rate.
   - We implement the MRG protocol in Linux operation system. The cross Pacific transmission testing shows that it can achieve more than 1.2Mbps throughput performance, 150% higher than the most widely deployed TCP-reno and 50% higher than TCP-cubic.

4. We also derive a QoS aware DCCP congestion control mechanism for unreliable service which can be used for multimedia streaming.

   - We redefine the congestion indicator for the unreliable service by the help of STT to accurately response to the network condition. It can faithfully indicate the load of the path from the sender to the receiver.
   - We also use the fluid model to analysis of the minimum guaranteed rate of our proposed DCCP-QoS. We test it by NS-2 simulations.
   - We model the average congestion cycle of the DCCP-QoS flow and the DCCP-TCP-like flow and compare their lost packet ratios when they achieve the same

bandwidth. Our proposed DCCP-QoS can maintain a drop ratio lower than that of DCCP-TCP-like.

1.4 Outline

The rest of this thesis is organized as follows. In Chapter 2, we discuss the related works. In Chapter 3, we propose our new utility-based protocol design methodology TERSE. In Chapter 4, we derive the TCP utility function by reverse engineering TERSE with TCP congestion control law. In Chapter 5, we present the unified end-to-end QoS aware traffic control protocols for reliable service and test them. In Chapter 6, we present a QoS aware DCCP CCID named DCCP-QoS. Chapter 7 gives the conclusion of our works and a brief discussion on future work.

CHAPTER 2
LITERATURE REVIEW

In this thesis, we focus on designing a family of optimization based end-to-end QoS-aware traffic control protocols. We discuss the related works from two sections, i.e. Quality of Service and utility-based protocol design methodology.

2.1 Quality of Service Aware Transport Layer Protocol

In today's Internet, there has been a growing demand to support different types of applications with multimedia steaming. The Internet applications can be classified into four classes [106]: Non-Real-time Elastic (NRE), i.e., best effort, Hard Real-time (HRT), Real-time Delay Adaptive (RDA) and Real-time Rate Adaptive (RRA) applications. For example, voice-over-IP and video conference are HRT applications and need strict delay constraints, while IPTV (e.g., BBC iPlayer [50]) can be a RRA application which can tolerate some rate fluctuations. The Real-time Transport Protocol (RTP) [102] and the Real-time Transport Control Protocol (RTCP) [102] are the two major congestion control protocols for real-time applications. RTP provides facility for jitter compensation and detection of out of sequence arrival in data. RTCP is used to monitor transmission statistics and QoS information. However, both RTP and RTCP themselves do not provide mechanisms to guarantee timely delivery, i.e., do not give any QoS guarantees. They still rely on the router to provide QoS, using technologies such as Integrated Services (IntServ [11]) with Resource ReserVation Protocol (RSVP [10]) support and Differentiated Services (DiffServ [9]) with Multiprotocol Label Switching (MPLS [98]) support.

IntServ is a framework that can provide the fine-graind QoS service. The idea of IntServ is for a flow to make every routers through the path to guarantee individual reservation specified by the Flow Specs. It can guarantee fine-graind QoS requirements such as jitter, delay and etc. IntServ needs RSVP protocol to signal it across the network. Because RSVP requires every routers to reserve corresponding resource for every flows, the IntServ is too heavy to scale to wide area networks, such as Internet. Comparing to IntServ, DiffServ is a simple and lightweighted corse-graind QoS service based on IP networks. It is often implemented by the help from MPLS to mark different classes of flows. But not all the routers in Internet can differentiate several classes of services, DiffServ can not be applied to everywhere in Internet.

There are so many works that have been done to develop transport layer protocols to provide Quality of Service. We do not want to provide an exhaustive survey of the literature. We present some of them on how they are designed to support reliable applications and unreliable applications.

2.1.1 Quality of Service for Reliable Applications

There are a lot of works focused on the design of QoS-aware congestion control protocols [2] [4] [25] [26] [45] [123]. Resilent overlay network (RON) [4] is a framework can provide a lot of service but it involves a lot with the cooperation from network nodes. In [25], Duan et al. constructed a virtual end-to-end QoS aware service overlay network (SON) via service layer agreement (SLA), in which the SLA needs helping from the network nodes. Hence SON is not a really end-to-end QoS aware solution. In [26], Elwalid et al. proposed the MATE mechanism which can provide adaptive traffic enigeering. But the MATE is based on MPLS network and hence it is not the end-to-end solution. In [45], a real-time utility function construction method was proposed, and the second order utility optimization is used to do congestion control, in which the real-time traffic and best effort traffic approach the utility

proportional fairness. This framework, however, is tied to the Active Queue Management (AQM) mechanism used in the routers and hence is not an end-to-end solution. In [123], a distributed flow control algorithm was designed to drive the network into a proportional (or max-min) fair state. This distributed flow control algorithm requires the feedback of the link price from the network nodes in the forwarding path. Hence it is not an end-to-end protocol. To the best of our knowledge, there has been no end-to-end, optimization-based, QoS-aware congestion control protocol being proposed to date.

Also, many researchers did a lot of works [1] [7] [21] [105] [111] [20] on how to deliver the multimedia streaming data to the end users. In [1], Ahmed et al. developed a cross layer adaptive streaming approach to transfer MPEG-4 data which needs the support of differentiated service framework. Like the work in [1], the distributed QoS multimedia transport mechanism [7] still needs the help of network nodes. Others [21] [105] [111] focus on delivery of multimedia streaming data by special encoding methods or frameworks and may not be applied to other multimedia streaming data in Internet.

2.1.2 Quality of Service for Unreliable Applications

Unlike UDP which has no congestion control mechanism, the Datagram Congestion Control Protocol (DCCP) [59] is a framework designed to provide congestion control for unreliable real-time traffic. DCCP can support multiple congestion control mechanisms (Congestion Control Identifiers, named CCID) and allow an application to choose the most suitable one. There are two main CCIDs: TCP-like (CCID2) [34] and TCP-Friendly Rate Control (TFRC, CCID3) [35] [36] [58]. DCCP is considered to be a promising control protocol for real-time applications [38] [129] [101]. But both TCP-like and TFRC cannot provide QoS guarantee. The TCP-like mechanism is similar to TCP, i.e., using Additive Increase and Multiplicative Decrease (AIMD) control laws, but without packet retransmission. The TFRC congestion control is an equation based rate control mechanism, the transmission rate is dependent on

the current Round Trip Time (RTT) and packet loss ratio.

There are many research works [6] [38] [47] [65] [85] [86] [101] [115] [119] [129] [116] [122] have been done to study the performance of DCCP protocol. Hisamatsu et al. [47] studied the DCCP and RED interacted by the fluid model. In [86], DCCP was been used to transfer adaptive video. Tong et al. [115] investigated how to improve fairness between TCP and DCCP in delivering real-time applications. In [129], DCCP has been tested to transfer H. 263 multimedia streaming. In [38], SVC coded video transmission were tested based on DCCP. There are works [101] [122] have been done to investigate the performance of DCCP on delivering VoIP streaming. A research work [116] found that DCCP results in poor quality for voice traffic compared to UDP, and some variants of DCCP have poor quality for voice traffic even compared to TCP. Azad et al. [6] found that TCP outperforms DCCP CCID2 and CCID3 in throughput and packet drop ratio. There is also a work [65] found that the unfair bandwidth sharing between DCCP and TCP still exists. A research work [116] found that DCCP results in poor quality for voice traffic compared to UDP, and some variants of DCCP have poor quality for voice traffic even compared to TCP.

Except the two basic congestion control mechanisms, many variants have been proposed and evaluated to test how efficient of DCCP to support different real-time applications [13] [49] [65] [71]. In [71], an interesting mechanism which inversely modifies the video quality to adapt to network condition was proposed. There is even a work [49] which designed a DCCP variant with retransmission mechanism. This retransmission DCCP variant can reduce wastefully retransmitted packets and increase the the MPEG-4 video quality comparing to the conventional scheme. Cai et al. [13] proposed an AIMD variant DCCP to improve QoS for one-hop wireless infrastructure networks. Their AIMD control law is modified by tuning increase and decrease factors. But these variants of DCCP cannot provide QoS guarantee. In [65], Lai supplemented DCCP with fast recovery mechanism which can achive fainess between DCCP and TCP flows.

2.2 Utility-based Protocol Design Methodology

The end-to-end TCP congestion control mechanism was developed empirically. There were quite a few existing mathematical techniques that can faithfully model TCP behaviors, notably, the models given in [66] [76] [87]. However, these techniques are descriptive by design, meaning that while being able to faithfully mimic the TCP behaviors, they provide no knowledge about the underlying design objective of TCP and its convergence and stability properties. Researchers have also made significant efforts to develop variations of TCP congestion control mechanisms (e.g., TCP Vegas [12], TFRC [42], DCCP [59]) and TCP-friendly protocols (e.g., see [51] and the references therein). However, similar to the congestion control mechanism of TCP, the congestion control mechanisms provided by these protocols were designed empirically without provable properties.

Significant research efforts [2, 15–17, 23, 25–29, 37, 39, 45, 51, 53–56, 61–64, 66, 67, 69, 74, 75, 79, 83, 95, 108, 114, 117, 118, 121, 124, 127, 128] have been made on the development of optimization-based, distributed traffic control laws that achieve certain global design objectives (e.g., the utility-based approach, including utility-maximization, i.e., maximization of the sum of individual user utilities [106], and utility max-min [15]). Some work focuses on the understanding of TCP behaviors from the utility-maximization point of view [56] [61] [74] [118]. In their seminal work, Kelly et al. [56] demonstrated that a utility function of $log(x)$ form (x is a flow data rate) leads to the so called proportional fairness, or proportional-fair allocation of the network resources, and a control law that exhibits Additive-Increase-and-Multiplicative-Decrease (AIMD) behavior, resembling the TCP behaviors. However, as pointed out in [56], the proportional-fair allocation may not always be in favor of flows with small Round-Trip Times (RTTs) in terms of rate allocation, a typical characteristic of TCP behavior. Nevertheless, the work in [56] has inspired significant research interests in an attempt to further understand the TCP behavior from an optimization point of view. In [118], by constructing a utility function inversely proportional to RTT,

Vojnovic, et. al. was able to show that the control law for utility-maximization not only exhibits AIMD behavior but also is in favor of flows with smaller RTT values. In [61], Kunniyur, et al. showed that the TCP behavior without the slow start phase can be modeled using a framework based on a variation of the utility-maximization approach. By taking into account the randomness of packet loss, their model leads to a TCP utility function of the form $-1/x$ as x becomes large, similar to the one used in [118]. Low [74] studied various variations of TCP using a primal-dual nonlinear programming technique, that solves the utility-maximization problem. While the primal algorithms (i.e., control laws) capture the TCP window control behavior (without the slow start phase) at the TCP source node, the dual algorithms translate into given Active Queue Management (AQM) algorithms running at the network nodes. In [23] [53], Deb, et al. studied the optimization problem of the network with multicast service. Many researchers did a lot of study on the heterogeneous network from the optimization view [62] [69] [108] [114]. In [79], Miller, et al. developed a congestion control mechanism which can achieve max-min optimization of network with arbitrary strict increasing utility function services. Voice [117] designed a congestion control algorithm for the Internet-like network by using the primal-dual model.

Stimulated by Kelly's model [56], many researchers investigated the transmission protocols and Internet from the optimization view [18, 22, 24, 41, 43, 68, 69, 73, 77, 89, 96, 97, 104, 112, 113, 124, 125, 131]. There are some works [18] [89] which studied optimization problem from a cross-layer view and not only based on the transmission layer of network. Deb investigated effect of the queue information in congestion control models [24]. Many works [41] [43] [73] [77] [96] [97] [112] [113] have been done on studying different properties, such as fairness, stability, and optimality of transmission layer protocols, such as TCP and its variants. Fluid model were be used to analysis the performance of congestion control protocols [104]. Besides studies on transmission layer protocols, there also is work [125] which developed active queue management (AQM) algorithm based on Kelly's model.

The fundamental design issues regarding the development of QoS services over the Internet was studied by Shenker [106], from a utility-based perspective. This utility-based approach associates each application-based flow with a user utility function $u(x)$, measuring the degree of satisfaction the user perceives at the allocated flow rate x. The design objective is to maximize the sum of individual user utilities, i.e., utility-maximization. The rationale behind this approach is the fact that the user satisfaction should be the ultimate performance measure of QoS, rather than network centric performance metrics, such delay and packet loss rate. In [106], Shenker identified four service classes or user utilities to meet various application needs, i.e., NRE, HRT, RDA, and RRA, as shown in Figure 2.1. These utility functions reflect to what degree a user is satisfied with the service at a given allocated rate. NRE (or elastic) applications are rather tolerant of delays and they also appear to have decreasing marginal degradation due to incremental increases in delays. HRT applications need their data to arrive within a given delay bound. They do not care if packets arrive earlier, but the applications perform very badly if packets arrive later than their bound. RDA applications are implemented to be rather tolerant of occasional delay bound violations and drop packets. But the performance degrades badly as soon as the bandwidth share becomes smaller than the application's intrinsic minimum rate. RRA applications can adjust their transmission rate in response to network congestion. Certainly at high bandwidths the marginal utility of additional bandwidth is very slight because the signal quality is much better than human need. It also appears that at very small bandwidths, the marginal utility is very slight because the signal quality is unbearably low. There are many research [3] [70] have been made to study the utilty functions of real-time applications.

A few results are available on the design of utility-based, QoS-aware congestion control protocols [15] [45] [123]. In [45], Harks, et al. proposed a utility function construction method and used the second order utility optimization to do congestion control, which ensures that the real-time and best effort flows approach the utility proportional fairness. This

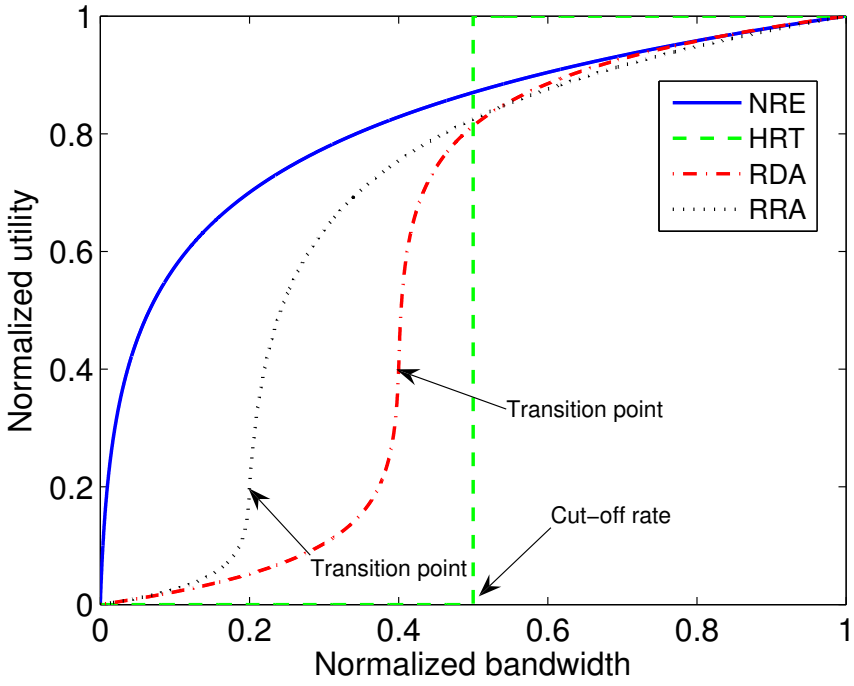

Figure 2.1. Four types of utility functions

framework, however, is tied to the AQM mechanism used in the routers and hence is not an end-to-end solution. In [123], Wang, et al. designed a distributed flow control algorithm to drive the network into the proportional (or max-min) fair state. This distributed flow control algorithm requires the feedback of the link price from the network nodes in the forwarding path. Hence again, it is not an end-to-end protocol. In [15], Cao, et al. found distributed control laws that achieve the utility max-min design objective. Again, the solution requires explicit information feedbacks from the network nodes and hence, is not an end-to-end solution.

A critical drawback of the existing solutions mentioned above is that they are point solutions with limited design scopes, such as modeling of TCP, modeling of variations of TCP tied to specific AQM mechanisms, QoS-aware control protocol that achieves a specific design objective, such as min-max or proportional fairness. Moreover, most of the existing solutions require significant involvement of network nodes for the control and hence are not end-to-end solutions.

Very recently, based on the sliding mode technique in control theory, Movsichoff, et al, [83] found a large family of optimization-based, distributed control laws, which drives the network to a globally stable state where the sum of utility functions is maximized. The needed information feedback for the control is discontinuous, aperiodic, and binary. These control laws provide the much needed framework to allow various end-to-end optimization-based traffic control protocols to be designed based on theory, rather than empirical design. In the method, the key QoS requirements are expressed as flow level constraints, not in the utility function. So we can use the same utility function for different type of application classes to avoid the difficulties for utility design.

CHAPTER 3

TERSE DESIGN METHODOLOGY

3.1 Overview

This chapter puts forward the design methodoolgy of TCP-Elastic Real-time SErvice (TERSE). TERSE is an end-to-end traffic control solution which can provide a unified end-to-end traffic control protocol that enables the non-real-time elastic service (i.e., the same as the one under the TCP control), real-time delay adaptive service, and real-time rate adaptive service. A specific service is enabled by properly setting a single parameter in the protocol only. While having its root in the well-known utility-based optimization approach, TERSE design methodology successfully addresses the limitations of the utility-based optimization approach. It leads to the successful design of the unified traffic control protocol, which enjoys some provable properties, including fairness, convergence, and stability. Other than the traditional utility-based approach which expresses the QoS features implicitly in the utility function, TERSE design methodology defines the utility functions of all the QoS services the same as TCP utility function and expresses the QoS features explicitly as boundary conditions of the optimization problem.

The rest of the chapter is organized as follows: Section 2.2 discusses the issues concerning about the traditional utility-based protocol design method. Section 2.3 introduces our utility-based design methodology of TERSE. Section 2.4 demonstrates the unified QoS congestion control protocol. Finally, Section 2.5 concludes the chapter.

3.2 Issues Concerning Traditional Utility-Based Approach

TCP and its variations are all designed empirically. Such a design makes the network unknown or difficult to reason whether it will ever converge to a stable state and what kind of fairness/friendliness could be achieve to these protocols as well as TCP. Although today's Internet has been stable with the majority of the applications running with TCP, it will get out of control if more and more empirically designed protocols are introduced into Internet. One of the most difficult challenges in designing end-to-end traffic control protocols is how to achieve desired performance with the global stable and fair state, impact minimally on the existing TCP flows.

Traditional utility-based optimization approach is to formulate the above problem as a distributed optimization problem aiming at achieving a given utility-related design objective, such as the maximization of the sum of individual user utilities (utility-maximization for short, e.g., [106]) or user utility max-min, e.g., [15]. The targeted solution is a set of end-to-end control laws governing the behaviours of individual user flows that drive the network to an operational state where the design objective is achieved. In the following discussion, we elaborate the fundamental issues concerning this traditional utility-based approach.

Although the traditional utility based methodology introduces the result that can achieve the global optimization, there are some fundamental issues which make it difficult to be applied to the design of end-to-end protocols to support real-time applications. First, the traditional utility based optimization approach try to encode all the QoS features and fairness objective into the utility function. This makes the design of utility function very difficult. Second, attempt to achieve a global design objective means that the achievable user utility is, in general, a complex function of the traffic load and pattern, making it difficult to quantify the QoS features of a given service, especially a real-time service. Third, TCP is the dominant transport layer protocol to provide NRE service, such as HTTP, FTP, and

TELNET. As a result, to be practically useful, a new protocol designed based on the utility-based optimization approach must demonstrate convergence and optimality in the presence of TCP controlled flows. As observed later in this thesis, the effective utility function of TCP is time-varying, i.e., a function of the current traffic load and pattern. This implies that the utility function of other services must also be the a similar time-varying pattern to be TCP friendly/fair. This requirement, however, is in conflict with the traditional definition of utility function that represents user satisfaction of a service and therefore should not change over time. Finally, since the actual price charged to the user for using a given service will have an impact on the user satisfaction of the service received, how the utility function design should be related to the pricing structure is still an open issue. All these issues have made the practical application of the utility-based optimization approach difficult.

As mentioned above, there are four fundamental issues concerning the utility-based approach, including (a) how to design utility functions to enable desired service features; (b) how to quantify QoS features; (c) how the utility-based approach and the pricing structure are related; and (d) how to deal with the time-varying TCP utility function. In what follows, we elaborate more on these issues.

Utility Function Design and Quantification of QoS Features: By the traditional definition, a user utility function measures to what degree a user is satisfied with a given service provided to him/her. It does not provide any information about the relative degrees of user satisfaction across different service classes. This is not an issue when network resources are not shared by flows from different service classes. However, it becomes one when flows from different service classes have to compete with one another for the network resources. In such a case, the relative value ranges and shapes of utility functions for different service classes will have a strong effect on the final rate allocation, making it difficult to design utility functions for different service classes. Moreover, for the traditional utility-based approach, the final rate allocated to a flow, whether it is non-real-time or real-time, is dependent on the

overall traffic load and pattern. This makes it difficult to quantify the QoS features for real-time service classes. Unfortunately, to the best of our knowledge, no literature has explicitly addressed these issues.

To understand the above issues better, let us take a look at the traditional NRT, RDA, RRA, and HRT utility functions and assume that they are all normalized, as given in Figure 2.1, i.e., the degree of user satisfaction reaches one when the received rate goes to infinity and zero when the received rate is zero for all services. We argue that the utility-based approach using such utility functions may lead to undesirable resource allocations. For example, in the case of the utility-maximization, with the utility functions in Figure 2.1, an HRT flow may not be allocated any link bandwidth, if it attempts to share the bandwidth of a link with a large number of NRE flows. This is simply because an NRE flow can achieve a rather high utility value at a rate much lower than the cut-off rate for an HRT flow and hence, the sum of the user utilities is maximized when the link bandwidth is allocated to NRE flows only. For the similar reason, both RDA and RRA flows may also perform poorly in the presence of a large number of NRE flows.

The situation can be even worse when the utility max-min design objective, which attempts to equalize the utility values for all the flows, is used. With an HRT flow attempting to share a link bandwidth with a large number of NRE flows, a feasible solution may not even exist. On one hand, we note that an HRT flow receives either zero utility or full utility (i.e., value 1) depending on whether the allocated bandwidth is less or no less than its cut-off rate. As a result, either all the flows receive utility value 1, which is obviously impossible (note that a NRE flow achieves utility value 1 at the expense of an extremely high bandwidth allocation as evidenced in Figure 2.1), or all the flows receive zero bandwidth. Similar problem occurs when a RDA or RRA flow coexists with a large number of NRE flows. In this case, the former is likely to experience poor performance (i.e., low utility) while consuming a sizable amount of link bandwidth, causing poor performance for NRE flows as

well.

Relation to Pricing Structure: The above examples suggest that the traditional utility functions must, somehow, be redesigned to reflect the relative importance of different service classes, which has a lot to do with the pricing structure in use. This makes sense from a user's perspective because a user generally will set his/her expectation of the service quality in proportion to the price he/she actually pays for the service. In other words, the user utility must, to some extend, reflect the price the user pays for the service. This also makes sense from the perspective of an Internet Service Provider (ISP) in the sense that the ISP would be willing to give better, higher priority service to users who pay more for the service. As a result, a utility-based solution should somehow account for such price-induced effects.

Based on the above understanding, a simple fix to the problem is to slightly modify the utility functions in Figure 2.1 by allowing the value ranges of the utility functions to be proportional to the price paid by the user for the use of one unit of network resources (the exact functional relationship is not important for the current discussion). For example, if we magnify the value range of the RRA utility in Figure 3.1 by, e.g., 2 times, the rate allocation will now be much more in favor of the RRA flows than the case with normalized utilities. Obviously, the need to incorporate the pricing effect in the utility function further complicates the design of utility functions.

Interacting with TCP: Since TCP is a dominant transport layer protocol, the utility-based approach must take its impact on TCP performance into account, e.g., the rate allocation needs to be TCP friendly. An interesting observation made in this chapter is that the effective TCP utility function is time-varying, i.e., it changes as traffic load fluctuates. This may sound a bit odd since a utility function is meant to serve as a measure of the degree of user satisfaction and hence, it should not be dependent on the traffic load in the network. The reason for this being true is that the TCP congestion control mechanism was not designed

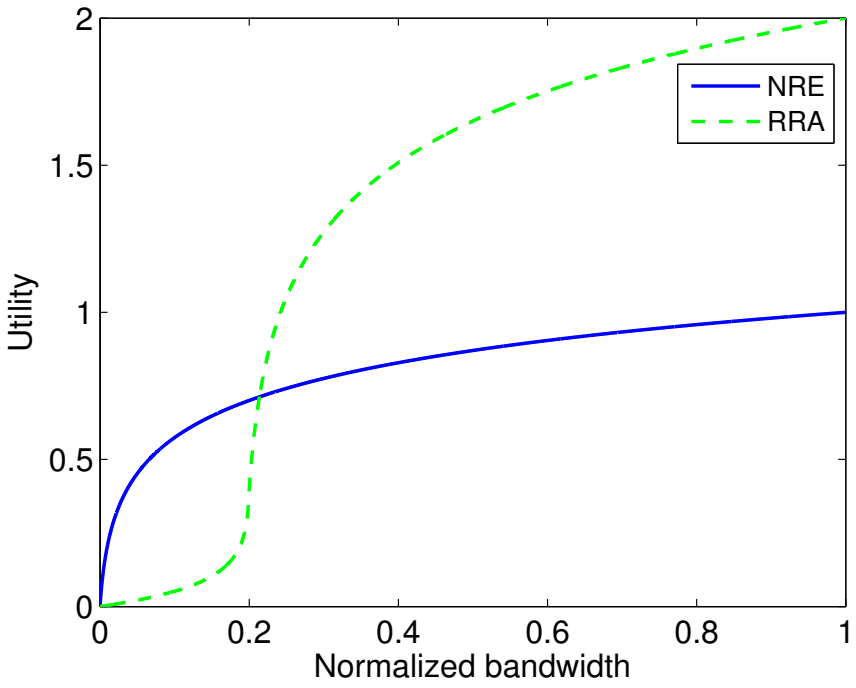

Figure 3.1. Two types of utility functions with different scales

with a user utility function in mind. As a result, it unintentionally leads to an effective TCP utility function whose value range and shape change from time to time. This observation suggests that to be TCP friendly, the time-varying components might need to be incorporated in the utility functions for new service classes as well. This however, makes it even harder to develop new service classes using the utility-based approach.

In summary, we conclude that for the utility-based approach to be practically useful for the design of new end-to-end transport layer protocols, the traditional utility function definition must be modified in some way to overcome the above difficulties. In the following subsection, we propose a new methodology rooted in the utility-based approach, aiming at overcoming these difficulties.

3.3 TERSE Methodology

In this section, we propose our TERSE design methodology aiming at addressing those issues, which leads to the design of the proposed protocol. The root cause of the aforementioned difficulties can be mainly attributed to the idea of encoding all the desired service features into the utility functions, while attempting to achieve a global design objective involving all the user utilities. On one hand, achieving such a global design objective, whether it is utility-maximization or utility max-min, is desirable because it enables well-defined fair sharing of the network resources among flows. On the other hand, it means that the actual resources allocated to flows of different service classes are a function of the overall traffic load and mixture. As a result, the desired service features of individual flows, including HRT features, may have to be compromised for the sake of fairness among all the flows, which is undesirable.

Basic Idea: A key idea in our methodology is to redefine the utility function such that it accounts for only the elastic part of user utility and letting the inelastic part be explicitly

expressed as flow-level constraints (or boundary conditions in the utility-based optimization problem). For example, the traditional HRT utility function in Figure 2.1 encodes a pure inelastic user utility, i.e., totally satisfied or totally unsatisfied depending on whether the cut-off rate is achieved or not. In our approach, this inelastic feature can be explicitly expressed as a flow-level constraint: $x =$ cut-off rate. As another example, for the service class corresponding to either RRA or RDA utility function in Figure 2.1, the user satisfaction of the service grows slowly in the convex part of the utility and starts to increase faster in the concave part of the utility. Usually, for this kind of service, a soft minimum guaranteed rate at the joint between the convex part and concave part of the utility function (i.e., the transition point in Figure 2.1) needs to be allocated to a flow to ensure that the user is reasonably happy with the service. In other words, we can view the RRA/RDA utility function as composed of two parts: an inelastic part requiring a minimum guaranteed rate and an elastic part corresponding to the concave part of the utility function in Figure 2.1. In our approach, the inelastic part is explicitly expressed as a flow-level constraint, i.e., x greater than the rate at the transition point, and the utility function only encodes the concave part of the utility functions in Figure 2.1 and other possible effects, such as pricing.

The implication of the above approach is significant. First, expressing inelastic features as flow-level constraints, or boundary conditions, in the problem space make these features explicitly quantifiable and not subject to traffic load and traffic mixture. This is also in better alignment with most existing pricing structures where the inelastic features are generally associated with some usage fees such as per unit guaranteed bandwidth usage charge. In return, the inelastic features must be protected from being compromised, which is enforced explicitly in our approach.

Second, involving only elastic components in the utility design makes it easier to justify and interpret the meaning of a specific choice of global design objective. In particular, in our protocol design, we choose to use the utility-maximization approach with all service

classes having the same utility function: the effective TCP utility function. This can be easily interpreted as providing fair share of network resources for the elastic part of the user utility across all service classes. More specifically, the elastic part of any traffic flow belonging to any service class will behave just like TCP and hence is guaranteed to be TCP friendly/fair, provided that the inelastic part of traffic flow achieves its target rate. This automatically resolves the issue concerning the time-varying nature of the TCP utility function. Note that other effects, such as pricing effects corresponding to the elastic components can also be incorporated to further differentiate the services by e.g., adding a weight proportional to the monthly fee to the corresponding utility function.

Finally, as a byproduct of extracting the inelastic features from the utility functions, the flow rate corresponding to the convex part of the traditional utility functions in Figure 2.1 is now outside the problem space. In other words, in the problem space constrained by the network resource and flow-level constraints, the utility functions are concave, freeing us from having to deal with a difficult non-convex optimization problem.

Discussion: Note that the HRT service cannot be supported by the protocol developed in this chapter. Our position is that the end-to-end transport layer protocol alone can only provide soft performance guarantee. Any HRT performance guarantee calls for sophisticated call admission control and resource reservation involving network nodes.

We assume that TCP will continue to be a dominant transport layer protocol to support NRE applications. Our aim is then to design a unified, end-to-end transport layer protocol that generalizes TCP for the support of RDA and RRA applications. This protocol degenerates to TCP congestion control when applied to the NRE flows.

Here we note that a real-time service user who receives the minimum guaranteed rate is reasonably happy with the service already. This makes it less important as to how to design the utility functions (i.e., the elastic parts of the utility functions in Figure 2.1) to

further differentiate the services. Instead, we simply let all three services share the same utility function. Of course, more elaborate utility can be used, by, e.g., adding a price-dependent weight to each utility in the utility-maximization objective. As we shall show later, this approach leads to an end-to-end traffic control protocol that *unifies* the controls for all three services. Service differentiation for the three services is achieved through a proper setting of a single control parameter in the protocol. The control parameter is set through the minimum required rate. The best effort service sets a zero minimum rate, and other service may have a non-zero rate.

Since the NRE service has been supported by TCP with great success, we simply adopt the TCP utility function (to be derived shortly) to be shared by all three service classes. This solution unifies the existing end-to-end TCP congestion control with the control of the other two service classes, rendering the TCP congestion control a special case of the proposed protocol. Moreover, this solution addresses the difficult issue as to how to cope with a dynamically changing TCP utility function when designing a new protocol on the basis of the utility-based approach. With this solution, all three service classes are subject to the same dynamics of network conditions for the sharing of the network resources. As such, the term 'fairness/friendliness' is now well defined. Namely, a TCP flow receives fair share of the network resources with the elastic parts of the flows belonging to the other two service classes, which also share network resources fairly among themselves.

Note that the negative impact on TCP performance due to the introduction of minimum guaranteed rates for real-time services must, in principle, be avoided or mitigated by a call admission control mechanism at global scale. In other words, the minimum guaranteed rates should not be severely compromised in the name of having to be friendly/fair to TCP. Otherwise, any attempt to enable meaningful QoS features and pricing models is guaranteed to fail. Nevertheless, in this chapter, we demonstrate that due to the softness of control, the current solution does not require call admission control to work properly.

In summary, the proposed methodology overcomes the difficulties that plague the traditional utility-based approach and sets foundation for TERSE. By redefining the utility function as the effective TCP utility function characterizing only the elastic part of the user utility, the proposed methodology makes the design of a new end-to-end protocol that unifies the control of NRE, RRA, and RDA service classes possible. Note that the HRT service class cannot be supported by the protocol developed in this thesis, which calls for hard rate guarantee and hence call admission control.

The above design methodology sets the foundation for TERSE. In the following chapters, the protocol design aspect of TERSE is described. It is composed of two steps. The first step is to derive the effective TCP utility function. Then this utility function is used in the utility optimization problem in the second step to derive the protocol that unifies the control of NRE, RRA, and RDA service classes.

3.4 A Family of QoS Aware Congestion Control Laws

This family of control law provides not only the minimum rate guaranteed service needed by TERSE, but also several other services. The distributed control law [83] that solves the optimization problem in Eqn. (1.1) is given by (for simplicity, we omit the index i),

$$\dot{x} = \{z(t,x)[f(x) - (1 - \bar{c_g} \times r(x))]\}_{x=0}^{+} \tag{3.1}$$

with

$$f(x) = 1 - e^{-\partial U(x)/\partial x} \tag{3.2}$$

and

$$(y)_{x=0}^{+} = \begin{cases} max(y,0) & \text{if } x = 0 \\ y & \text{if } x \neq 0 \end{cases} \tag{3.3}$$

where x is the flow rate; $U(x)$ is a differentiable concave and strictly increasing function of x; $z(x,t)$ can be any strictly positive function, satisfying some loose condition (see [83]);

cg is the binary congestion indicator (cg=1 if the packet forwarding path is congested and 0 otherwise); $\bar{cg}$ is the logical negation of cg; and $r(x)$ is a scalar factor, taking different values for different types of traffic. It is 1 for Best Effort (BE) flow. For flow with Minimum Rate Guarantee Service (MRG), i.e., a flow-level constraint: $x > \theta$ is added to the optimization problem in (1) and (2), $r(x)$ is given by:

$$r(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ r^m_{max} > 1 & \text{if } x < \theta \end{cases} \tag{3.4}$$

Here r^m_{max} is a tunable parameter. For an HRT application, it needs a constant data rate no matter what the network condition is. Hence the end-to-end congestion control mechanism cannot be applied to this type of applications. Here we aim to do congestion control for RDA and RRA applications which can tolerate some rate fluctuations. A user using such type of service should be happy if the average rate is above the minimum guaranteed rate. Note that the above family of control laws applies to any concave utility function; enable minimum rate guaranteed services; and allow flexible engineering of $z(x,t)$ function to realize various control behaviors.

Also by assigning different functions to $r(x)$, the different control laws can be derived to support different QoS requirements. The $r(x)$ for Assured Forwarding (AF) flow is set as follow:

$$r(x) = \begin{cases} r_{min} < 1 & \text{if } x \geq \Lambda \\ r_{max} > 1 & \text{if } x < \Lambda \end{cases} \tag{3.5}$$

where Λ is the target rate.

The $r(x)$ for Upper Bounded Rate Service (UBR) flow is:

$$r(x) = \begin{cases} r^M_{min} < 1 & \text{if } x > \Theta \\ 1 & \text{if } x \leq \Theta \end{cases} \tag{3.6}$$

where Θ is the upper bounded rate.

The $r(x)$ for Minimum Rate Guarantee and an Upper Bounded Rate Service (MR-GUBR) flow is:

$$r(x) = r^m(x) \times r^M(x) \tag{3.7}$$

where

$$r^m(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ r^m_{max} > 1 & \text{if } x < \theta \end{cases} \tag{3.8}$$

and

$$r^M(x) = \begin{cases} r^M_{min} < 1 & \text{if } x > \Theta \\ 1 & \text{if } x \leq \Theta \end{cases} \tag{3.9}$$

where θ is the targeted minimum rate, Θ is the upper bounded rate.

Moreover, for all the control laws in this family, the minimum non-local information required for the control is simply a discontinuous, aperiodic, single-bit binary indicator, cg, indicating whether the forwarding path is congested or not. This makes it possible to implement this family of control laws end-to-end. Also, this family of control laws is globally stable and optimal, as shown in [83]. It is also shown in [83] that the corresponding family of control laws in the discrete time domain with a finite time interval leads to stable and near optimal control. Clearly, this family of control laws provides the much needed mathematical tool for the implementation of the TERSE methodology.

The control law proposed in [83] is only a general framework to support the end to end solutions. From the framework to real protocol, there are challenging issues to be addressed. Firstly, how to select the utility function for different types of services? Do the various types of services use different utility functions or the same utility function? Secondly, how to ensure the QoS-aware control protocol friendly with TCP? Thirdly, how to accurately feed back the congestion indicator? Finally, implementation of the protocol and verified in the real Internet ia also important. The proposed TERSE methodology solved all the above

concerns. It uses TCP utility function for all the types of services. TCP is considered as a type of service, thus making the QoS aware service flows friendly with TCP.

The major difference for the TERSE methodology compared to other utility-based optimization methods is that the QoS requirement is explicitly expressed in the flow level constraint instead of encoding it into the utility function, and hence we can use a single utility function for all application classes and differentiate them by tuning different minimum guaranteed rates. For example, the minimum guaranteed rate is set to 0 for NRE flow, and non-zero value for other types of flows. By using the same utility function, the flows in different classes are considered to fairly sharing the network resources except the minimum guaranteed rate. Due to the dominant of TCP used in today's Internet, we use the utility function of TCP to derive the control laws for other application classes so that the derived protocols can be TCP friendly.

3.5 Summary

In this chapter, we proposed a new utility-based protocol design methodology TERSE which can provide a unified end-to-end traffic control protocol that enables the non-real-time services. Unlike the traditional utility-based methodology which employs different utility functions for different services, TERSE only encodes the elastic part of non-real-time service as utility function of TCP and expresses the non-elastic part as boundary conditions in the utility optimization problem.

CHAPTER 4
TCP UTILITY FUNCTION

4.1 Overview

Understanding the TCP congestion control mechanism from a global optimization point of view is not only important in its own right, but also crucial to the design of other transport layer traffic control protocols with provable properties. In this chapter, we derive a global utility function and the corresponding optimal control law, known as TCP control law, which maximizes the global utility. The TCP control law captures the essential behaviors of TCP, including slow start, congestion avoidance, and the binary nature of congestion feedback in TCP. We find that the utility function of TCP is linear in the slow start phase and is proportional to the additive increase rate and approaches the well-known logarithm function as the data rate becomes large in the congestion avoidance phase. We also find that the slow start phase with a fixed threshold is critical to the bandwidth allocation between the flows in network.

The rest of the chapter is organized as follows: Section 3.2 derives the utility function of TCP and TCP control law. The proposed TCP model is tested against NS-2 simulation results in Section 3.3. Section 3.4 concludes the chapter.

4.2 Utility Function of TCP

In this section, we derive the global utility function of TCP based on a large family of distributed control laws proposed in [83]. As we showed in Chapter 2, with any given concave

user utility functions, a family of distributed control laws with different positive $z(x,t)$ functions can be readily derived from Eqns. (3.1)-(3.3), which will drive the network to an operational point where the sum of the utility functions is maximized. On the other hand, an empirically designed control protocol, such as the TCP congestion control protocol, can be *reversely engineered* by matching its behavior with the above family of control laws. This matching process, if successful, will lead to an "optimal control law" that best describes TCP behavior, and hence, to the discovery of its underlying utility function. In this section, we demonstrate how the utility function that underlies the TCP behavior can be identified through such a reverse engineering procedure.

Since the TCP congestion control mechanism is designed empirically with many detailed fine tunings, it is unlikely that there exists a utility based control law that can perfectly match all the details of TCP behaviors. In this chapter, we attempt to match two major TCP behaviors, i.e., the MIMD behavior in the slow start phase and the AIMD behavior in the congestion avoidance phase. The TCP behavior due to timeout (i.e., reducing the window size to one) is not matched. In other words, we assume that any congestion indications, such as timeout and three repetitive ACKs, have the same effect on the TCP behavior, i.e., causing the reduction of the window size by half.

Assume that the increase rate is αx ($\alpha >$0) and the decrease rate is βx ($0 < \beta \leq 1$) in the slow start phase. In the absence of congestion, i.e., $cg = 0$, the control law in Eqn. (3.1) is given by

$$\dot{x} = z(t,x)f(x) = \alpha x \tag{4.1}$$

In the presence of congestion, i.e., $cg = 1$, we have

$$\dot{x} = z(t,x)\left[f(x) - 1\right] = -\beta x \tag{4.2}$$

Then we have the utility function

$$U(x) = xlog\left(1 + \frac{\alpha}{\beta}\right) \tag{4.3}$$

and

$$z(t, x) = \frac{\alpha x}{f(x)} = (\alpha + \beta)x \tag{4.4}$$

Since $U(x)$ is a differentiable, concave, and strictly increasing function, and $z(t, x)$ is positive for $x > 0$, the MIMD control law can achieve globally optimal rate allocation. The coefficient of the utility depends on the ratio α/β. A flow with a larger increase factor (α) or smaller decrease factor (β) leads to higher utility. For the slow start phase in TCP, assume α=1 and β=1/2, then $U(x) = xlog3$.

For the congestion avoidance phase, assume the additive-increase rate is $\mu > 0$, and multiplicative-decrease rate is βx. In the absence of congestion, i.e., $cg = 0$, the control law is given by

$$\dot{x} = z(t, x)f(x) = \mu \tag{4.5}$$

In the presence of congestion, i.e., $cg = 1$, we have

$$\dot{x} = z(t, x)[f(x) - 1] = -\beta x \tag{4.6}$$

Then we have

$$U(x) = \left(\frac{\mu}{\beta} + x\right)[log(\mu + \beta x) - 1] - x[log(\beta x) - 1] + C \tag{4.7}$$

and

$$z(t, x) = \mu + \beta x \tag{4.8}$$

where C is a constant; $U(x)$ is a differentiable, concave, and increasing function; and $z(t,x)$ is positive. Hence the AIMD control law achieves globally optimal rate allocation maximizing the utility function described in Eqn. (4.7).

Now, consider the steady state where the slow start phase ends at a constant rate x_s. We set C at a value such that $U(x)$ is continuous at this point. Then we have

$$C = x_s log\left[\frac{(\alpha+\beta)x_s}{\mu+\beta x_s}\right] - \frac{\mu}{\beta} log(\mu+\beta x_s) + \frac{\mu}{\beta} \tag{4.9}$$

The proposed utility function $U(x)$ in Eqns. (4.3) and (4.7) is obtained by matching the TCP control behaviour with the control law proposed in [83]. In here, we model TCP behavior not only based on the AIMD of congestion avoidance phase as other researchers did, but also on the MIMD of slow start phase of TCP. The utility function $U(x)$ is a logarithm form in the congesiton avoidance phase , e.g., Eqn. (4.7). And it is a linear form in the slow start phase, e.g., Eqn. (4.3).

In the TCP congestion avoidance phase, $\beta = 1/2$. Note that the β value is found to be larger than 1/2 in [61] and [48]. However, our analysis in the next section shows that β value has limited effect on the equilibrium rate allocation and hence, we simply let $\beta = 1/2$.

By letting $\beta = 1/2$, we can obtain

$$U(x) = \begin{cases} xlog(3) & \text{if } x < x_s \\ (2\mu+x)\left[log\left(\mu+\frac{x}{2}\right)-1\right] - x\left[log\left(\frac{x}{2}\right)-1\right] + C & \text{if } x \geq x_s \end{cases} \tag{4.10}$$

and

$$C = x_s \cdot log\left(\frac{3x_s}{2\mu+x_s}\right) - 2\mu \cdot log\left(\mu+\frac{x_s}{2}\right) + 2\mu \tag{4.11}$$

here x_s is the slow start threshold and

$$z(t,x) = z_{be}(t,x) = \begin{cases} \frac{3x}{2} & \text{if } x < x_s \\ \mu+\frac{x}{2} & \text{if } x \geq x_s \end{cases} \tag{4.12}$$

In this thesis, we set $\alpha = 1$ and $\beta = 1/2$. This setting is the same as the congestion control law of the most prevalent TCP Reno.

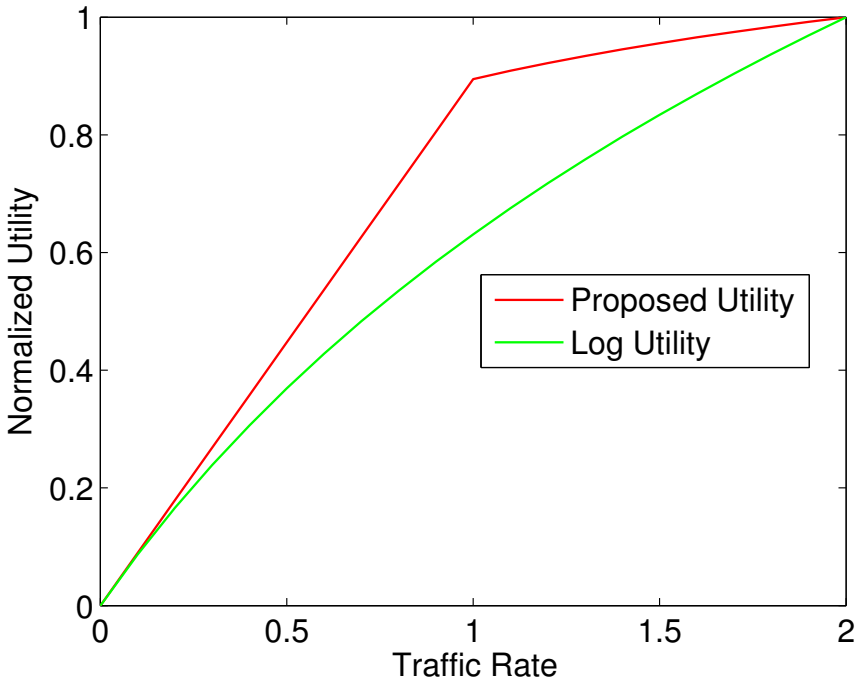

Figure 4.1. Utility Functions

The proposed, normalized utility function with $\mu = 0.1$ and $x_s = 1$, as well as the logarithm utility function ($log(1+x)$) are presented in Figure 4.1. We see that the utility in the slow start phase constitutes a significant part of the proposed utility function. A flow achieves more utility in the slow start phase than that in the congestion avoidance phase. Hence, flows with larger slow start thresholds tend to grab the network resource more aggressively.

However, when TCP timeout is a rare event, the slow start phase plays an important role only during the initial TCP ramp-up phase. This is because TCP uses a dynamic slow start threshold. A flow always adjusts its slow start threshold to be half of the peak rate (the rate at the congestion time). When TCP is in the steady state, the peak rate is fixed and so is the slow start threshold. But note that different flows may have different slow start thresholds. By ignoring the timeout effect, whenever a congestion occurs, the rate is set to

the slow start threshold and the flow immediately enters the congestion avoidance phase. In this case, the slow start phase does not play a role.

4.2.1 Analysis

To the best of our knowledge, the TCP utility function derived in the previous section has been by far the only utility function that can capture both MIMD and AIMD behaviors of TCP. In this section, we provide detailed analysis of this utility function aiming at a better understanding of the end-to-end TCP congestion control mechanism and for the benefit of the proposed protocol design.

Dynamic Utility: We first note that in the above derivation of the TCP utility function, we implicitly assumed that the slow start threshold x_s is a constant. This is a steady state solution that holds only when the traffic load and pattern in the network stay unchanged. However, when the traffic load and pattern changes, x_s will change. Besides, x_s is also a function of RTT of a TCP session. This means the TCP utility function will change its value range and shape as traffic load and pattern changes, and also changes with RTT. Figure 4.2 demonstrates this by plotting the TCP utility function at various x_s values. As one sees from these plots that the TCP utility function is indeed concave and resemble the NRE utility function in Figure 2.1.

Slow Start Phase: The result in Eqn. (4.3) shows that the underlying utility function for the slow start phase is a linear function with a system-parameter-dependent coefficient. It is well-known that the user utility of a linear form can easily lead to unfair share of the link bandwidth. In fact, it can be easily shown that when multiple flows sharing a single-hop link, the one with the largest coefficient will take over all the link bandwidth, starving all the rest of the flows.

The above observations explain why the slow start threshold in TCP must be dynam-

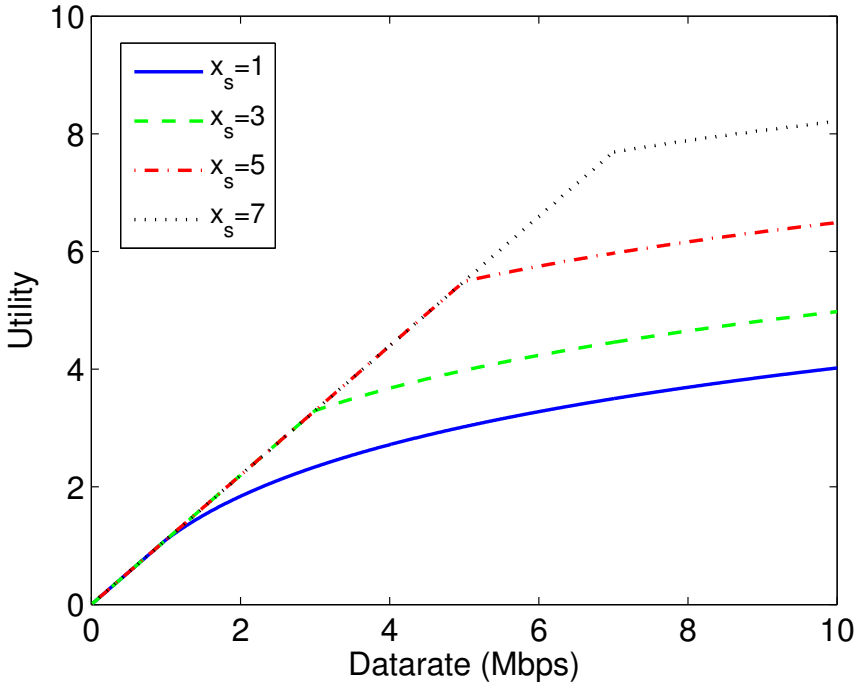

Figure 4.2. Utility values with different slow start threshold

ically adjusted to allow fair share of the network resources. In TCP, a flow always adjusts its slow start threshold to be half of the peak rate (the rate when congestion starts). When TCP is in the steady state, the peak rate is fixed and so is the slow start threshold. As we shall see shortly, the bandwidth allocation in the congestion avoidance phase is roughly inversely proportional to RTT and does not cause strong bias in favor of a particular flow. This ensures that no flow will be able to grab the entire link bandwidth, starving other flows.

Congestion Avoidance Phase: The utility function for the congestion avoidance phase is given in Eqn. (4.7). First, we note that when $x >> \mu/\beta$, $U(x) \simeq (\mu/\beta)log(x) + const$, reminiscent of the logarithm utility that leads to the well known proportional fairness in [56]. However, the current result improves the one in [56] in the following sense. The utility function given in Eqn. (4.7) is explicitly proportional to the additive increase rate μ, whereas the one given in [56] is not dependent on μ. Since for the TCP window-based congestion control, the congestion window size is increased by one packet every RTT in the congestion avoidance phase, the additive increase rate μ is inversely proportional to RTT. In other words, the current utility function is inversely proportional to RTT, whereas the one given in [56] is independent of RTT. The implication of this improvement is significant in terms of steady state rate allocation. In what follows, we use an example similar to the one in [19] to elaborate more on this.

In [19], Chiu showed that with the $log(x)$ utility function in [56], a TCP flow 0 traversing n-hops sharing the link bandwidth with some single-hop flows, one per link, receives a rate allocation ratio $x_0/x = 1/n$, where x_0 and x are the rates allocated to flow 0 and each single-hop flow (note that the rates allocated to different single-hop flows are the same), respectively. This implies that the proportional fairness derived from the $log(x)$ utility function in [56] does lead to rate allocations reflecting the TCP behavior, i.e., the flow traversing more number of hops tends to receive less resource. However, Chiu went on to show that this is true only when all the links are bottleneck links for flow 0. In the case where

there is only one single-hop flow, e.g., at the first link, flow $x_0/x = 1$, independent of RTT, i.e., achieving max-min rate allocation. This is in contradiction with the intuition that a TCP flow with a larger RTT tends to achieve smaller rate allocation. As a result, Chiu concludes that TCP does not implement proportional fairness.

Now, what we are interested in is whether the TCP utility function given in Eqn. (4.7) can lead to a rate allocation that captures the dependence of TCP rate allocation on RTT. To this end, let us calculate x_0/x using the TCP utility function given in Eqn. (4.7). First, we consider the case where all the links are bottleneck links for flow 0. To be more general, we allow m single-hop flows running on each link. The flow rate x for each single-hop flow can be expressed in terms of x_0 as $x = (B - x_0)/m$, where B is the link bandwidth for all the links, and $\mu_i = \mu$ (i=1, 2, ..., m). Then at the maximum utility, we have $\partial U(\mathbf{x})/\partial x_0$=0, i.e., the flow value of x_0 can be given by

$$\left[\frac{\mu/\beta + (B - x_0)/m}{(B - x_0)/m}\right]^n = \frac{\mu_0/\beta + x_0}{x_0} \tag{4.13}$$

When $x = (B - x_0)/m >> \mu/\beta$, or equivalently, when the TCP utility function can be approximated by $U(x) \simeq (\mu/\beta)log(x) + const$, Eqn. (4.16) can be approximated by

$$\frac{x_0}{x} = \frac{x_0}{(B - x_0)/m} \simeq \frac{\mu_0}{n\mu} \tag{4.14}$$

What significant about this result is that the relative rate allocation for flow 0 here is not only dependent on n but also proportional to the ratio μ_0/μ, or equivalently, inversely proportional to the ratio RTT_0/RTT, in the case of window-based flow control, where RTT_0 and RTT are round-trip times for flow 0 and a single-hop flow, respectively. We also note that this result is independent of β. For that reason, we have simply set $\beta = 1/2$.

It can be easily shown that in the case where single-hop flows are only placed on

one link, e.g., the first one, the relative rate allocation for flow 0 (without approximation) is then equal to μ_0/μ, or reversely proportional to RTT_0/RTT, in agreement with the intuition about the TCP rate allocation. This also explains why the utility function constructed in [118] has to be inversely proportional to RTT.

The above results clearly demonstrate that the utility function derived in the previous section can capture the essential aspects of the TCP behaviors qualitatively. We also note that the dependence of the utility on RTT is derived from the dependence of the utility on μ as a special case, i.e., in the context of the TCP window-based flow control. If the additive-increase μ rate is set to a constant value for all the flows, regardless of their respective RTT values, then all the flows have the same utility function and hence achieves proportional fairness as described in [56]. This explains why keeping a constant additive-increase rate in TCP leads to equal allocation of traffic rate to every flow under certain conditions, independent of RTT, as observed in [29] and [46]. Hence, our utility can lead to different rate allocations, depending on how μ is implemented.

The remaining question to be answered is how well the proposed TCP model captures the steady-state behaviors of TCP quantitatively. We test this question in the next section.

4.3 Verification of Utility Function

In this section, we test the accuracy of the utility function by comparing our control law directly against the end-to-end TCP protocol. The behaviours of end-to-end TCP is fully decided by the aperiodic, binary nature feedback of ACKs. It means that the dynamic characteristics of TCP throughput depend on the dynamics of the critic links in the path from the sender to the receiver. In this section, we design the simulation network topologies to make sure that the flows we want to test would pass through several critic links.

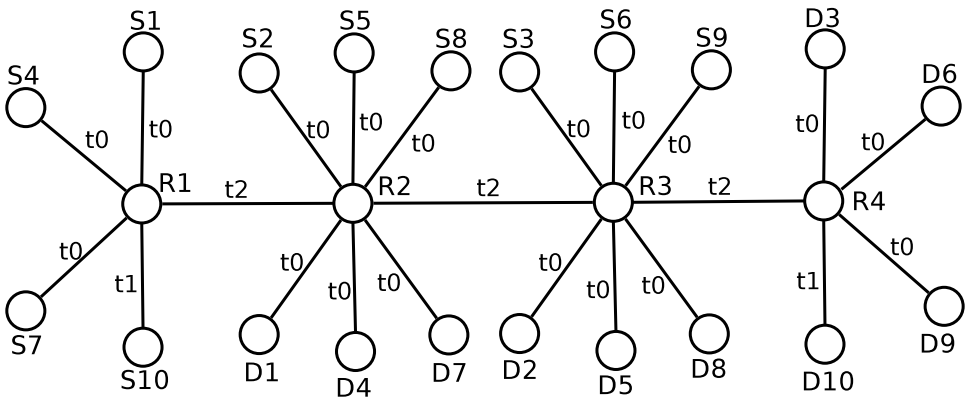

Figure 4.3. Network topology I

4.3.1 Rate allocation based on utility function

The TCP control law and the corresponding utility function in Section 3.2 are derived in the continuous time domain without taking into account the effect of TCP timeout. A natural question to be asked is how accurate the control law and the corresponding utility function describe the TCP behavior. To test the accuracy, we compare our control law directly against the end-to-end TCP protocol. We first compare the measured TCP utility against the optimal utility, and then the measured rate allocation of TCP sessions against the optimal rate allocation. The existing optimization-based TCP control laws as derived in [56] [61] [74] did not compare their control laws directly against TCP because their control laws require significant involvement of network nodes with given AQM mechanisms, which are not end-to-end design.

Let us consider a network with a total number of 24 nodes, as shown in Figure 4.3. Assume there are 10 flows in the network. Flow i that starts at the source node S_i and ends at the destination node D_i (i=1, 2, ..., 10). The link bandwidth for each of the links between nodes R_i and $R_i + 1$ (i=1, 2, and 3) is 100 Mbps (Mega bit per second) and the link bandwidth for each of the rest of links is 200 Mbps. The three 100 Mbps links are the

Table 4.1. Link Propagation Delay of Three Cases

	t_0	t_1	t_2
Case I	5 ms	5 ms	5 ms
Case II	2 ms	5 ms	5 ms
Case III	5 ms	5 ms	2 ms

bottleneck links, each of them is shared by four flows. Three different propagation delays (t_0, t_1 and t_2) are assumed for different links as shown in Figure 4.3. We study three different cases with different link propagation delays as shown in Table 4.1.

We first estimate the utility functions in Eqns. (4.3) and (4.7) for the equilibrium rate allocations of TCP-Reno and compare it with the theoretical optimal utility. In TCP, the threshold of the slow start phase is dynamically adjusted based on the current congestion window size. Whenever congestion is detected, the slow start threshold is set to half of the current window size. In the simulation, we first measure an average threshold value (in Mbps) for each flow which is computed as the average threshold window size (in Mega bits) divided by the average measured RTT (in seconds). Then the theoretical maximum utility is calculated based on the measured average thresholds, and the measured utility of TCP at time t is estimated based on both the measured average thresholds and the average rates from time 0 to t of all the TCP flows. Both theoretical and measured average network utility for Case I are given in Figure 4.4 (similar results are obtained for Cases II and III and they are not given here). The close match of the two curves indicates that the proposed model captures the underlying utility of TCP well.

To further test the accuracy of the proposed TCP model, we compare the measured rate allocations of TCP flows against the theoretical ones that maximize the proposed utility function. As mentioned before, a TCP flow is always in the congestion avoidance phase in the stationary state, assuming that the timeout is ignored. Hence, the utility of each flow can be calculated based on Eqn. (4.7), assuming it is always in the congestion avoidance phase.

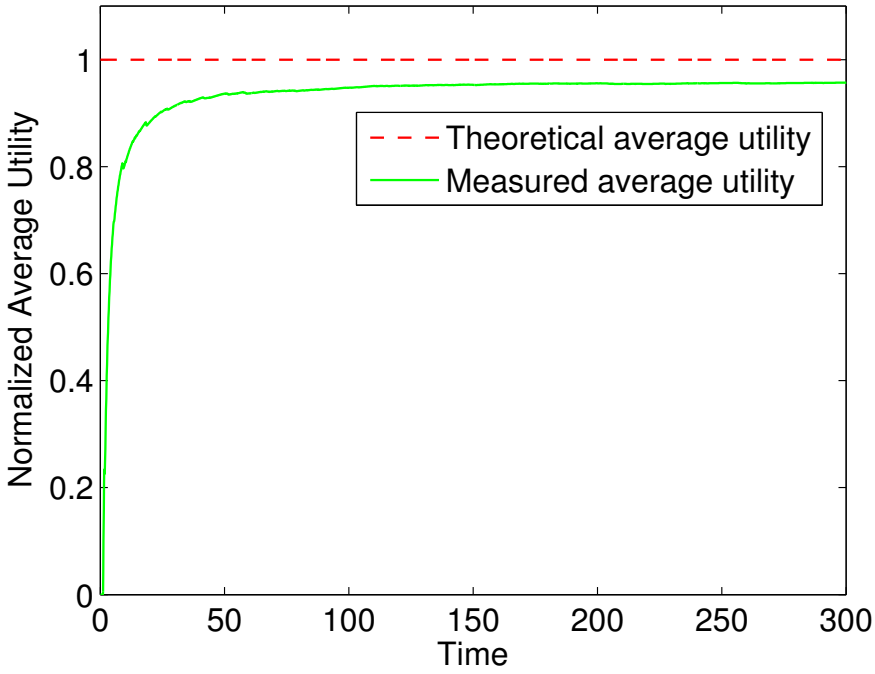

Figure 4.4. Measured average TCP Utility

Denote the additive increase rate and slow start threshold of flow i are μ_i and x_s^i, respectively. Let **x**=($x_1, x_2, ..., x_{10}$). Then the utility of the system is

$$U(\mathbf{x}) = \sum_{i=1}^{10} \left\{ \left(\frac{\mu_i}{\beta} + x_i \right) [log(\mu_i + \beta x_i) - 1] - x_i[log(\beta x_i) - 1] + C_i \right\} \tag{4.15}$$

subject to

$x_i + x_{i+3} + x_{i+6} + x_{10} \leq B$ for i=1, 2 and 3

where B is the shared link bandwidth. Since flows 1-9 have the same RTT, the flow rates and the additive increase rates of these flows are the same, i.e., $x_i = x = (B - x_{10})/3$ and $\mu_i = \mu$ (i=1, 2, ..., 9). Then the maximum utility is achieved by setting $\partial U(\mathbf{x})/\partial x_{10}$=0, i.e., the flow value of x_{10} can be given by

$$\left[\frac{\mu/\beta + (B - x_{10})/3}{(B - x_{10})/3}\right]^3 = \frac{\mu_{10}/\beta + x_{10}}{x_{10}} \tag{4.16}$$

In Eqn. (4.16), the power coefficient takes value 3 because flow 10 traverses 3 bottleneck links. In general, the power coefficient will be n if flow 10 traverses n bottleneck links. When $x = (B - x_{10})/3 >> \mu/\beta$ (in all three cases of the simulation, $\beta x/\mu > 20$), Eqn. (4.16) can be approximated by

$$\frac{x_{10}}{x} = \frac{x_{10}}{(B - x_{10})/3} \simeq \frac{\mu_{10}}{3\mu} \tag{4.17}$$

The flow allocation is dependent on the additive increase rate and the number of bottleneck links, but is independent of β and B. For that reason, we have simply set $\beta = 1/2$. The flow rate is proportional to the additive increase rate and inversely proportional to the number of shared bottleneck links. Hence TCP achieves proportional fairness: inversely proportional to the RTT and the number of shared bottleneck links. When $\mu = \mu_{10}$, the flow ratio $x_{10} : x \simeq 1 : 3$, the same as the optimal rate allocation for the logarithm utility function with a constant scale factor.

In the congestion avoidance phase, the congestion window is increased by 1 packet per RTT. If the packet size is w bits, the additive increase rate of a flow is w/RTT. In our simulation, the packet size is set to 1,000 bytes, i.e., $w = 8000$. The theoretical rate ratios ($x_{10} : x$) based on Eqn. (4.16) for Case I, II and III are: 0.2, 0.12 and 0.26, respectively. Here the RTTs are simply counted as the double propagation delay from the source to the destination. The RTTs for flow 10 and other 9 flows in all three cases are: 50 ms vs 30 ms in Case I; 50 ms vs 18 ms in Case II; and 38 ms vs 30 ms in Case III.

Although not tightly coupled with the queuing mechanisms in use in network nodes, the family of control laws presented in the previous section does implicitly assume that different flows sharing the same congested link should have equal opportunity to sense the

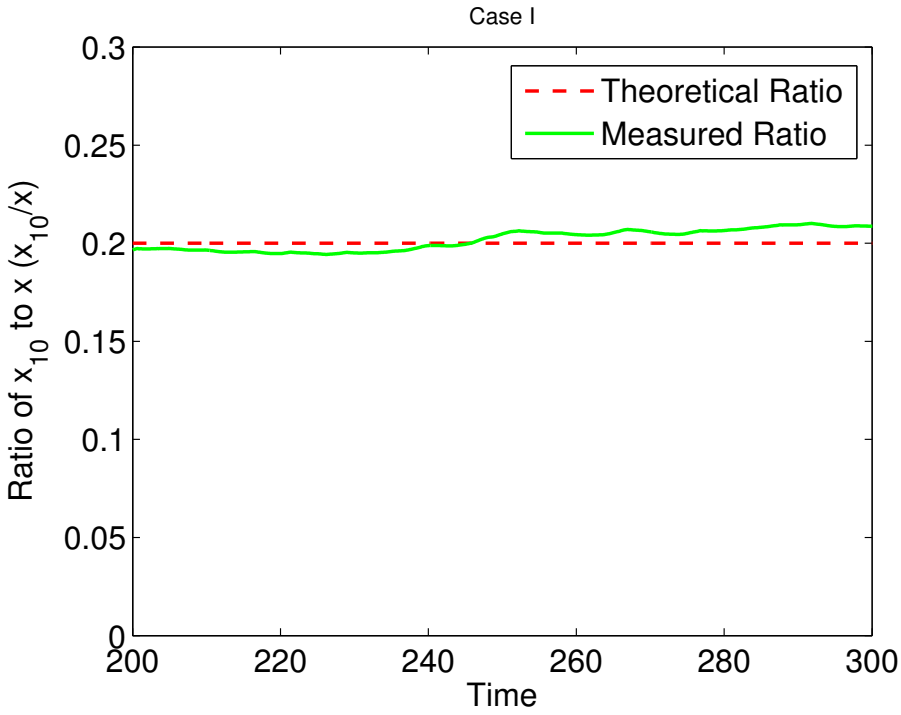

Figure 4.5. Rate Ratio, Case I

congestion. Hence we let all the flows share a single Random Early Detection (RED) first-in-first-out (FIFO) queue at each router. Flows are generated by TCP-Reno. Each simulation run is 300 seconds and the statistics are collected during the last 100 seconds.

Figures 4.5 - 4.7 show the flow ratio versus time in three cases, respectively. The flow rate x in the three cases is computed as the average rate of flows 1-9. Here, we observe that the measured TCP flow ratios are close to the theoretical ratios for all three cases. These results demonstrate that the derived utility function and hence the TCP control law captures the essential behavior of TCP.

These results verify that the utility function of TCP in congestion avoidance is dependent on its additive increase rate. In other words, the utility function of TCP in congestion avoidance phase is delay sensitive. [91] also studied delay sensitive utility functions for

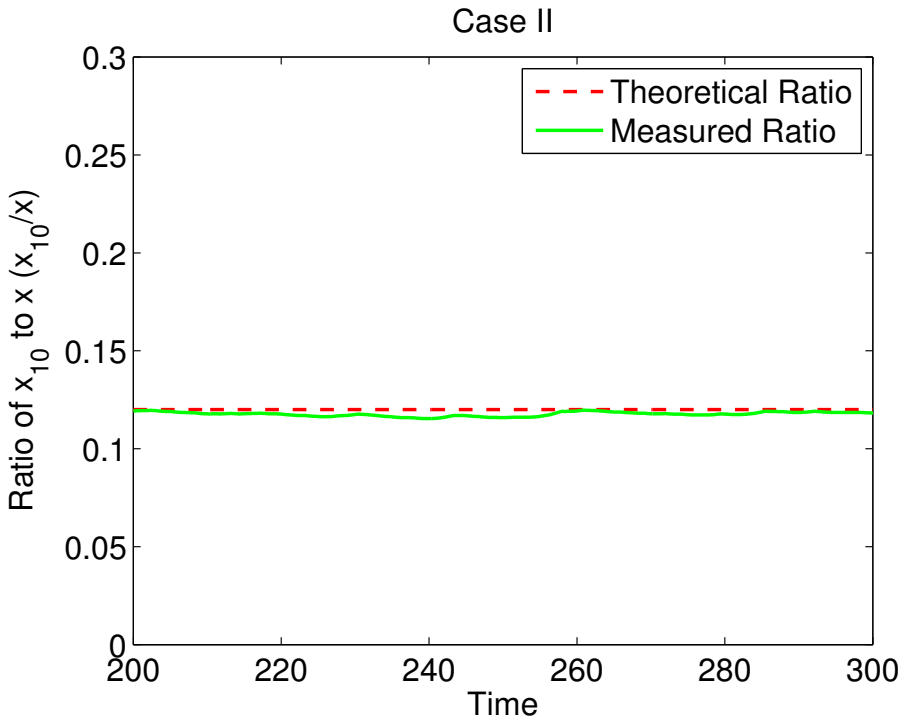

Figure 4.6. Rate ratio, Case II

TCP/IP-like networks.

4.3.2 Effect of Slow Start Threshold

For TCP using dynamical slow start threshold, the slow start phase has no impact on the equilibrium rate allocation among flows, given that the TCP timeout is a rare event. To test the linear utility function for the slow start phase, we modify TCP-Reno by setting a fixed threshold value x_s for every flow in the network. Then we run this modified TCP-Reno on the same network topology with propagation delay given in Case I. We consider $x_s > B/4$ and $x_s < B/4$, separately.

For fixed x_s and $x_s > B/4$, due to the linear utility function in the slow start phase, the theoretical optimal rate allocations are approximately: $x_i \simeq x_s$ (i=1, 2, ..., 9) and $x_{10} \simeq$

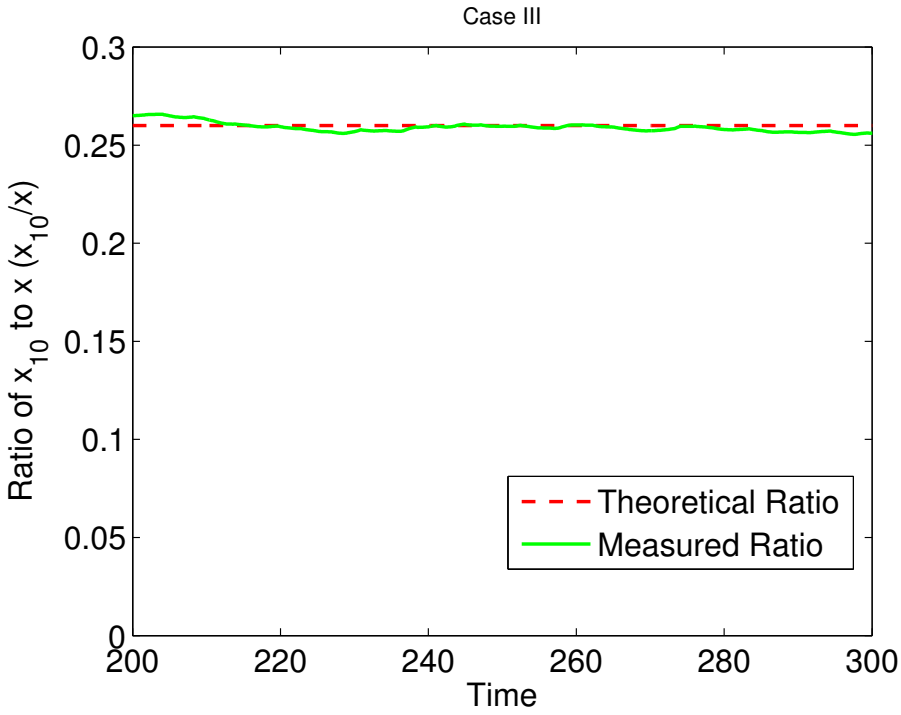

Figure 4.7. Rate Ratio, Case III

$B - 3x_s$, independent of RTT. For $x_s = 30$ Mbps, $x_{10} \simeq 10$ Mbps, $x_i = x \simeq 30$ Mbps (i=1, 2, ..., 9), i.e., the ratio $x_{10} : x \simeq 1/3$; For $x_s = 40$ Mbps, $x_{10} \simeq 0$, $x_i = x \simeq 33.33$ Mbps (i=1, 2, ..., 9). Figures 4.8 and 4.9 present the simulation results for TCP-Reno with fixed threshold at x_s = 30 and 40 Mbps. The curve "average rate" in the figures represents the average rate of flows 1-9. We see that the rate ratio $x_{10} : x$ is around 0.328 for $x_s = 30$ Mbps, very close to the theoretical ratio. The rate of flow 10 is close to 0 for $x_s = 40$ Mbps. For the linear utility, a flow sharing a single bottleneck link with a flow that traverses multiple bottleneck links tends to grab all the available link bandwidth. These results strongly suggest that the utility of TCP in the slow start phase is a linear function.

Now let's look at the case where x_s is fixed and $x_s < B/4$. Due to the linear part in the utility function, the optimal rate of each flow in the steady state is above the slow

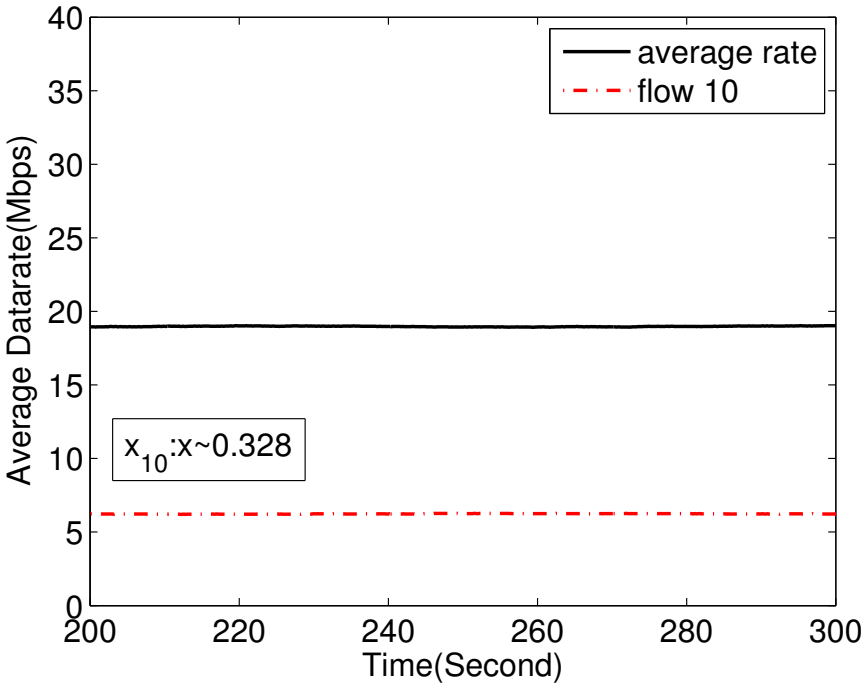

Figure 4.8. Rate allocation, with fixed threshold x_s=30 Mbps

start threshold x_s. Among all the flows, flow 10 has the longest RTT and thus the smallest average and peak rate (x_p^{10}) in the steady state. First consider the case where $x_s < x_p^{10}/2$. When congestion occurs, the rates of all flows are reduced to half of their respective peak rates. The rate of flow 10 is reduced to $x_p^{10}/2$, larger than x_s. By ignoring the timeout effect, when congestion occurs, the flow rates are reduced by half and then all the flows enter the congestion avoidance phase immediately. So the utility function for flows with a fixed small slow start threshold is expected to be the same as that for TCP with a dynamic threshold.

For the case where $x_s < B/4$ and $x_s > x_p^{10}/2$. Now the peak rate of flow 10 (i.e., x_p^{10}) is less than $2x_s$ and greater than x_s (above the threshold) in the steady state. When congestion occurs, flow 10 goes to the slow start phase due to the fact that the rate is reduced to $x_p^{10}/2$ which is below the threshold. In this case, flow 10 does not enter the congestion avoidance

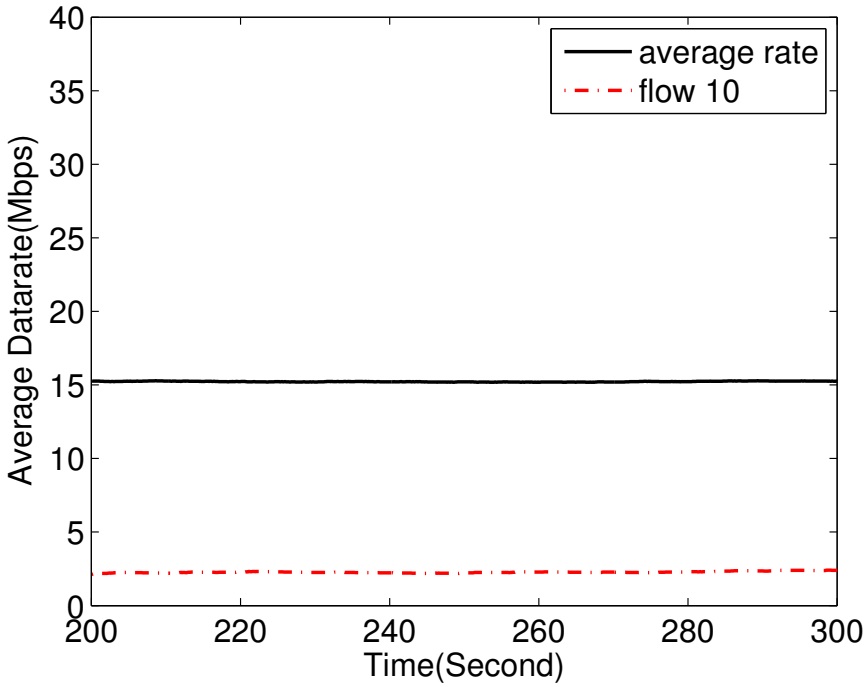

Figure 4.9. Rate allocation, with fixed threshold x_s=40 Mbps

phase immediately after the flow rate is reduced by half. A single utility function in either Eqn. (4.3) or Eqn. (4.7) cannot be applied to compute the optimal rate allocation, because the steady state is a combination of the slow start and congestion avoidance phases. But this by no means suggests that the TCP control law in the congestion avoidance phase does not maximize the utility function given in Eqn. (4.7). It is just that the optimal rate allocation cannot be derived from this utility function. To simplify the computation of the theoretical optimal rate allocation, we use an approximated control law to capture the behavior of the modified TCP. Note that, after flow 10 goes to the slow start phase, it enters the congestion avoidance phase at flow rate x_s after one RTT. If we consider the rate change from the peak value to the slow start threshold to be a single rate decrease at rate $x - x_s$, then the flow can be viewed always in the congestion avoidance phase in the steady state. Then the control law

can be approximately modeled as: (1) the rate increases at rate μ in the absence of congestion (the same as in Eqn. (4.5)), and (2) the rate decreases at $x - x_s$ in the presence of congestion, i.e.,

$$\dot{x} = z(t, x)f(x) = -x + x_s \tag{4.18}$$

Then the utility function derived from Eqns. (4.5) and (4.18) for this approximated control law can be expressed as

$$U^2(x) = (\mu + x - x_s)log(\mu + x - x_s) - (x - x_s)log(x - x_s) \tag{4.19}$$

Now, we study the rate allocation in Case I using fixed x_s at 1, 10, and 15 Mbps. For x_s =1, the peak rates of all the flows in the steady state are above $2x_s$, the utility function of these flows is given by Eqn. (4.7), denoted as $U^1(x)$. For x=10, the peak rates of flows 1-9 in steady state are over $2x_s$, and their utility functions are $U^1(x_i)$ (i=1, 2, ..., 9). For flow 10, the peak rate in steady state is less than $2x_s$ and hence follows the approximated control law, its utility function is given by Eqn. (4.19). In this case, the optimal rate allocation maximizes the total utility $U(\mathbf{x}) = \sum_{i=1}^{9} U^1(x_i) + U^2(x_{10})$. For x_s=15, the peak rates of all ten flows in steady state are less than $2x_s$, and hence all ten flows follow the approximated control law. The optimal theoretical rate ratio of flows in this case is to maximize the total utility $U(\mathbf{x}) = \sum_{i=1}^{10} U^2(x_i)$. The optimal rate ratios ($x_{10} : x$) for slow start threshold x_s at 1, 10 and 15 Mbps are 0.2, 0.44 and 0.63, respectively.

Figures 4.10-4.12 show the simulated rate allocation at x_s = 1, 10 and 15 Mbps, respectively. The rate ratios ($x_{10} : x$) are close to their theoretical counterparts for all three cases. We observe that despite the fact that flow 10 traverses three critical links and has a larger RTT value than the rest of the flows, its rate increases as the threshold increases and stays above the threshold value. This suggests that the fixed slow start threshold could

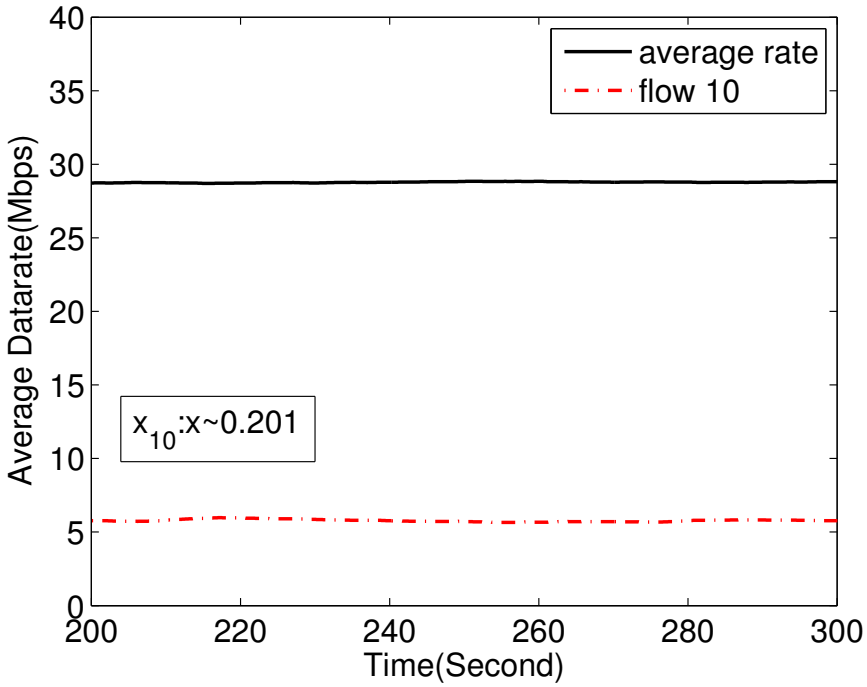

Figure 4.10. Rate Allocation for fixed threshold x_s=1 Mbps

play a key role for providing user fairness, provided that it does not exceed in certain value ($x_s < B/4$ for the current case).

Based on these results, we conclude that the linear part in the utility function plays a key role for rate allocation. Although the slow start phase does not play an important role in TCP (assuming TCP timeout is a rare event) due to its use of a dynamic slow start threshold, understanding the slow start phase provides valuable insights on how to design new end-to-end transport layer protocols, in particular, the protocols that can provide guaranteed rate services. In summary, we verify that the utility functions in the slow start phase is linear.

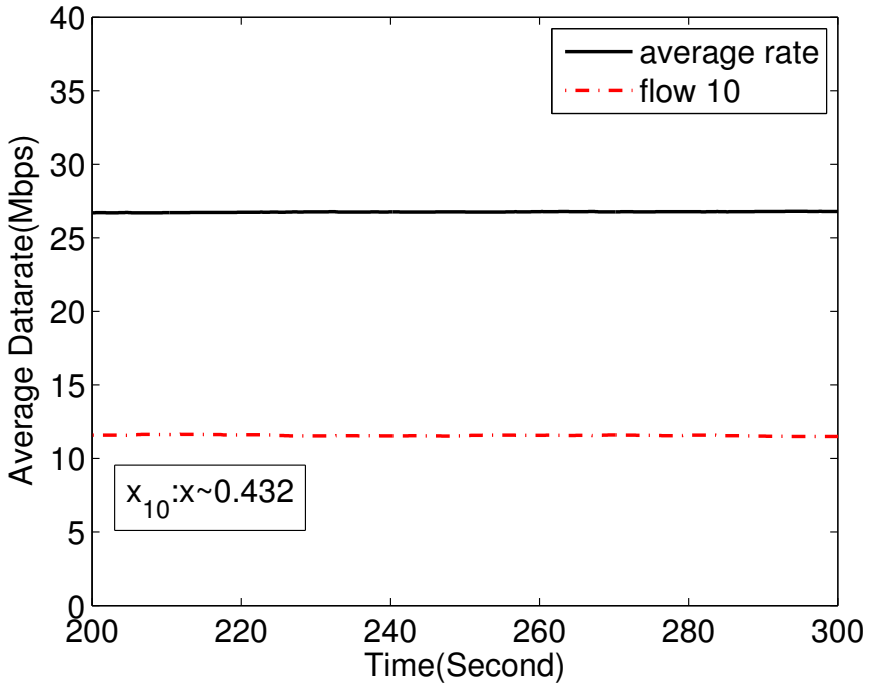

Figure 4.11. Rate Allocation for fixed threshold x_s=10 Mbps

4.4 Summary

The TCP congestion control mechanism used in today's Internet is a fully distributed, end-to-end traffic control mechanism. It uses only resource inferable, single-bit binary information as input for the control, i.e., whether the forwarding path is congested or not. These salient features make TCP highly robust and deployable at global scale.

This chapter aimed at understanding, interpreting, and modeling of end-to-end TCP behavior on the basis of a global optimization perspective. We derived a utility function of TCP and its corresponding control law, called TCP control law. The TCP control law captures the TCP slow start, congestion avoidance, and the aperiodic, binary nature of the TCP information feedbacks. We found that the utility function in the slow start phase is linear. This term indicates that the user fairness in terms of achievable user utilities can be

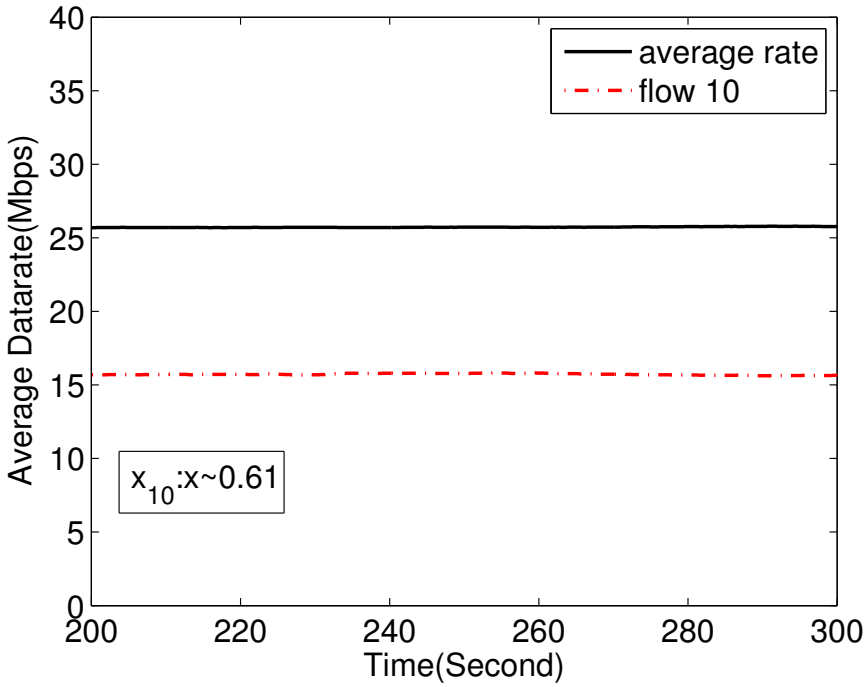

Figure 4.12. Rate Allocation for fixed threshold x_s=15 Mbps

improved by setting the slow start threshold at a fixed, larger values for flows with longer RTT values.

CHAPTER 5
A FAMILY OF QOS AWARE CONGESTION CONTROL LAWS

5.1 Overview

Providing guaranteed end-to-end quality of service (QoS) is important for real-time applications such as Voice-over-IP. In this chapter, a family of optimal, distributed, QoS-aware, end-to-end congestion control laws is derived. It enables a set of class of services (CoSs) including Assured Forwarding Service (AF), Minimum Rate Guaranteed Service (MRG), and Minimum Rate Guaranteed and Upper Bounded Rate Service (MRGUBR). These control laws maximize the same utility function as the TCP congestion control protocol does. As a result, they are by design TCP friendly. The theoretical upper bound of MRG rate is derived and verified through simulations. These control laws are implemented as window-based congestion control protocols, similar to the window-based TCP congestion control protocol. The performance of these control protocols is tested based on NS-2 simulation. The results indicate that these control protocols are indeed TCP friendly and can provide end-to-end service assurance as long as the percentage of network bandwidth consumed by the flows using these control laws is moderately small. Consequently, this family of QoS-aware control protocols has the potential to be used to provide end-to-end QoS guaranteed services for low-bandwidth applications, such as Voice-over-IP. We implement the MRG protocol in Linux operation system. The cross Pacific transmission testing shows that it can achieve more than 1.2Mbps throughput performance, 150% higher than TCP-reno and 50% higher than TCP-cubic.

The rest of the chapter is organized as follows: Section 4.2 describes a family of QoS-

aware traffic control laws. Section 4.3 analyzes the capacity of MRG protocol. In Section 4.4, the proposed QoS-aware control laws are mapped to the unified packet-based control protocols, which are evaluated through NS-2 simulations and real-world Linux implementation testing. Section 4.5 concludes the chapter.

5.2 QoS aware control laws based on the utility function of TCP

In this chapter, the design objective is to develop a set of optimization-based, end-to-end, QoS-aware traffic control protocols, which are TCP friendly by design. We derive a set of QoS-aware control laws by simply substituting the TCP utility function obtained in the previous chapter into Eqs.(3.1) - (3.3). Since all the QoS-aware control laws thus derived share the same utility function with TCP, they are TCP friendly or TCP fair by design.

All these control laws possess the following three desirable features. First, it maximizes the same utility function as the TCP control law does, i.e., the utility function given in Eqns. (4.3) and (4.7). Hence, by design, these control laws are TCP fair and friendly. Second, just like end-to-end TCP, they are end-to-end in the sense that they can all be implemented without involving the network nodes. Third, flows using the these control laws can coexist with the flows using the TCP control law, and they jointly drive the network to a globally stable and optimal state. In what follows, we present the control laws as below.

Based on the TCP utility function, and Eqns. (3.1) - (3.9), we immediately get a family of QoS-aware control laws as follows:

Without congestion, the increasing rate is

$$\dot{x} = \begin{cases} (3r(x) - 1)x/2 & \text{if } x < x_s \\ (r(x) - 1)x/2 + \mu r(x) & \text{otherwise} \end{cases} \tag{5.1}$$

where the $r(x)$ is given in Eqns. (3.5), (3.4), (3.6) and (3.7) for different CoSs. If $r(x) = 1$, the control laws in Eqn. (6.10) degenerate to the TCP control law.

With congestion, the QoS-aware control law acts the same as the TCP control law, i.e., the decreasing rate is $x/2$.

$$\dot{x} = -x/2 \tag{5.2}$$

5.2.1 Assured Forwarding Service

By bringing Eqn. (3.5) into Eqn. (3.1) and assume $x < \Lambda$ (suppose $x_s < \Lambda$), we can obtain the increasing rate of AF as:

$$\dot{x} = \begin{cases} (3r_{max} - 1)x/2 & \text{if } x < x_s \\ (r_{max} - 1)x/2 + \mu r_{max} & x_s < x < \Lambda \\ (r_{min} - 1)x/2 + \mu r_{min} & \text{otherwise} \end{cases} \tag{5.3}$$

here we can see that the increase rate is still exponential even if the rate exceeds the slow start threshold but is smaller than the targeted rate. But when the rate is above the targeted rate, the rate decreases even if no congestion occurs.

5.2.2 Minimum Rate Guaranteed Service

By bringing Eqn. (3.4) into Eqn. (3.1), we can obtian the increase rate of MRG without congestion as,

$$\dot{x} = \begin{cases} (3r^m_{max} - 1)x/2 & \text{if } x < x_s \\ (r^m_{max} - 1)x/2 + \mu r^m_{max} & x_s < x < \theta \\ \mu & \text{otherwise} \end{cases} \tag{5.4}$$

where θ is the targeted minimum rate and r^m_{max} is a tunable parameter. If $r^m_{max} = 1$, the MRG control law degenerates to the TCP control law. If r^m_{max} is greater than 1, the exponential increase rate for MRG control law is faster than TCP in the slow start phase, and the increase rate is still exponential in the congestion avoidance phase, but at a lower rate. In our implementation of the MRG control law, the rate of a flow at time t is calculated as the congestion window size at time t divided by the average measured RTT at time t. We also use a dynamic slow start threshold. In other words, when the path is congested, the slow start rate is set to half of the current flow rate and the rate is reduced by half. However, the increase rate is

always exponential if the rate is smaller than the target rate. Hence the target rate plays a role similar to the fixed threshold and hence it may provide rate guarantee.

From Eqn. (5.4), we note that if $r^m_{max} = 3$, the increase rate is $x+3\mu$, i.e., the increase rate is the current flow rate plus 3μ in each RTT. If the rate is reduced by half from the target rate when congestion occurs, such increase rate makes the flow rate back to the targeted rate in one RTT. A larger r^m_{max} results in a larger increase rate. However, a larger r^m_{max} still leads the flow rate back to the targeted rate in one RTT, same as in case of $r^m_{max} = 3$. Hence any $r^m_{max} > 3$ has the similar effect as $r^m_{max} = 3$ in the most cases. $r^m_{max} > 3$ has impacts only for the case that the a flow senses continuous congestions and goes back to initial slow start rate. Hence we set r^m_{max} to a value between 1 and 3.

5.2.3 Upper Bounded Rate Service

By bringing Eqn. (3.6) into Eqn. (3.1), we can obtian the increase rate of UBR without congestion as,

$$\dot{x} = \begin{cases} x & \text{if } x < x_s \\ \mu & \text{if } x_s < x < \Theta \\ (r^M_{min} - 1)x/2 + \mu r^M_{min} & \text{otherwise} \end{cases} \tag{5.5}$$

where Θ is the upper bounded rate and r^M_{min} is a tunable parameter. In Eqn. (5.5), we assume $\Theta > x_s$. If $\Theta <= x_s$, the increase rate of UBR without congestion is as,

$$\dot{x} = \begin{cases} x & \text{if } x < \Theta \\ (3r^M_{min} - 1)x/2 + \mu r^M_{min} & \text{if } \Theta < x < x_s \\ (r^M_{min} - 1)x/2 + \mu r^M_{min} & \text{otherwise} \end{cases} \tag{5.6}$$

5.2.4 Minimum Rate Guarantee and an Upper Bounded Rate Service

By bringing Eqn. (3.7) into Eqn. (3.1) and assuming $x_s < \theta < \Theta$, we can obtian the increase rate of MRGUBR without congestion as,

$$\dot{x} = \begin{cases} (3r^m_{max} - 1)x/2 & \text{if } x < x_s \\ (r^m_{max} - 1)x/2 + \mu r^m_{max} & \text{if } x_s < x < \theta \\ \mu & \text{if } \theta < x < \Theta \\ (r^M_{min} - 1)x/2 + \mu r^M_{min} & \text{otherwise} \end{cases} \tag{5.7}$$

where θ is the targeted minimum rate, Θ is the upper bounded rate, r^m_{max} and r^M_{min} are two tunable parameters. It is easy to obtain other congestion control law for situation other than $x_s < \theta < \Theta$.

5.3 Theoretical Maximum Achievable Guaranteed Minimum Rate

The control laws derived in the previous section allow the sending rates to fall below their minimum guaranteed rates. This raises the concern whether, on average, the minimum guaranteed rate for a real-time application using this control law can be achieved or not. In this section, we derive a theoretic maximum achievable guaranteed minimum rate for the above MRG control law based on the fluid model. We consider a network with one bottleneck link as shown in Figure 5.1. Assume N_t TCP flows labeled as $1, \ldots, N_t$ and N_m MRG flow labeled as $N_t + 1, ..., N$ (here $N = N_t + N_m$) sharing a bottleneck link with bandwidth B as shown in Figure 5.1.

In fluid model [80] [109], the window size of TCP flow i can be modeled as:

$$\dot{W}_i(t) = \frac{[1 - p(q(t - R_i(t))]}{R_i(t)} - \frac{W_i(t)W_i(t - R_i(t))}{2R_i(t - R_i(t))} p(q(t - R_i(t))) \tag{5.8}$$

where $W_i(t)$ is the window size of TCP flow i at time t, $R_i(t)$ is the RTT of flow i at time t, $p(q)$ is the packet drop probability of a router at queue length q.

For an MRG flow, when the congestion occurs, its rate is reduced by half, then it will go back to the targeted minimum rate after one RTT if r^m_{max} is over 3. In the case that the

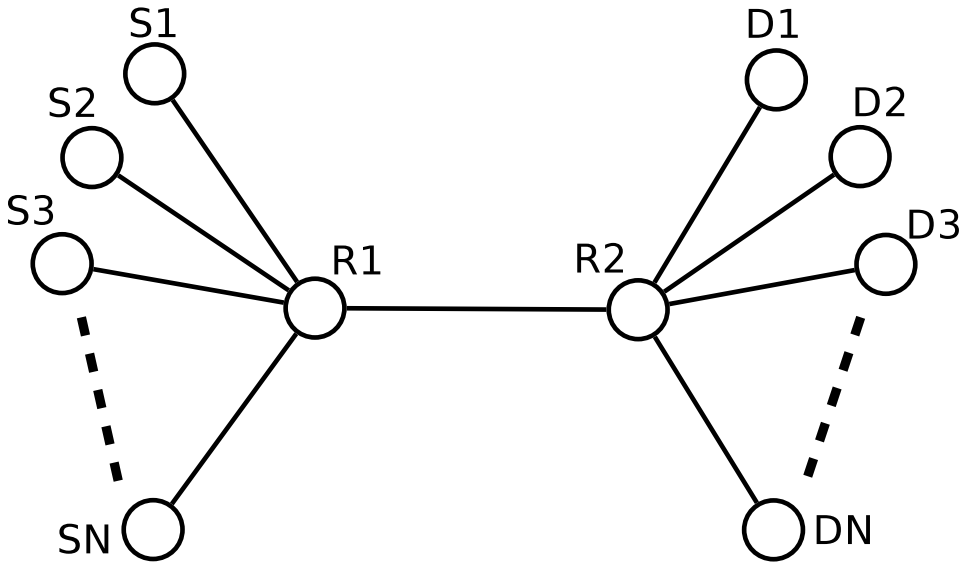

Figure 5.1. Network topology 4.1

target rate is greater than the slow start threshold, the control law can be approximated as the rate dropped from the peak rate to the targeted rate. Hence the window size of MRG flow j with targeted minimum rate θ_j can be approximately modeled as:

$$\begin{aligned}\dot{W}_j(t) \simeq & \frac{[1-p(q(t-R_j(t))]}{R_j(t)} \\ & -\frac{[W_j(t)-\theta_j R_j(t)]W_j(t-R_j(t))}{R_j(t-R_j(t))}p(q(t-R_j(t)))\end{aligned} \tag{5.9}$$

The packet drop probability $p(q)$ of RED takes the form,

$$p(q) = \begin{cases} 0 & 0 \le q < q^{min} \\ \frac{q-q^{min}}{q^{max}-q^{min}}p^{max} & q^{min} \le q \le q^{max} \\ \frac{q-q^{max}}{q^{max}}(1-p^{max})+p^{max} & q^{max} < q \le 2q^{max} \\ 1 & 2q^{max} < q \end{cases} \tag{5.10}$$

The RTT of flow k (k=1,2,..., N) $R_k(t)$ can be expressed as follows,

$$R_k(t) = a_k + \frac{q(t)}{B} \tag{5.11}$$

where a_k is the fixed propagation delay of flow k.

For the queue length $q(t)$,

$$\dot{q}(t) = -B + \sum_{k=1}^{N} \frac{W_k(t)}{R_k(t)} \tag{5.12}$$

By taking the expectation of each side of the Eqns. (6.12), (6.15) and (6.17), and following the approximated method used in [80], we get

$$\begin{aligned} E[\dot{W}_i(t)] =& \frac{1 - p(E[q(t - R_i(t))])}{E[R_i(t)]} \\ & - \frac{E[W_i(t)]E[W_i(t - R_i(t))]}{2E[R_i(t - R_i(t))]} p(E[q(t - R_i(t))]) \end{aligned} \tag{5.13}$$

for i=1, 2, ..., N_t, and

$$\begin{aligned} E[\dot{W}_j(t)] =& \frac{1 - p(E[q(t - R_j(t))]}{E[R_j(t)]} \\ & - \frac{E[W_j(t - R_j(t))]}{E[R_j(t - R_j(t))]} \cdot (E[W_j(t)] - \theta_j E[R_j(t)]) p(E[q(t - R_j(t))]) \end{aligned} \tag{5.14}$$

for $j = N_t + 1$, ..., N, and

$$E[\dot{q}(t)] = -B + \sum_{k=1}^{N} \frac{E[W_k(t)]}{E[R_k(t)]} \tag{5.15}$$

The system stable point can be achieved at $E[\dot{W}_k] = 0$ and $E[\dot{q}] = 0$.

For the N_t TCP flows,

$$\begin{aligned} & E[\dot{W}_i(t)] = 0 \Longrightarrow \\ & \qquad E[W_i(t)]E[W_i(t - R_i(t))] = \frac{2(1 - p(E[q(t - R_i(t))]))}{p(E[q(t - R_i(t))])} \end{aligned} \tag{5.16}$$

In the stable point, we can assume that $E[W_i(t)] = E[W_i(t - R_i(t))]$. That means that the average peak rate of TCP flow i at the stable state should be fixed. So we have

$$E[W_i] = \sqrt{\frac{2(1 - p(E[q]))}{p(E[q])}} \tag{5.17}$$

Similarly, the average peak rate of MRG flow j at the stable state is calculated as

$$E[\dot{W}_j(t)] = 0 \Longrightarrow$$
$$E[W_j]^2 - \theta_j E[R_j]E[W_j] - \frac{(1 - p(E[q]))}{p(E[q])} = 0 \tag{5.18}$$

Then we have

$$E[W_j] = \frac{\theta_j E[R_j] + \sqrt{(\theta_j E[R_j])^2 + \frac{4(1-p(E[q]))}{p(E[q])}}}{2} \tag{5.19}$$

The average queue length at the steady state can be obtained as

$$E[\dot{q}] = 0 \Longrightarrow \sum_{k=1}^{N} \frac{E[W_k]}{a_k + \frac{E[q]}{B}} = B \tag{5.20}$$

The average congestion window size of each flow can be solved through Eqns. (6.22), (6.26) and (6.27)). Then the average peak rate (the rate at the congestion time) of flow k is $E[W_k]/E[R_k]$.

Recall the MRG control law, i.e., the flow increases exponentially when its rate is smaller than its target rate; and increases additively after it reaches the target rate. The number of RTTs (n_j) of MRG flow j with additive increase rate μ_j in its additive increase phase is calculated as

$$n_j = \frac{E[W_j]/E[R_j] - \theta_j}{\mu_j} \tag{5.21}$$

Figure 6.3 shows the rate changing of the MRG flow of a congestion cycle in the average situation. After the MRG flow reaches the target rate θ, it enters the linear increasing phase. With n RTT, it reaches the $E[W_N]/E[R_N]$ and encounters the packet drop accident and then halves the flow rate when sensing the losses a RTT later.

Now let us compute the average flow rate of an MRG flow. Assume MRG flow j senses a congestion when its flow rate is at its average congestion rate $E[W_j]/E[R_j]$ (assume

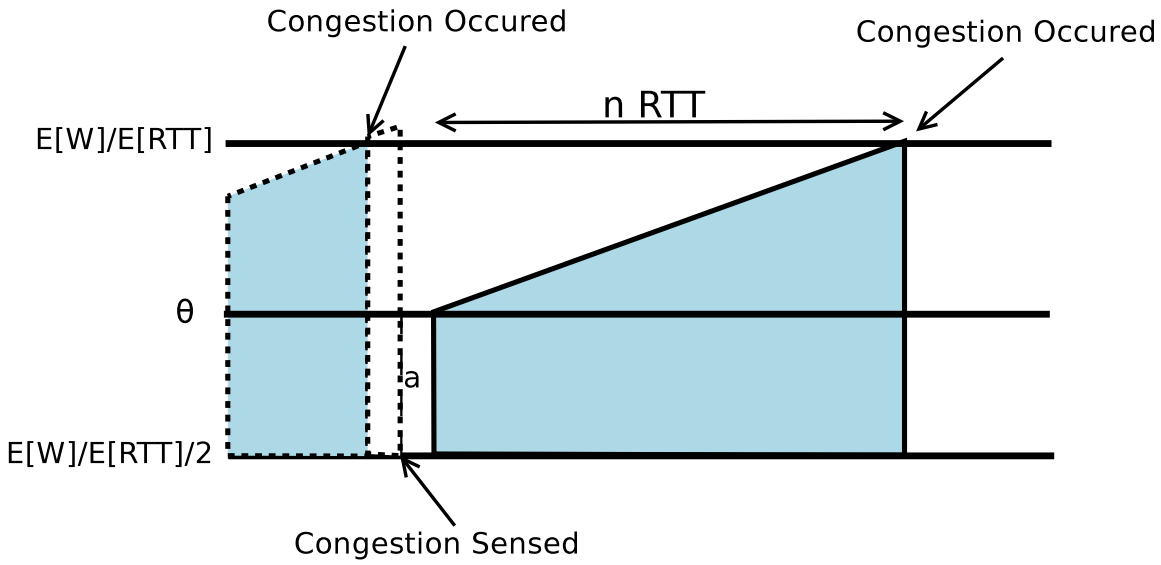

Figure 5.2. MRG flow congestion situation

this rate is above its target rate θ_j). In the next RTT, the flow rate decreases to $E[W_j]/2E[R_j]$, and then increases back to its target rate (θ_j) in the following RTT by assuming r^m_{max} is no less than 3. In the following n_j RTTs, the flow rate increases additively to the congestion rate $E[W_j]/E[R_j]$. After it reaches the congestion rate, the packets in the following RTT may be dropped in the following RTT. We assume the effective flow rate is 0 in that RTT to calculate the theoretic maximum achievable minimum guaranteed rate. So to guarantee the target rate θ_j, n_j has to satisfy the following equation

$$\frac{n_j(n_j+1)}{2} \cdot \mu_j \geq \left(\theta_j - \frac{E[W_j]}{2E[R_j]}\right) + \theta_j \tag{5.22}$$

or

$$\theta_j \leq \frac{E[W_j]}{E[R_j]} - \frac{\sqrt{12\mu_j \frac{E[W_j]}{E[R_j]} + 25\mu_j^2} - 5\mu_j}{2} \tag{5.23}$$

Eqn. (5.23) shows the maximum target rate can be guaranteed for MRG flow j under certain network conditions. It means that all the target rates no larger than the value calculated in Eqn. (5.23) can be guaranteed using the MRG control law. It is verified through the following NS-2 simulation study.

In our simulation, we set $N = 11$ in the network. Among these flows, flows 1-10 are TCP flows, and flow 11 is an MRG flow. The bandwidth for the bottleneck link is 100 Mbps. Figure 5.3 show the theoretical minimum guaranteed rate, the actual minimum guaranteed rate in the simulation, and the rate of flow 11 using the TCP control law. Here, RTTs of the TCP flows 1-10 linearly increases from 20 ms to 56 ms, i.e., the RTT of flow 1 is 20, of flow 2 is 24 ms and so on. The results show that the theoretical result in Eqn. (5.23) gives a little bit lower maximum achievable rate for the MRG flow, because we assume all the packets are dropped after congestion. The results also show that the minimum guaranteed rate of MRG is much greater than the average rate of TCP flow.

Now let us examine the rate allocation of link shared by multiple MRG flows and multiple TCP flows. In the simulation, we set 100 flows, the shared bandwidth is 500 Mbps. All flows have the same RTT 60 ms. From Figure 5.4, we know that the average rate is about 4.5 Mbps when all flows are TCP flow. Then we set N_m flows to be MRG flows, and the remaining $100 - N_m$ flows be TCP flows, and change N_m from 10 to 100. We can see that the target rate can be guaranteed varying from about 9 Mbps to 4.5 Mbps which are about 1 Mbps above the theoretical maximum achievable rate. The average rate of TCP flows drops from about 4.5 Mbps to 2.5 Mbps. This shows that the MGR flow can significant improve the average target rate. When all the flows are MRG flows, the average guaranteed rate is the same as TCP flows.

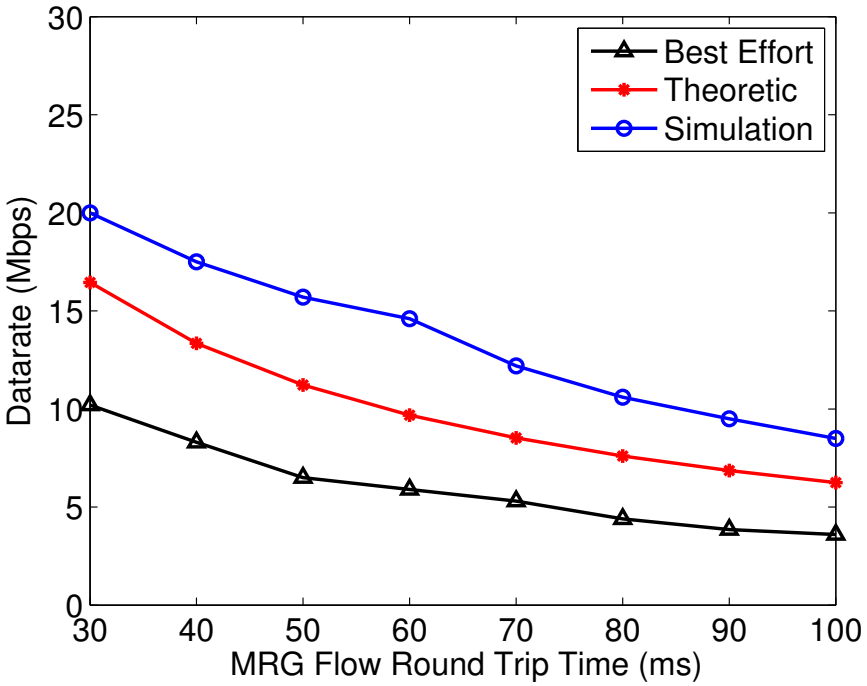

Figure 5.3. The maximum guaranteed target Rate of MRG flow, RTTs of TCP flows vary from 20ms to 56ms

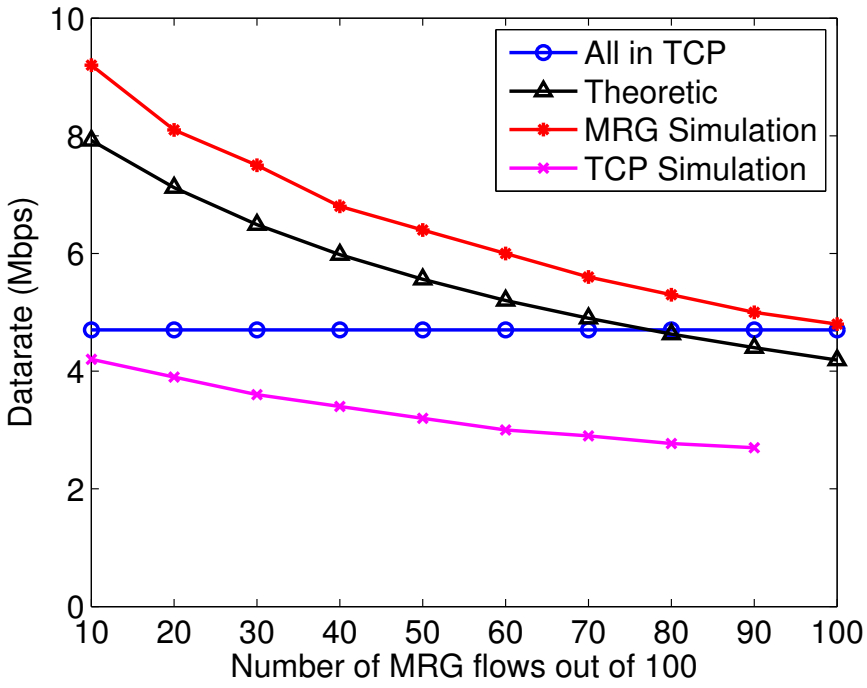

Figure 5.4. The maximum guaranteed target rates of MRG flows

5.4 Performance Evaluation

The family of control laws derived in the previous section is rate based. It is also shown in [83] that the corresponding control laws in discrete time domain with a finite time interval lead to stable and near optimal control. Hence we map the continuous, rate based control laws into the unified end-to-end discrete packet-based protocols. The window-based TCP congestion control has been proven to be a successful approach. Hence, in our protocol design, we use window-based congestion control approach, in line with the TCP congestion control.

To map the rate based control laws into the unified window-based protocols, we need to measure the transmission rate at the sender. A round trip time (RTT) based rate calculation is used. The rate is counted as follows:

$$x(t) = w(t) \cdot s/T(t) \tag{5.24}$$

where $x(t)$ is the transmission rate at time t, $w(t)$ is the sliding window size at the sender at time t, s is the packet size, and $T(t)$ is the average measured RTT at time t. The average measured RTT at time t can be calculated as follows:

$$T(t) = (1 - \eta) \cdot T(t-1) + \eta \cdot MT(t) \tag{5.25}$$

with $0 < \eta \leq 1$, $T(t-1)$ is the latest calculated average RTT before time t, $MT(t)$ is the measured RTT at time t, and η is a tunable parameter, i.e., the relative weight of the historical RTT estimations on the estimation of the current RTT. The sender uses the measured rate as well as the targeted rate to do congestion control.

5.4.1 NS-2 Simulations

We conducted simulations to evaluate the QoS-aware protocols carrying AF, MRG, and MR-GUBR traffic. Let us consider a network with 33 nodes, as shown in Figure 5.5. The performance of TCP and our protocols are mainly depended on the number of bottle neck links,

but little in the network size. In real network, there is usually one or two congested links for a flow. In the experiments setting, we carefully choose the paths of all the flows to make sure that they would pass at least one congested link or more. Hence, the simulation topology catches up most features of the real Internet even if it is small size. In fact, this topology is used for most congestion control paper to verify new protocols. There are 12 flows running in this network. Flow i starts at the source S_i and ends at the destination D_i. The RTTs excluding queuing delay for flows 1 to 12 are 40, 32, 32, 32, 38, 34, 38, 36, 100, 108, 100 and 110 ms, respectively.

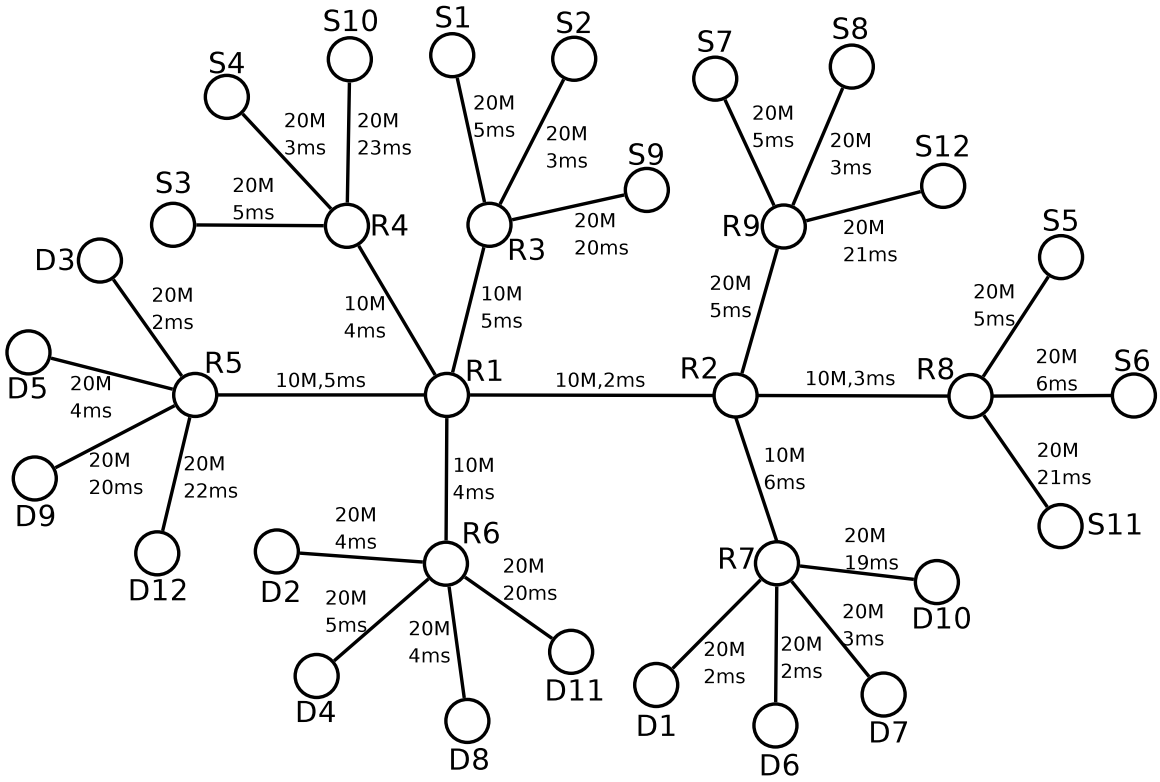

Figure 5.5. Network topology 4.2

In the simulations, we first run all the 12 flows using TCP-Reno. Figure 5.6 shows the average rate of all the 12 flows. The rates of flows 1 to 12 are around 2.6, 3.1, 3.7, 2.9, 2.9, 3.0, 2.9, 2.5, 1.46, 1.1, 1.05 and 1.2 Mbps, respectively.

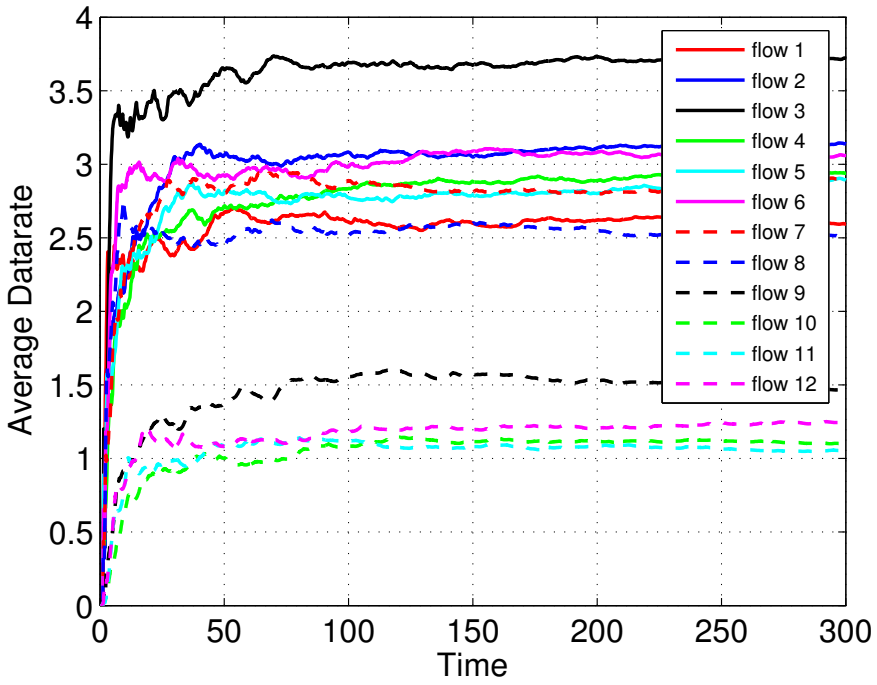

Figure 5.6. Flow Rate Allocation, TCP-Reno for all flows

Now we run the MRG control law for flow 9 with minimum guaranteed rate of 2.5 Mbps and TCP-Reno for other flows. We found that the target average rate 2.5 Mbps can be achieved when the r^m_{max} is 5 or above. It is shown in Figure 5.7 with $r^m_{max} = 5$. Again we run MRG control law for flow 11 with minimum guaranteed rate of 2 Mbps and TCP-Reno for other flows. Figure 5.8 shows the average rates of all the 12 flows by setting $r^m_{max} = 6$ of MRG control law on flow 11.

Then, we run the MRGUBR control law for flow 10 with minimum guaranteed rate of 1.5 Mbps and upper bounded rate of 2.5Mbps. As in the previous simulations, other flows are TCP-Reno. Figure 5.9 indicates that the average rate of flow 10 can achieve 1.75 Mbps with $r^m_{max} = 5$ and $r^M_{min} = 0.75$.

Finally, we run the AF control law for flow 12 with assured rate of 2 Mbps and TCP-

Reno for all other flows. Figure 5.10 shows that the average rate of flow 12 can come close to 2 Mbps with $r_{max} = 7$ and $r_{min} = 0.9$.

The above results indicate that running a single flow using a QoS control law (AF, MRG or MRGUBR) and all other flows using TCP, the TCP flows only suffer minor rate reduction. Now we run flows 9, 10, 11 and 12 with corresponding QoS control laws simultaneously. In this simulation, we set parameters as follows:

Flows 9 and 11: MRG, minimum guaranteed rate 2Mbps, $r^m_{max} = 7$;

Flows 10: MRGUBR, minimum guaranteed rate 2Mbps, upper bounded rate 3Mbps, $r^m_{max} = 7$ and $r^M_{min} = 0.9$;

Flow 12: AF, assured forward rate 1.75Mbps, $r_{max} = 7$ and $r_{min} = 0.9$.

Figure 5.11 depicts the simulation results. One notes that all four QoS flows reach their targeted rate, and in the meantime, the other TCP flows do not suffer too much rate losses. This indicates that the family of QoS aware laws are indeed TCP-friendly.

Now we design the following simulation to test our MRG control law's performance in multimedia streaming. Still considering the topology in Figure (5.5), we assume that the users of flow 9, 10, 11, and 12 want their flows to convey the multimedia stream with the rates of 3.5, 3.5, 3 and 3.5Mbps respectively. Firstly, we run our MRG control law by setting the client-side buffer size as 8 seconds multimedia content. Figure (5.12) shows the client-side buffer changing in 300 seconds by using our MRG control law after the multimedia streams begin to transfer. Because the four client-side buffer sizes are all greater than zero through 300 seconds, it means that our MRG can support the required multimedia streaming with prefetched data of 8 seconds content. And we can find from the figure that the multimedia replay on about 10 seconds later than the transmission begin time. Figure (5.13) shows the client-side buffer changing in the same multimedia streaming by using TCP control law. In

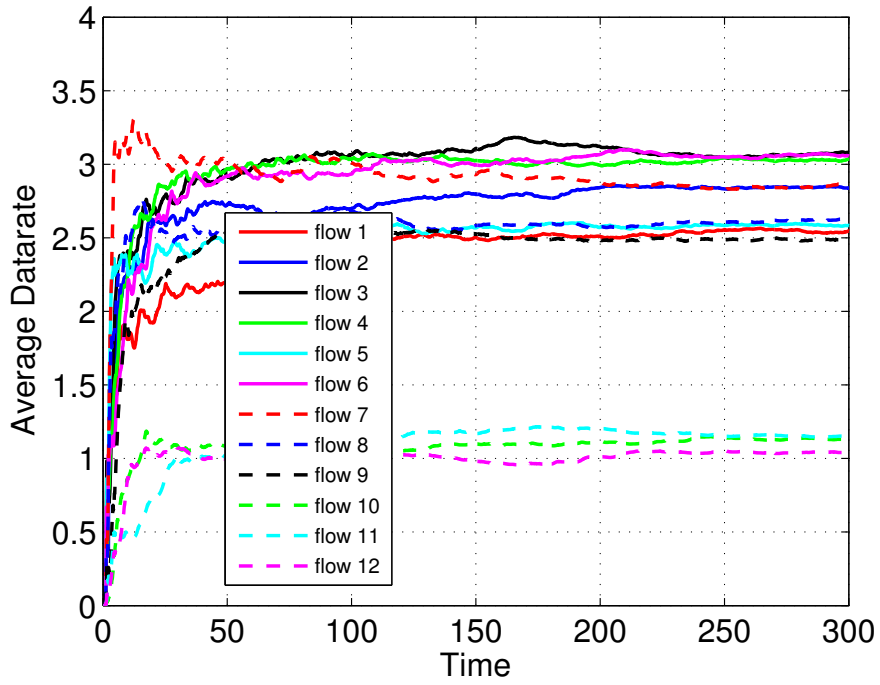

Figure 5.7. Flow Rate Allocation, MRG for flow 9

this time, we set the client-side buffer for 50 seconds multimedia content. In this figure, we can see that the multimedia replay begins on about 90 seconds. And within 300 seconds, the four client-side buffer sizes are all becomes negative (We only set a virtual data consumer on client side). It means that there are no multimedia data to replay on client side and the multimedia streaming applications need to rebuffer the data. In this simulation, our MRG is much better in delivering multimedia stream data than TCP.

5.4.2 Real-World Testing

We implement the MRG protocol in Linux systems and test its performance over the Internet. We use the ftp application in Linux to test the protocol with soft minimum rate guarantee and compare its performance with TCP-reno and TCP Cubic. The sender machine is installed with Ubuntu 8.04 with kernel version 2.6.24 at the Hong Kong Polytechnic University and

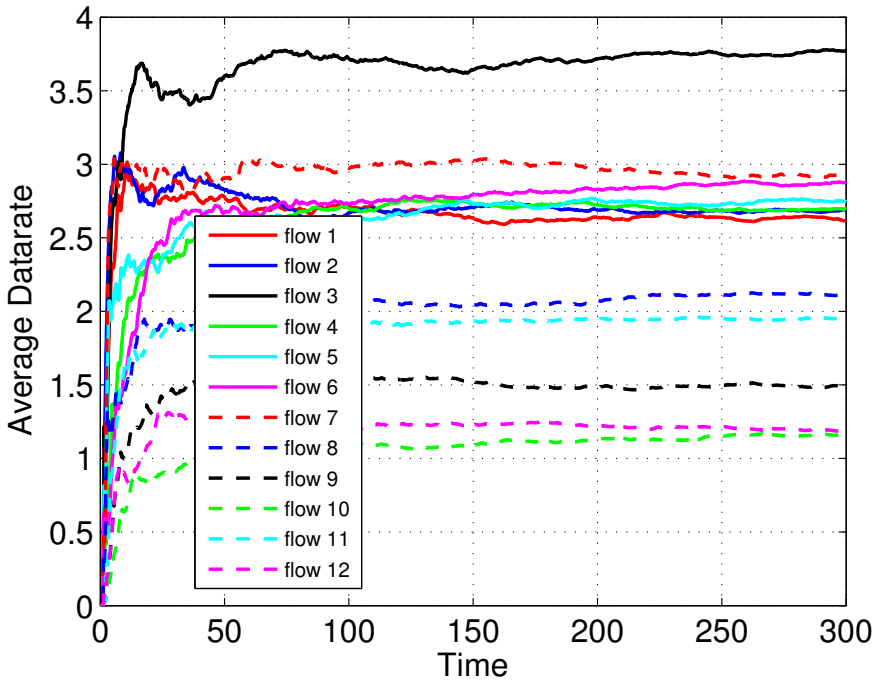

Figure 5.8. Flow Rate Allocation, MRG for flow 11

the receiver is a Linux server at the University of Texas at Arlington in USA.

In this thesis, we focus on designing QoS aware transport layer protocols. To the transport layer protocol, the application layer's data has no difference. So we only use the FTP application layer protocol to test our MRG protocol. In the experiment, a 50MB file is sent from the sender to the receiver. The minimum target rate is set at 1.2Mbps and r_{max} is set at 3. Figure 5.14 compare the transmission rate (the average rates over 10 seconds intervals) of MRG, TCP-reno and TCP Cubic [39]. We can see that the proposed protocol transfers the data with an average rate over 1.2Mbps, while the TCP-reno achieves an average rate around 480 Kbps and the TCP Cubic 800 Kbps. Note that TCP Cubic has been considered to be the fastest variation of TCP. These results demonstrated that the TERSE protocol can provide rate guarantee in the Internet environments.

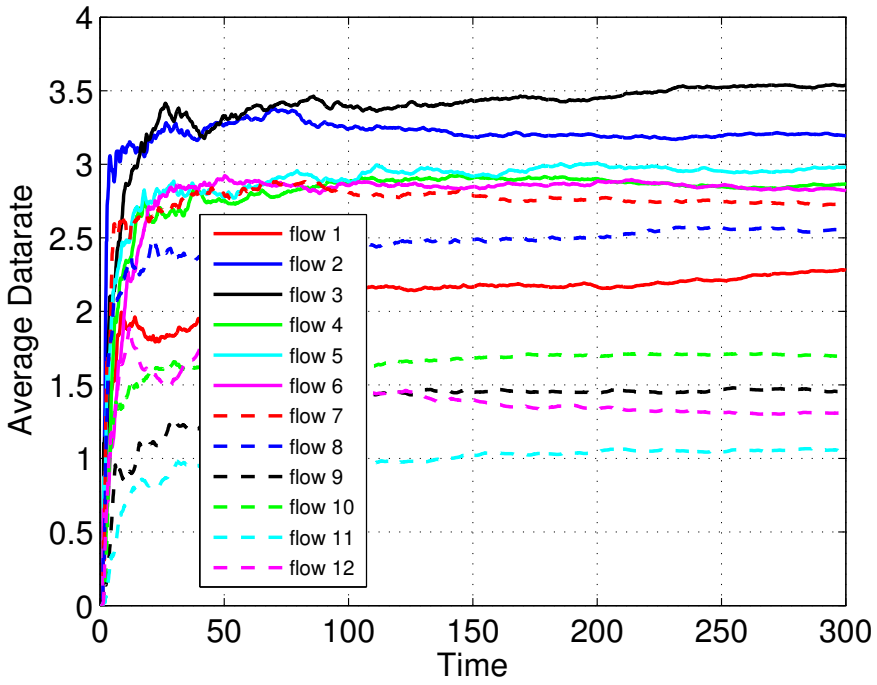

Figure 5.9. Flow Rate Allocation, MRGUBR for flow 10

5.5 Summary

This chapter aimed at designing the unified optimization-based, end-to-end transport layer protocols to support RRA and RDA real-time applications. Based on TERSE design methodology and TCP utility function, we derived a family of rate-based QoS aware congestion control laws for reliable services and mapped them into the unified packet-based protocols. The protocols can support different services, including AF, MRG, UBR, MRGUBR. We also studied the capacity of MRG comparing to TCP protocol by fluid model. The protocols were tested by NS-2. The MRG was implemented in Linux based system and experimented in cross Pacific test. Both the NS-2 simulations and real world test showed that our protocols are the promissing method to support low bandwidth real-time applications.

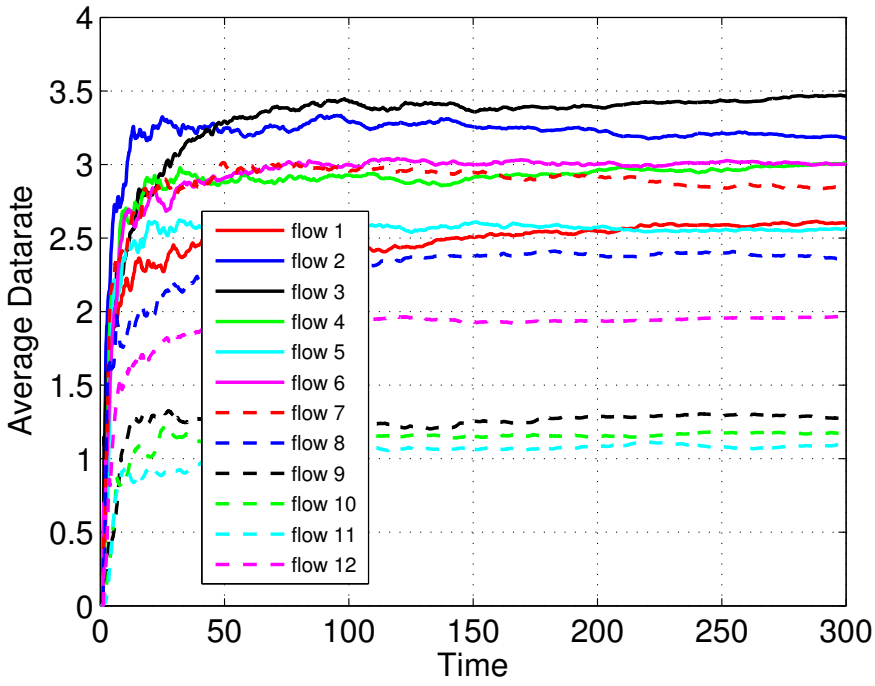

Figure 5.10. Flow Rate Allocation, AF for flow 12

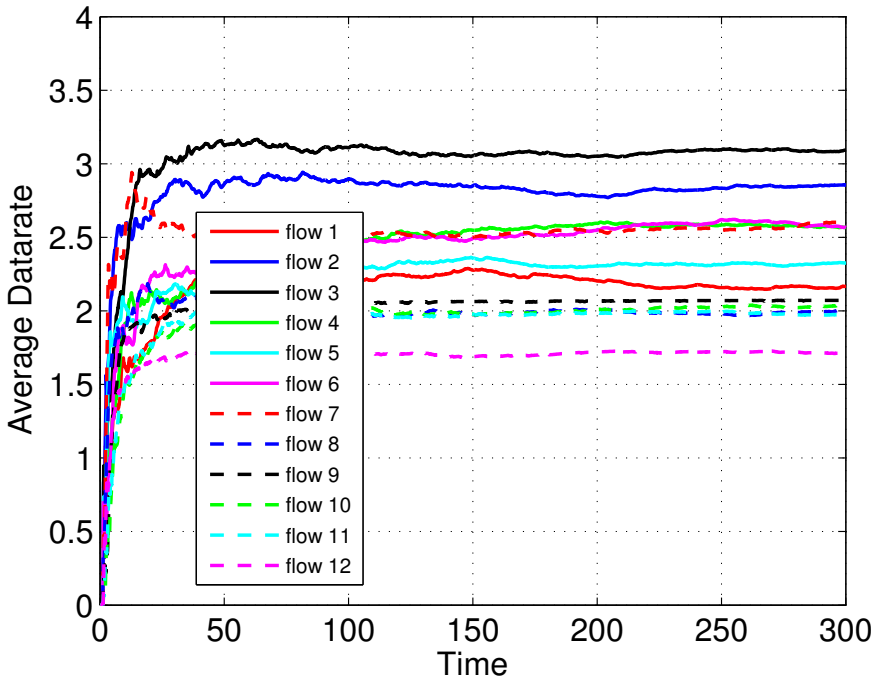

Figure 5.11. Flow Rate Allocation, MRG for flow 9 and 11, MRGUBR for flow 10, AF for flow 12

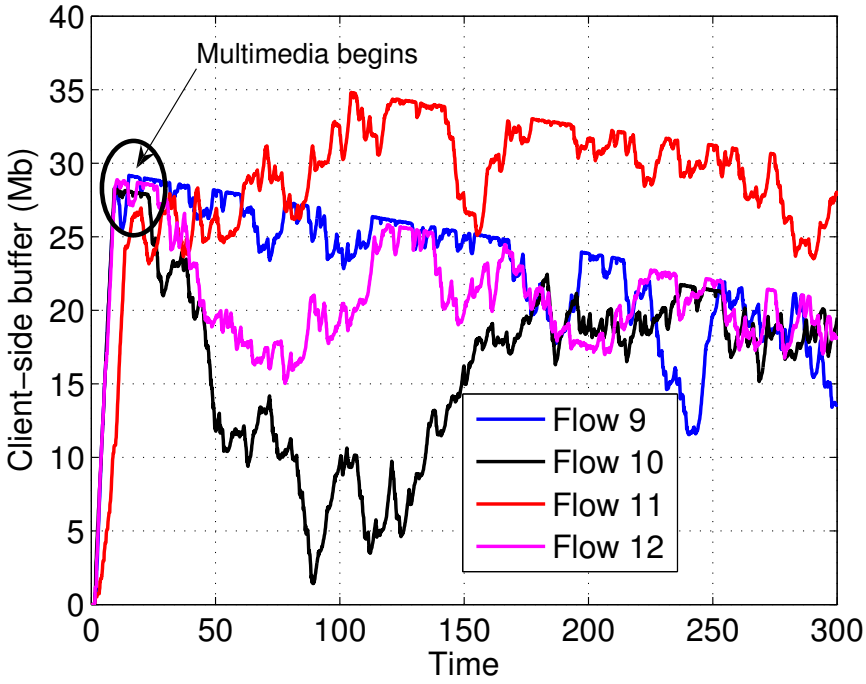

Figure 5.12. Client-side buffer by using MRG Control Law for Flow 9, 10, 11, and 12

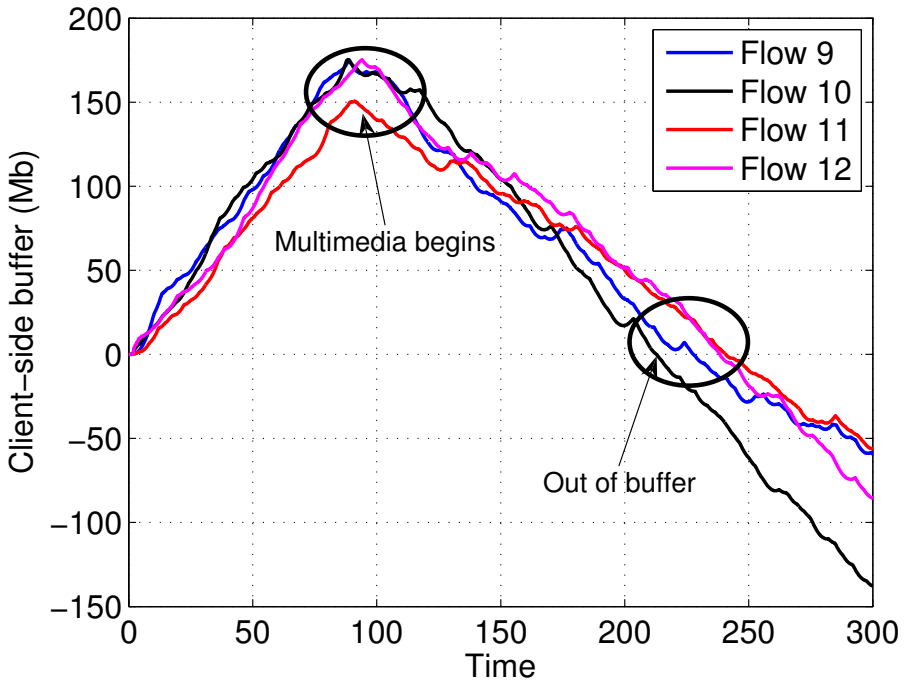

Figure 5.13. Client-side buffer by using TCP Control Law for Flow 9, 10, 11, and 12

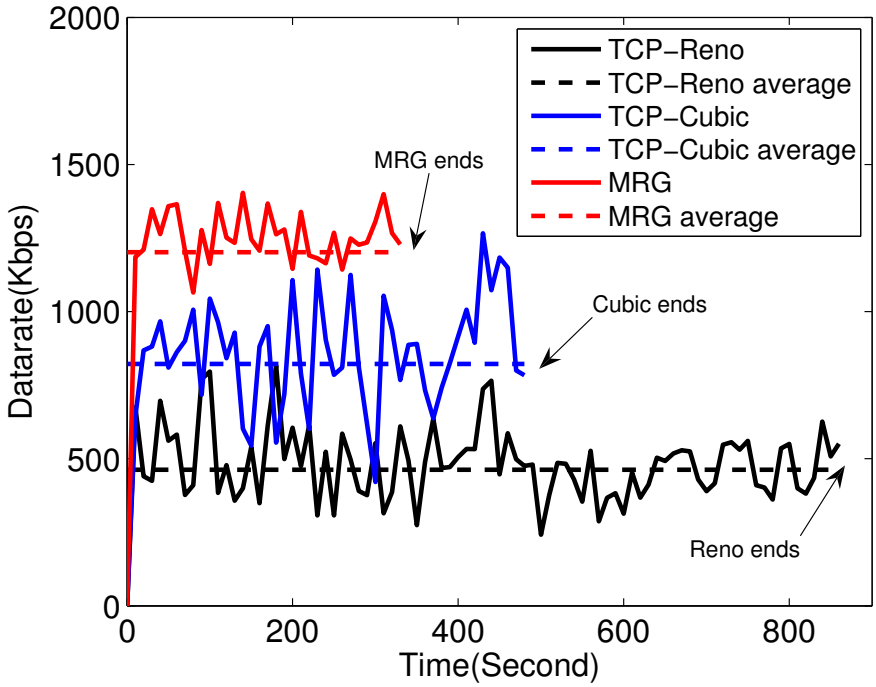

Figure 5.14. Average Throughput of 10 seconds

CHAPTER 6

A QOS-AWARE DCCP CONGESTION CONTROL MECHANISM FOR MULTIMEDIA STREAMING

6.1 Overview

The Datagram Congestion Control Protocol (DCCP) was designed to provide congestion control for unreliable applications, such as voice-over-IP and IPTV. But the current congestion control mechanisms of DCCP do not have the ability to support Quality of Service (QoS) features. This chapter aims to design an end-to-end QoS-aware DCCP Congestion Control Identifier (CCID) for multimedia streaming. Based on TERSE methodology, we design an end-to-end traffic control mechanism to support Real-time Delay Adaptive (RDA) and Real-time Rate Adaptive (RRA) multimedia streaming. The mechanism possesses several provable properties, including friendliness to TCP, stability, and optimality. The theoretical upper bound of guaranteed minimum rate is derived and verified through simulations. This chapter also compares the theoretical packet drop ratio of proposed DCCP-QoS mechanism with that of the DCCP-TCP-like mechanism, and verifies the result by simulation. The experimental results show that the proposed mechanism can provide minimum rate guarantee for multimedia streamings and maintain a lower packet drop ratio.

The rest of the chapter is organized as follows: Section 6.2 presents the end-to-end QoS aware congestion control mechanism. Section 6.3 analyzes the minimum guaranteed rate of DCCP-QoS flows. Section 6.4 compares the packet drop ratio of the proposed DCCP-QoS congestion mechanism to that of the DCCP-TCP-like congestion mechanism. Section 6.5 evaluates the performance of the DCCP-QoS mechanism. Finally, Section 6.6 concludes

the chapter.

6.2 QoS-Aware CCID for DCCP

In this section, we derive the QoS-aware CCID for DCCP, i.e., DCCP-QoS.

6.2.1 Congestion Indicator

The family of TCP protocols (Tahoe, Reno, Vegas etc.) was designed to serve the reliable applications. They cannot accept any packet loss. If there is a packet loss event, TCP considers the path is congested. The packet loss is sensed in two ways in TCP: the three repeated acknowledgements (ACKs) and ACK timeout. Tahoe and Reno do congestion control only based on this information: when the congestion indicator $cg = 1$, the data rate is reduced to the half of the current rate (0 for ACK timeout). However, this congestion indicator may not respond to the network condition timely. For example, the queue in a router is almost full, but some flows may not sense congestion yet and still increase the data rate, thus causing the link congestion quickly; While in some cases, the packet drop is not due to queue overflow, but the flows that sense the packet drop still need to reduce the data rate. TCP Vegas ([12]) estimates the congestion by comparing the actual data rate with the expected data rate. The data rate is calculated based on RTT. When RTT becomes longer than a threshold, Vegas slows down its transmission rate. The RTT method can reflect the network condition more accurately. But it is possible that the path from the sender to the receiver is light loaded and the ACKs need much longer time to go back to the sender through jammed path. In this case, the sender may sense a false congestion and reduce its rate. To avoid false congestions caused by asymmetric path conditions, we use more accurate STT to estimate the path condition from the sender to the receiver. Due to the clocks in the sender and receiver may not be synchronized, the exact STT is hard to be measured. So we introduce a pseudo STT (explain later) as the congestion indicator. For simplicity, we will still use STT instead of

pseudo STT.

The congestion indicator cg is a function of STT, it is defined as:

$$cg = \begin{cases} 0 & stt \leq stt_{th}^{cg} \\ 1 & stt > stt_{th}^{cg} \end{cases} \tag{6.1}$$

where stt is the current measured STT and stt_{th}^{cg} is the congestion threshold. This means that the path is considered to be congested if the STT is over the threshold.

If we assume that the flow goes through only one critical link, stt can be expressed as:

$$stt = stt_{min} + \frac{q}{B} \tag{6.2}$$

where stt_{min} is the minimum STT which is the propagation time from the sender to the receiver, q denotes how many bytes are in the queue of the critical link router/switcher, and B is the bandwidth of the critical link.

The packet drop probability $p(q)$ of Random Early Drop (RED) queue takes the form,

$$p(q) = \begin{cases} 0 & 0 \leq q < q^{min} \\ \frac{q-q^{min}}{q^{max}-q^{min}} \cdot p^{max} & q^{min} \leq q \leq q^{max} \\ 1 & q > q^{max} \end{cases} \tag{6.3}$$

From Eqn. (6.2), we can obtain how many bytes q_{th} in the queue when we define the congestion.

$$q_{th} = (stt_{th}^{cg} - stt_{min}) \cdot B \tag{6.4}$$

We assume that q_{th} is within q_{min} and q_{max}, and the corresponding drop ratio of q_{th} in the queue is p_{th}. Taking Eqn. (6.4) into Eqn. (6.3), we can obtain relationship between the packet drop ratio and the queue length with the STT threshold stt_{th}^{cg} as:

$$stt_{th}^{cg} = stt_{min} + \frac{p_{th}(q^{max} - q^{min})/p^{max} + q^{min}}{B} \tag{6.5}$$

Eqn. (6.5) can be used to determine stt_{th}^{cg}. Its value has to ensure the largest drop ratio in a piece of time not to exceed a required drop ratio p_{th}. But it is difficult to know the

actual queue parameters, and hence we use a dynamic threshold to adaptively measure the path condition. In our mechanism, the stt_{th}^{cg} is updated as:

$$stt_{th}^{cg} = stt_{k}^{cg} - \Delta \cdot \delta_k \tag{6.6}$$

where stt_{k}^{cg} is the moving average of STT on the k-th packet loss event, δ_k is the standard deviation of STT on the k-th packet loss event, and Δ is a constant used to adjust the degree of sensitiveness of stt_{th}^{cg}.

stt_{k}^{cg} and δ_k are computed as:

$$stt_{k}^{cg} = (1-\omega) \cdot stt_{k-1}^{cg} + \omega \cdot stt_{sample}^{cg} \tag{6.7}$$

and

$$\delta_k = \sqrt{(1-\omega) \cdot (\delta_{k-1})^2 + \omega \cdot (stt_{sample}^{cg} - stt_{k}^{cg})^2} \tag{6.8}$$

here ω is a smoothing factor between 0 and 1, and stt_{sample}^{cg} is the measured STT for the latest packet loss event.

6.2.2 Scalar Function

We modify the scalar function $z(t,x)$ of TCP to distinct the path conditions at the congestions so that the sender can more adaptively adjust its sending rate to better match the path condition to achieve high data rate and minimize the packet drop ratio. Our path condition is measured based on STT. If STT is small, we consider the path is relative light loaded. Hence we encode STT into the scalar function $z(t,x)$ so that the increase and decrease rates are dependent on the current STT in both non-congested and congested cases. The scalar function $z(t,x)$ is defined as in Eqn. (6.9).

$$z(t,x) = \begin{cases} z_s \cdot \left(1 - \frac{stt - stt_{min}}{stt_{th}^{cg} - stt_{min}}\right) \cdot z_{be}(t,x) & \text{if } stt < stt_{th}^{cg} - \beta \cdot \delta \\ z_s \cdot \frac{\beta \cdot \delta}{stt_{th}^{cg} - stt_{min}} \cdot z_{be}(t,x) & \text{if } stt_{th}^{cg} - \beta \cdot \delta \leq stt \leq stt_{th}^{cg} \\ \frac{stt - stt_{th}^{cg}}{stt_{max} - stt_{th}^{cg}} \cdot z_{be}(t,x) & \text{otherwise} \end{cases} \tag{6.9}$$

Here β is a constant, and δ is the latest measured standard deviation.

In Eqn. (6.9), $z_{be}(t,x)$ is the scalar function of the best effort TCP given in Eqn. (4.12), and $z_s > 1$ is a constant scalar parameter. The key feature of $z(t,x)$ is that the increase and decrease rates are calculated based on stt, and hence the control law can quickly respond to path condition changes. The scalar function is z_s times of z_{be} when the path is very light, i.e. $stt = stt_{min}$. When the path is close to be congested (i.e., stt is close to the congestion threshold), the scalar function decreases to $z_s \cdot \frac{\beta \cdot \delta}{stt_{th}^{cg} - stt_{min}} \cdot z_{be}(x)$. If the path is congested, i.e. stt is greater than stt_{th}^{cg}, the decrease rate also depends on stt, greater stt, more decrease.

6.2.3 STT Measurement

In [132], Zhou proposed a vegas variant by using STT to do the congestion control. But he did not illustrate how to get STT from the sender to receiver. We show our STT measure method to obtain the STT without any modification to the packet header of the DCCP packet. In Internet, the sender and the receiver do not share the same system clock. There is no way to measure the accurate STT of the path from two endpoints. However, we can measure the queuing time of a packet from the sender to the receiver by using a pseudo STT which includes the time difference between two endpoints' clocks.

In the DCCP packet header, there are several options, i.e. Timestamp, Timestamp Echo, and Elapsed Time options, which concerning about helping DCCP protocol to explicitly measure RTT. Our proposed DCCP-QoS measures STT only by two DCCP packet header options without modifying the DCCP header. We use the Timestamp and the Elapsed Time options to calculate a pseudo STT which can serve as the actual STT. Figure 6.1 shows the formats of Timestamp option and Elapsed Time option. The timestamp value is a 32-bit unsigned integer that increases monotonically with time, at a rate of 1 unit per 10 microsec-

onds [60]. It is easy to obtain that the timestamp value wraps itself about every 11.9 hours.

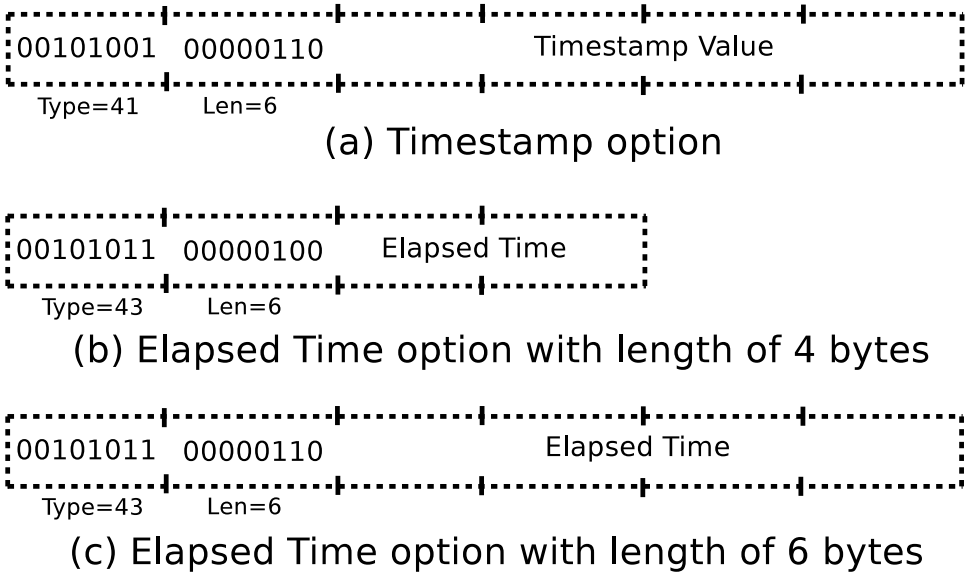

Figure 6.1. DCCP packet head options of Timestamp and Elapsed Time

Now we demonstrate how DCCP-QoS calculate the STTs. The sender records the sending time t_s of all the DCCP packets while it sends them. The receiver acknowledges the packets with the Timestamp option with value t_r. The DCCP uses ACK vectors to acknowledge several packets at the same time. The Timestamp option value t_r would be the arriving time of the last packet acknowledged by the vector if the receiver sends out the ACK vector when it receives the last packet in the vector. Sometimes, the acknowledgements may not be sent on the same time with the last received packet. The Elapsed Time option of ACK packet can be used together with the Timestamp option by the sender to obtain the real arriving time on the receiver of data packet. In the following discussion, we would take the Timestamp as the arriving time of the last packet on the receiver for convenience. After received the acknowledgement packet, the sender retrieves the Timestamp value t_r from the packet head and uses it minus the sending system time t_s of the last acknowledged packet. We call the result of $t_r - t_s$ pseudo STT and denote it as $pstt$. The Timestamp value t_r increases monotonically

with the system time of the receiver. By assuming the difference between Timestamp and receiver's system time is d_t (receiver's system time minus Timestamp value) and the difference between sender's system time and receiver's system time is d_s (sender's system time minus receiver's system time), it is easy to obtain that the STT from the sender to the receiver is $stt = t_r + d_t + d_s - t_s$. So the pseudo STT $pstt$ can be expressed as $pstt = stt - ds - dt$. Although there is no way to obtain the accurate value of d_s, it should be fixed if the users do not change the computers' system time of the sender or the receiver. And the d_t value is changed while the timestamp value wrap itself at a rate about 11.9 hours once [60]. So in normal situations, the $d_s + d_t$ would not change for a long time. We denote the pseudo STT of packet i as $pstt_i$ and the STT of packet i as stt_i. It is clear that $pstt_i = stt_i - d_s - d_t$ and we proved that $pstt_i - pstt_j = stt_i - stt_j$.

Although $pstt$ is not the real STT value, we only care the relative values among different STTs. This is because the congestion indicator and rate adjust are based on the difference between STT and the STT threshold. Hence the pseudo STT can be good enough to measure the path condition.

6.2.4 QoS-Aware Congestion Control

Based on the utility function of TCP and the scalar function in Eqn. (6.9), we can immediately derive the congestion control mechanism (called as DCCP-QoS control mechanism) as the follows:

(i) Without congestion, the increase rate is given in Eqn. (6.10).

$$\dot{x} = \begin{cases} z_s \cdot \frac{(3r(x)-1)x}{2} & \text{if } x < x_s \\ z_s \cdot \left(1 - \frac{stt - stt_{min}}{stt_{th}^{cg} - stt_{min}}\right)\left(\frac{(r(x)-1)x}{2} + r(x)\mu\right) & \text{if } x \geq x_s \text{ and } stt < stt_{th}^{cg} - \beta \cdot \delta \\ z_s \cdot \frac{\beta \cdot \delta}{stt_{th}^{cg} - stt_{min}} \cdot \left(\frac{(r(x)-1)x}{2} + r(x)\mu\right) & \text{if } x \geq x_s \text{ and } stt_{th}^{cg} - \beta \cdot \delta \leq stt \leq stt_{th}^{cg} \end{cases} \tag{6.10}$$

Here $r(x)$ for MRG flows is given in Eqn. (3.4).

(ii) With congestion, the decrease rate is

$$\dot{x} = -\frac{stt - stt_{th}^{cg}}{stt_{max} - stt_{th}^{cg}} \cdot \frac{x}{2} \tag{6.11}$$

If $r(x) = 1$, the control mechanism in Eqns. (6.10) and (6.11) becomes the best effort control mechanism. There are several differences between our congestion control mechanism and DCCP-TCP-like. If there is no queuing delay (i.e., $stt = stt_{min}$), the sender increases the sending rate z_s times fast as the DCCP-TCP-like sender does. With queuing delay, the sender slows down the increase rate inversely proportional to STT. In the presence of congestion, the DCCP-TCP-like sender would always halve the sending rate. But here, the sender decreases the sending rate based on the measured STT. Only when $stt = stt_{max}$, i.e. the buffer is fully occupied, the congestion control halves the sending rate. In any other situations, the decrease rate is smaller than half.

If $r(x) > 1$, the control mechanism in Eqns. (6.10) and (6.11) is expected to provide a soft minimum guaranteed rate for RDA and RRD applications. If there is no congestion, the flow rate keeps increasing exponentially even in the congestion avoidance phase (i.e., $\theta > x \geq x_s$) until the minimum guaranteed rate is reached. Beyond that, the acceleration process is the same as that we described above. This indicates that whenever the minimum guaranteed rate is reached, the flow will complete for elastic part with other flows fairly (i.e., the same increase speed). Hence the proposed mechanism is TCP friendly.

Although the minimum guaranteed rate is explicitly specified as a flow level constraint, but the rate can still be below the targeted rate. This ensures that the Internet will not experience partial or complete shutdown in the presence of resource shortages.

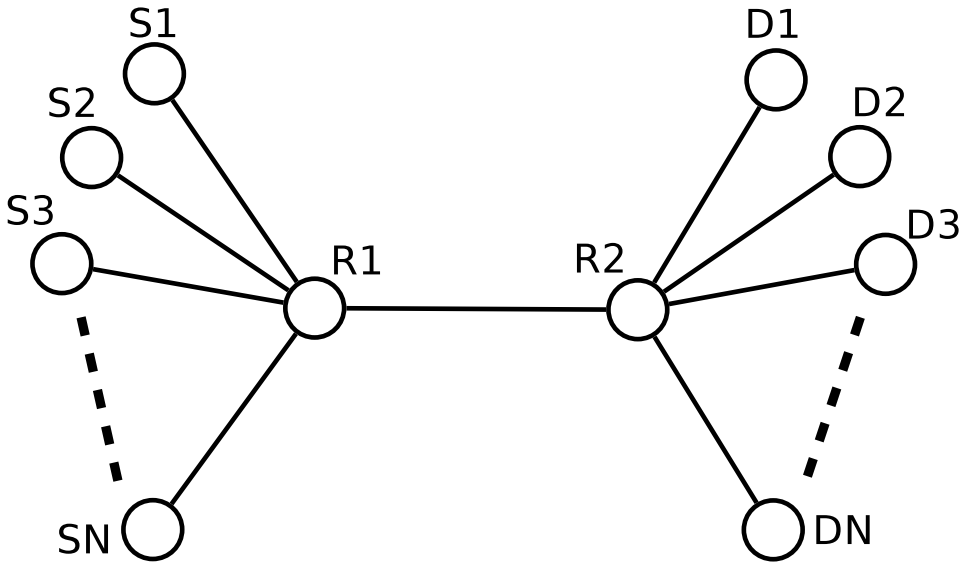

Figure 6.2. Netowrk Topology 5.1

6.3 Theoretic Upper Bounded Minimum Guaranteed Rate

The derived DCCP-QoS allows the sending rate below the minimum guaranteed rate. A natural question to ask is whether a targeted rate can be guaranteed or not. In this section, we derive an theoretical upper bounded minimum guaranteed rate. It means that all the targeted rate less than the upper bound can be guaranteed by the mechanism.

We consider a network with only one congested link as shown in Figure 6.2. Let N_t TCP flows labeled $i = 1, \ldots, N_t$ and N_m DCCP-QoS flows labeled as $N_t + 1, \ldots, N$ (here $N = N_t + N_m$) traverse the bottleneck link with bandwidth B.

The window size of TCP flow i can be modeled as,

$$\dot{W}_i(t) = \frac{1}{R_i(t)}[1 - p(q(t - R_i(t))] - \frac{W_i(t) \cdot W_i(t - R_i(t))}{2R_i(t - R_i(t))} \cdot p(q(t - R_i(t))) \qquad (6.12)$$

where $W_i(t)$ is the window size of flow i at time t, $R_i(t)$ is the RTT of flow i at time t, $p(q)$ is the packet drop probability of the congested router when the queue length is q.

For the DCCP/QoS flow, it can be expressed as in Eqn. (6.13),

$$\dot{W}_j(t) = \begin{cases} \frac{z_s \cdot (stt^{cg}_{th,j} - stt_j(t-R_j(t)))}{R_j(t) \cdot (stt^{cg}_{th,j} - stt_{min,j})} & \text{if } stt_j < stt^{cg}_{th,j} \\ -\frac{W_j(t) \cdot W_j(t-R_j(t)) \cdot (stt_j(t-R_j(t)) - stt^{cg}_{th,j})}{2R_j(t-R_j(t)) \cdot (stt_{max,j} - stt^{cg}_{th,j})} & \text{if } stt_j \geq stt^{cg}_{th,j} \end{cases} \tag{6.13}$$

Because stt^{cg}_{th} is used as the congestion indicator in DCCP-QoS flows. On the average, we can assume that the packet loss happens at $stt = stt^{cg}_{th}$. Hence the window increase and decrease probabilities of a DCCP-QoS flow can be approximated to be the same as these in a TCP flow.

$$\begin{aligned} \dot{W}_j(t) \simeq & \frac{z_s \cdot (stt^{cg}_{th,j} - stt_j(t-R_j(t)))}{R_j(t) \cdot (stt^{cg}_{th,j} - stt_{min,j})}[1 - p(q(t-R_j(t)))] \\ & - \frac{W_j(t) \cdot W_j(t-R_j(t)) \cdot (stt_j(t-R_j(t)) - stt^{cg}_{th,j})}{2R_j(t-R_j(t)) \cdot (stt^{cg}_{max,j} - stt^{cg}_{th,j})} p(q(t-R_j(t))) \end{aligned} \tag{6.14}$$

For a DCCP-QoS flow, when the congestion occurs, its rate is reduced by at most half, then it will go back to the targeted minimum rate within one RTT if r^m_{max} is equal to or over 3. In the case that the targeted rate is greater than the slow start threshold, the control law can be approximated as the rate dropped from the peak rate to the targeted rate. If the target rate is less than the slow start threshold, the targeted rate is absolutely guaranteed. Hence we do not consider this case. Hence the window size of DCCP-QoS flow j with targeted rate θ can be approximately modeled as:

$$\begin{aligned} \dot{W}_j(t) \simeq & \frac{z_s \cdot (stt^{cg}_{th,j} - stt_j(t-R_j(t)))}{R_j(t) \cdot (stt^{cg}_{th,j} - stt_{min,j})}[1 - p(q(t-R_j(t))] \\ & - \frac{(W_j(t) - \theta \cdot R_j(t)) \cdot W_j(t-R_j(t))}{R_j(t-R_j(t))} p(q(t-R_j(t))) \end{aligned} \tag{6.15}$$

$R_k(t)$ can be expressed as the follow,

$$R_k(t) = a_k + \frac{q(t)}{B} \tag{6.16}$$

where $k = 1, 2, ..., N$.

For the queue length $q(t)$,

$$\dot{q}(t) = -B + \sum_{k=1}^{N} \frac{W_k(t)}{R_k(t)} \tag{6.17}$$

where a_k is the fixed propagation delay of flow k.

Following the approximated method used in [80] and [87], take the expectation of each side of the Eqns. (6.12), (6.15) and (6.17), we can obtain

$$\begin{aligned} E[\dot{W}_i(t)] \simeq & \frac{1 - p(E[q(t - R_i(t))])}{E[R_i(t)]} \\ & - \frac{E[W_i(t)] \cdot E[W_i(t - R_i(t))]}{2E[R_i(t - R_i(t))]} \cdot p(E[q(t - R_i(t))]) \end{aligned} \tag{6.18}$$

Here, we assume $E[W_i(t) \cdot p(q(t - R_i(t)))] \simeq E[W_i(t)] \cdot p(E[q(t - R_i(t))])$,

$$\begin{aligned} E[\dot{W}_j(t)] \simeq & \frac{z_s \cdot (E[stt_{th,j}^{cg}] - E[stt_j(t - R_j(t))])}{E[R_j(t)] \cdot (E[stt_{th,j}^{cg}] - stt_{min,j})} \cdot (1 - p(E[q(t - R_j(t))]) \\ & - \frac{(E[W_j(t)] - \theta \cdot E[R_j(t))] \cdot E[W_j(t - R_j(t))])}{E[R_j(t - R_j(t))]} \cdot E[p(q(t - R_j(t)))] \end{aligned} \tag{6.19}$$

and

$$E[\dot{q}(t)] = -B + \sum_{k=1}^{N} \frac{E[W_k(t)]}{E[R_k(t)]} \tag{6.20}$$

The system stable point can be achieved at $E[\dot{W}_k] = 0$ and $E[\dot{q}] = 0$.

For N_t TCP flows,

$$E[\dot{W}_i(t)] = 0 \Longrightarrow E[W_i(t)] \cdot E[W_i(t - R_i(t))] = \frac{2(1 - p(E[q(t - R_i(t))]))}{p(E[q(t - R_i(t))])} \tag{6.21}$$

In the stable point, we assume $E[W_i(t)] = E[W_i(t - R_i(t))]$. That means the average data rate of a TCP flow should be fixed after reaching the stable state. So we can obtain

$$E[W_i] = \sqrt{\frac{2(1 - p(E[q]))}{p(E[q])}} \tag{6.22}$$

For N_m DCCP-QoS flows,

$$\begin{aligned} &E[\dot{W}_j(t)] = 0 \Longrightarrow \\ &\quad E[W_j]^2 - \theta \cdot E[R_j] \cdot E[W_j] = \frac{z_s \cdot (E[stt_{th,j}^{cg}] - E[stt_j])(1 - p(E[q]))}{(E[stt_{th,j}^{cg}] - stt_{min,j}) \cdot p(E[q])} \end{aligned} \tag{6.24}$$

We can obtain the general solution of Eqn. (6.23) as:

$$\begin{aligned} E[W_j] =& \theta \cdot E[R_j]/2 \\ &+ \left(\sqrt{(\theta \cdot E[R_j])^2 + \frac{4 \cdot z_s \cdot (E[stt_{th,j}^{cg}] - E[stt_j])(1 - p(E[q]))}{(E[stt_{th,j}^{cg}] - stt_{min,j}) \cdot p(E[q])}} \right) /2 \end{aligned} \tag{6.25}$$

Because every STT is no greater than sst_{max}, we approximate $E[stt_{th,j}^{cg}] \simeq stt_{max,j}$, and then have

$$\begin{aligned} E[W_j] \simeq& \theta \cdot E[R_j]/2 \\ &+ \left(\sqrt{(\theta \cdot E[R_j])^2 + \frac{4 \cdot z_s \cdot (stt_{max,j} - E[stt_j])(1 - p(E[q]))}{(stt_{max,j} - stt_{min,j}) \cdot p(E[q])}} \right) /2 \end{aligned} \tag{6.26}$$

where $E[stt_j] = stt_{min,j} + E[q]/B$.

For the queue length q,

$$E[\dot{q}] = 0 \Longrightarrow \sum_{k=1}^{N} \frac{E[W_k]}{a_k + \frac{E[q]}{B}} = B \tag{6.27}$$

From these equations, we can solve $E[W_k]$ and $E[R_k]$ for flows k ($k = 1, 2, ..., N$). The average rate of flow k at the congestion time is estimated as $E[W_k]/E[R_k]$.

Recall the control law of DCCP-QoS, its rate increases exponentially when the flow rate is smaller than the minimum guaranteed rate θ. After the DCCP-QoS flow reaches its targeted rate, it increases the rate linearly as a TCP flow does. The relationship between the DCCP-QoS flow's minimum guaranteed rate θ and $E[W_j]$ can be expressed as follow,

$$n = \frac{E[W_j]/E[R_j] - \theta}{\mu_j} \tag{6.28}$$

where

$$\mu_j = \frac{z_s \cdot (stt_{max,j} - E[stt]))}{E[R_j(t)] \cdot (stt_{max,j} - stt_{min,j})} \tag{6.29}$$

.

Figure 6.3 shows the approximated rate changing of a DCCP-QoS flow between two congestions an the average situation. After the DCCP-QoS flow reaches its minimum guaranteed rate θ, it enters the additive increase phase. After n RTTs, it reaches the rate $E[W_j]/E[R_j]$ and encounters the congestion event again.

To ensure that the average rate exceeds the minimum guaranteed rate θ, the shaded area in Figure 6.3 above the line of θ must not be less than the area a. In the most congested situation, when the congestion occurs, the DCCP-QoS flow halves the flow rate, i.e., reduce the rate to be $E[W_j]/E[R_j]/2$. We assume the minimum guaranteed rate θ is between $E[W_j]/E[R_j]$ and $E[W_j]/E[R_j]/2$. Note that θ is absolutely guaranteed if θ is smaller than $E[W_j]/E[R_j]/2$. Starting from $E[W_j]/E[R_j]/2$, the DCCP-QoS flow doubles its rate when the rate is smaller than θ. After one RTT, it reaches the minimum guaranteed rate θ, and then increases its rate additively. The area a is equal to $(\theta - \frac{E[W_j]}{2E[R_j]}) \cdot E[R_j]$. If the area of a is smaller than the shaded area above the θ, the minimum guaranteed rate θ can be achieved, so we have

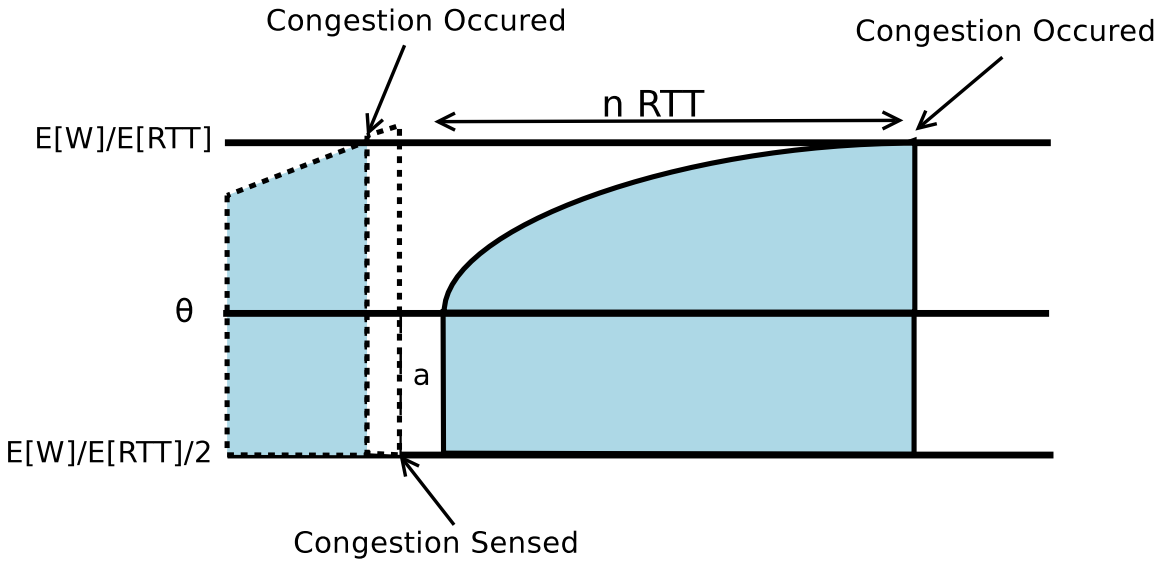

Figure 6.3. A DCCP-QoS flow congestion cycle

$$\frac{n(n+1)}{2} \cdot \mu_j \geq \theta - \frac{E[W_j]}{2E[R_j]} \tag{6.30}$$

or

$$\theta \leq \frac{E[W_j]}{E[R_j]} - \frac{\sqrt{4\mu_j \frac{E[W_j]}{E[R_j]} + 9\mu_j^2} - 3\mu_j}{2} \tag{6.31}$$

Eqn. (6.31) gives an upper bounded minimum guaranteed rate of DCCP-QoS flow j under certain network conditions. It means that all the targeted rate no greater than the value computed by the Eqn. (6.31) can be guaranteed by using the DCCP-QoS control mechanism.

We verify the theoretical upper bounded minimum guaranteed rate by NS-2 simulations. In the experiment, we tried to find the maximum rate can be guaranteed. If the minimum rate is less than that rate, the required rate can be ensured for sure. If the required rate is over that rate, it cannot be guaranteed, but still a little increase as the required rate

increases. In the simulations, we set $\omega = 0.3$, $\beta = 0.5$ and $\delta = 0.5$. First, we set $N = 11$, i.e., there are 11 flows in the network. Among these flows, flows 1-10 are TCP flows, and flow 11 is a DCCP-QoS flow. The bandwidth for the bottleneck link is 100Mbps. Figure 6.4 shows the theoretical minimum guaranteed rate, the actual minimum guaranteed rate, and the rate of flow 11 using DCCP-TCP-like control mechanism. Here, RTTs of TCP flows 1-10 linearly increase from 20ms to 56ms in all the cases while the RTT of flow 11 changes from 30ms to 100ms. The results show that the theoretical result in Eqn. (6.30) is close to the simulation results. The simulation results give a little bit higher than the theoretical upper bound, this is because we assume the rate at the congestion is always reduced by half, but it may be less than the half rate in the real case (dependent on STT). The results also show that the minimum guaranteed rate of DCCP-QoS flow can greatly improve the performance compared to DCCP-TCP-like mechanism.

Now let us exam the rate allocation of link shared by multiple DCCP-QoS flows and multiple NRE TCP flows. In the simulation, we set 100 flows, the shared bandwidth is 500Mbps. All flows have the same RTT 60ms. Among these 100 flows, there are N_m DCCP-QoS flows and $100 - N_m$ TCP flows. We change N_m from 10 to 100. From Figure 6.5, one can see that the maximum minimum guaranteed rate of DCCP-QoS flows decreases from about 10 Mbps to 4.5Mbps when N_m increases from 10 to 100. The DCCP-QoS flows have much higher minimum guaranteed rate than that of TCP flow when the number of DCCP-QoS flows is small. Even if all the flows are DCCP-QoS flows, the minimum guaranteed flow rate is the same as that of TCP flows. This shows that the DCCP-QoS can significant improve the minimum guaranteed rate and enable QoS features.

6.4 Packet Drop Ratio Analysis

In this section, we compare the packet drop ratio of a DCCP-QoS flow to that of a DCCP-TCP-like flow on the condition that they achieve the same flow rate. Considering a DCCP-

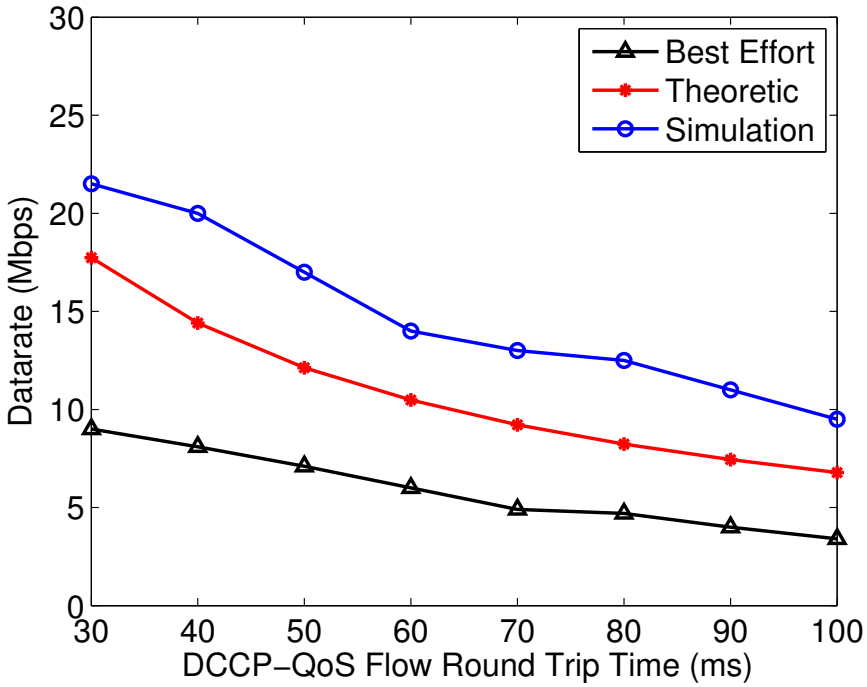

Figure 6.4. The maximum guaranteed minimum rate of DCCP-QoS flows.

QoS flow and a DCCP-TCP-like flow pass through a critical link with many other TCP flows as shown in Figure 6.2. We assume that the link uses RED queue management mechanism and modulate the RTT parameter to make sure that the two DCCP flows can achieve the same flow rate to compare their packet drop ratios.

Let's consider the expected congestion cycle from a viewpoint on the critical link shown in Figure 6.6. In the expected congestion cycle, we assume that the queue level of the critical link increases linearly from 0 to q_{max}; two DCCP flows sense the network congestion and halve their sending rate when the queue length reached the q_{max}. In this chapter, we does not model the timeout events. So in the expected congestion cycle shown in Figure 6.6, we assume that the two DCCP flows start to increase their sending rate from $E[W]/2$ and halve their sending rate when their windows achieve $E[W]$. The DCCP-TCP-like flow and

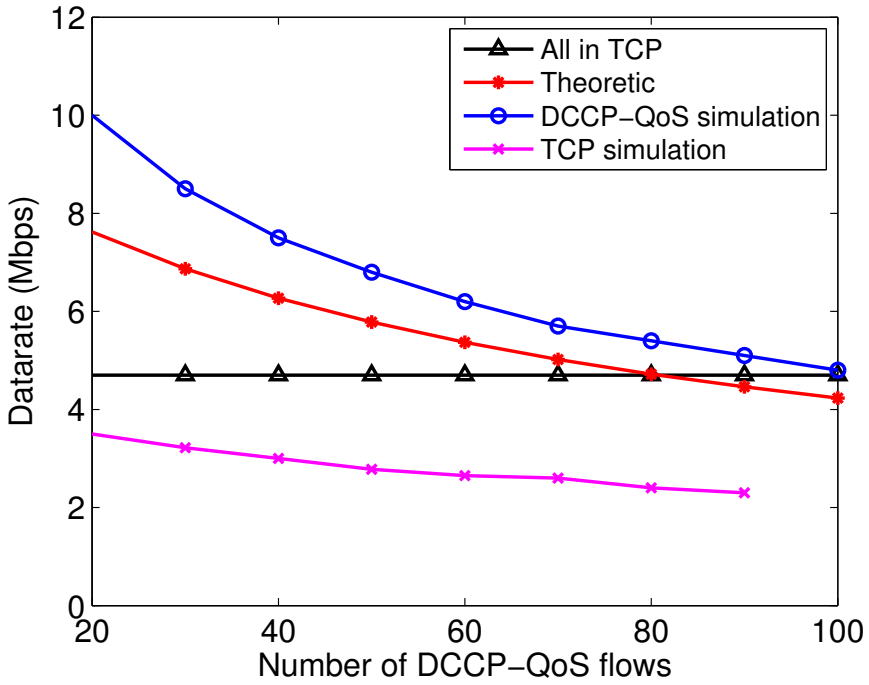

Figure 6.5. The maximum guaranteed minimum rate of DCCP-QoS

the DCCP-QoS flow do not have the same $E[W]$.

Assume that the packet queuing time of both DCCP flows is quite smaller than their RTTs. The DCCP-TCP-like flow's transmission rate increases linearly within the congestion cycle. The DCCP-QoS flow's increasing rate is influenced by the scalar function in Eqn. (6.9). It is obvious that the DCCP-QoS flow's transmission rate is an increasing concave function within the expected congestion cycle.

The expected DCCP-QoS window can be modelled in Eqn.(6.32):

$$W_q(t) = \begin{cases} \theta \cdot RTT_q & t < RTT_q \\ \theta \cdot RTT_q + ps \cdot \int_{RTT_q}^{t} \frac{z_s(stt_{max} - stt(x - RTT_q))}{stt_{max} - stt_{min}} dx & t \geq RTT_q \end{cases} \tag{6.32}$$

where ps is the packet size.

Assume that the estimated queue size of RED increases linearly in an expected con-

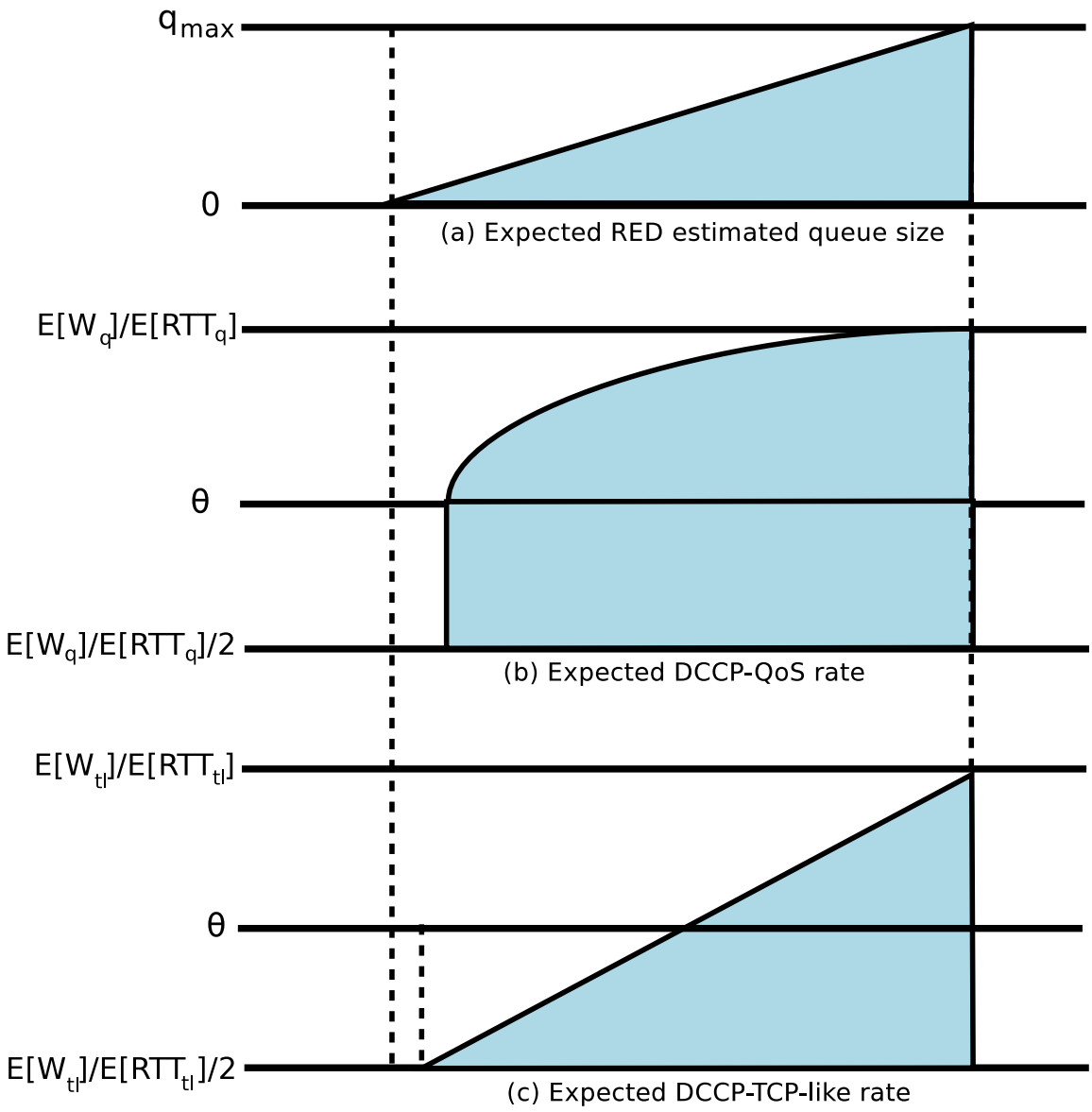

Figure 6.6. A Congestion Cycle

gestion cycle with time length T, we can transform Eqn. (6.32) into Eqn.(6.33).

$$W_q(t) = \begin{cases} \theta \cdot RTT_q & t < RTT_q \\ \theta \cdot RTT_q + ps \cdot \int_{RTT_q}^{t} \frac{z_s(q_{max} - q_{max}(x - RTT_q)/T)}{q_{max}} dx & t \geq RTT_q \end{cases} \tag{6.33}$$

By calculating the integral of Eqn. (6.33), we can obtain Eqn. (6.34).

$$W_q(t) = \begin{cases} \theta \cdot RTT_q & t < RTT_q \\ \theta \cdot RTT_q + ps \cdot z_s \cdot \left(t - RTT_q + \frac{RTT_q \cdot t}{T} - \frac{t^2}{2T} - \frac{RTT_q^2}{2T} \right) & t \geq RTT_q \end{cases} \tag{6.34}$$

The expected DCCP-TCP-like window can be modelled as:

$$W_{tl}(t) = \begin{cases} \frac{E[W_{tl}]}{2} & t < RTT_{tl} \\ \frac{E[W_{tl}]}{2} + \frac{E[W_{tl}] \cdot (t - RTT_{tl})}{2(T - RTT_{tl})} & t \geq RTT_{tl} \end{cases} \tag{6.35}$$

It is obvious that $E[W_{tl}]$ satisfy the following relationship,

$$\left(\theta - \frac{E[W_{tl}]}{2RTT_{tl}} \right) \cdot T = (T - RTT_{tl}) \cdot \frac{E[W_{tl}]}{4RTT_{tl}} \tag{6.36}$$

and it is easy to obtain $E[W_{tl}]$ of the DCCP-TCP-like flow as:

$$E[W_{tl}] = \frac{4 \cdot \theta \cdot T \cdot RTT_{tl}}{3T - RTT_{tl}} \tag{6.37}$$

By bringing Eqn. (6.37) into Eqn. (6.35), we can obtain

$$W_{tl}(t) = \begin{cases} \frac{2 \cdot \theta \cdot T \cdot RTT_{tl}}{3T - RTT_{tl}} & t < RTT_{tl} \\ \frac{2 \cdot \theta \cdot T \cdot RTT_{tl} \cdot (T + t - 2RTT_{tl})}{(3T - RTT_{tl})(T - RTT_{tl})} & t \geq RTT_{tl} \end{cases} \tag{6.38}$$

The RED queue's drop ratio takes the following form:

$$p(q) = \begin{cases} 0 & 0 \leq q < q^{min} \\ \frac{q - q^{min}}{q^{max} - q^{min}} \cdot p^{max} & q^{min} \leq q \leq q^{max} \\ 1 & q^{max} < q \end{cases} \tag{6.39}$$

From Eqn. (6.39), we can find that the RED queue drops packets with a probability of $p(q)$. Considering the extreme case that all the packets of a congestion window of a flow are waiting in the queue, the RED queue would drop $W(t) \cdot p(q(t))$ packets of a flow. So in one congestion cycle, a flow's upper bound of dropped/marked packets can be calculated by Eqn. (6.40).

$$D = \int_0^T \frac{W(t)}{RTT} \cdot t \cdot p(q(t)) dt \tag{6.40}$$

Within the expectation congestion cycle, the ratio of two flows' packets in the queue to the window size should be the same. So the packet drop ratio of DCCP-QoS and DCCP-TCP-like can be compared by their upper bounds of lost packets.

Assume that the estimated queue size is increasing linearly with a fixed slope, then we obtain the following equation of the queue size raise from 0 to q^{min} as

$$t^{min} = \frac{q^{min} \cdot T}{q^{max}} \tag{6.41}$$

where t^{min} is the time that the queue size raise from 0 to q^{min}.

We can obtain Eqn. (6.42) by bringing Eqn. (6.41) into Eqn. (6.39).

$$p(t) = \begin{cases} 0 & 0 \leq t < t^{min} \\ \frac{t \cdot q^{max} - T \cdot q^{min}}{T \cdot (q^{max} - q^{min})} \cdot p^{max} & t^{min} \leq t \leq T \end{cases} \tag{6.42}$$

The Eqn. (6.40) can be rewriten as follow.

$$D = \int_{t^{min}}^{T} \frac{W(t)}{RTT} \cdot t \cdot \frac{t \cdot q^{max} - T \cdot q^{min}}{T \cdot (q^{max} - q^{min})} \cdot p^{max} dt \tag{6.43}$$

It is obvious that RTTs of both two flows are smaller than t^{min}. Then in calculating the drop packets, we can only consider the $t \geq RTT$ part of $W(t)$ functions.

For the DCCP-QoS flow, we can obtain its upper bound of lost packets in the expected phase as Eqn.(6.44).

$$\begin{aligned} D_q = \int_{t^{min}}^{T} & \left[\theta + ps \cdot z_s \left(\frac{t}{RTT_q} + \frac{t}{T} - \frac{t^2}{2T \cdot RTT_q} - 1 - \frac{RTT_q}{2T} \right) \right] \\ & \cdot t \cdot \frac{t \cdot q^{max} - T \cdot q^{min}}{T \cdot (q^{max} - q^{min})} \cdot p^{max} dt \end{aligned} \tag{6.44}$$

By calculating the integral of Eqn. (6.44), we can obtain the Eqn.(6.45).

$$\begin{aligned} D_q =& \frac{p^{max}}{T \cdot (q^{max} - q^{min})} \cdot \left\{ \frac{ps \cdot z_s \cdot (2T + RTT_q) - 2\theta \cdot T}{4} \cdot q^{min} \cdot (T^2 - (t^{min})^2) \right. \\ & + \left[\frac{\theta \cdot q^{max}}{3} - ps \cdot z_s \cdot \left(\frac{(2T + RTT_q) q^{max}}{6T} + \frac{T \cdot q^{min}}{3RTT_q} + \frac{q^{min}}{3} \right) \right] \cdot (T^3 - (t^{min})^3) \\ & + ps \cdot z_s \left(\frac{q^{max}}{4RTT_q} + \frac{q^{max}}{4T} + \frac{q^{min}}{8RTT_q} \right) \cdot (T^4 - (t^{min})^4) \\ & \left. - \frac{ps \cdot z_s \cdot q^{max}}{10T \cdot RTT_q} \cdot (T^5 - (t^{min})^5) \right\} \end{aligned} \tag{6.45}$$

The lost packets in the expected phase of the DCCP-TCP-like flow is as Eqn. (6.46).

$$D_{tl} = \int_{t^{min}}^{T} \frac{2 \cdot \theta \cdot T \cdot (T + t - 2RTT_{tl})}{(3T - RTT_{tl})(T - RTT_{tl})} \cdot t \cdot \frac{t \cdot q^{max} - T \cdot q^{min}}{T \cdot (q^{max} - q^{min})} \cdot p^{max} dt \tag{6.46}$$

By calculating the integral of Eqn. (6.46), we can obtain the Eqn.(6.47).

$$\begin{aligned} D_{tl} =& \frac{2p^{max} \cdot \theta}{(3T - RTT_{tl})(T - RTT_{tl})(q^{max} - q^{min})} \\ & \cdot \left\{ \frac{(2RTT_{tl} - T) \cdot T \cdot q^{min}}{2} \cdot [T^2 - (t^{min})^2] \right. \\ & + \frac{(T - 2RTT_{tl}) \cdot q^{max} - T \cdot q^{min}}{3} \cdot (T^3 - (t^{min})^3) \\ & \left. + \frac{q^{max}}{4} \cdot (T^4 - (t^{min})^4) \right\} \end{aligned} \tag{6.47}$$

These equations only give the upper bound of lost packets in an expected time phase. We can assume that the average drop packets of different flows in the time cycle have the same proportion to their flows' window size W. To compare the lost packets between two DCCP flows, we define a parameter R_{tl}^q as,

$$R_{tl}^q = \frac{D_q}{D_{tl}} \tag{6.48}$$

The value of Eqn.(6.48) can be considered as the ratio of lost packets of DCCP-QoS flow to those of DCCP-TCP-like flow.

We use the average rate of DCCP-QoS flow to work out the cycle time T. Recall the $E[W_q]$ of DCCP-QoS flow by model the DCCP-QoS flow as average linearly increases after it achieves the target rate. By using the Eqn. (6.28), we can obtain the expected number of DCCP-QoS RTTs. Then we can get T by $T = (n+1) \cdot RTT_q$.

We construct a network topology as in Figure 6.2. There are one DCCP-TCP-like flow, one DCCP-QoS flow and 100 TCP flows in the topology. The 100 TCP flows are with randomly generated uniform distributed RTT from 100ms to 260ms. From Figure (6.7), we can find that the DCCP-QoS flow and the DCCP-TCP-like can both achieve about 1.5Mbps. Figure (6.8) shows the theoretic R_{tl}^q and the actual value of the number of DCCP-QoS flow's drop packets divided by that of DCCP-TCP-like flow. From Figure (6.8), we can see that the DCCP-QoS flow has lower packet drop ratio. The packet drop ratio in DCCP-QoS flow is about 0.85 times of the DCCP-TCP-like flow. It means that the DCCP-QoS flow drops 15% packets less than that of the DCCP-TCP-like flow.

6.5 Performance Evaluation

The DCCP-QoS derived in the previous section is rate based. It is shown in [83] that the corresponding control mechanism in the discrete time domain with finite control epochs and granularity lead to stable and near optimal control. Hence we map the continuous, rate based DCCP-QoS into a window based packet control protocol, in line with the TCP congestion control.

To translate the rate based control mechanism into a window based packet control protocol, we need to measure the transmission rate at the sender. A RTT based rate calculation is used. The rate is estimated as the follow:

$$x(t) = w(t) \cdot s/T(t) \tag{6.49}$$

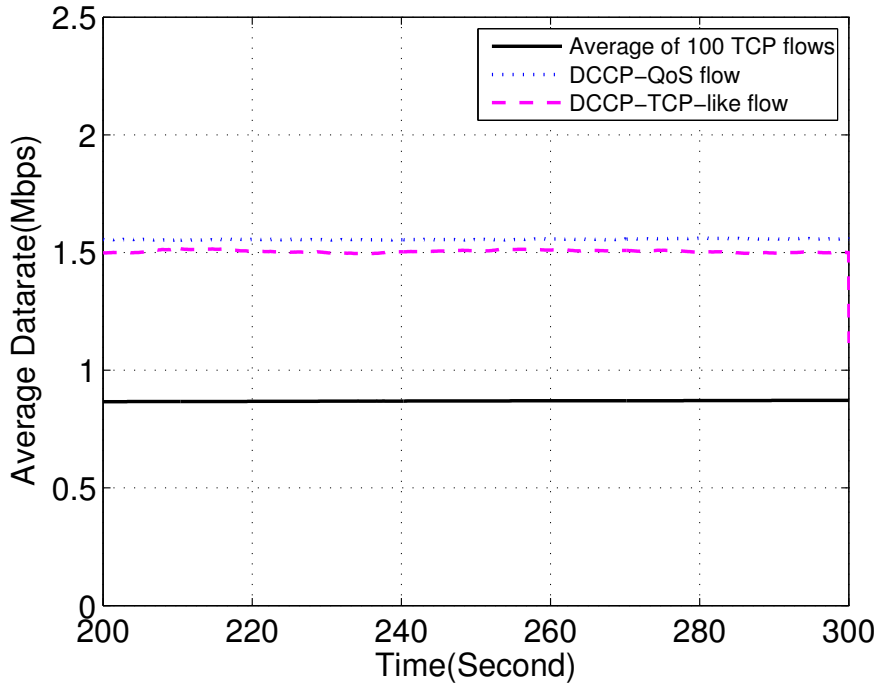

Figure 6.7. Throughput

where $x(t)$ is the transmission rate at time t, $w(t)$ is the sliding window size at the sender at time t, s is the packet size, and $T(t)$ is the average measured RTT at time t. The average measured RTT at time t can be calculated as follows:

$$T(t) = (1 - \eta) \cdot T(t - 1) + \eta \cdot MT(t) \tag{6.50}$$

with $0 < \eta \leq 1$, $T(t - 1)$ is the latest calculated average RTT before time t, $MT(t)$ is the measured RTT at time t, and η is a tunable parameter, i.e., the relative weight of the historical RTT estimations on the estimation of the current RTT. The sender uses the measured rate as well as the targeted rate to do congestion control.

We consider a network with only one congested link as Figure 6.2. There are 10 TCP flows, 5 DCCP-TCP-like flows and 5 DCCP-TFRC flows. The capacity of the link between

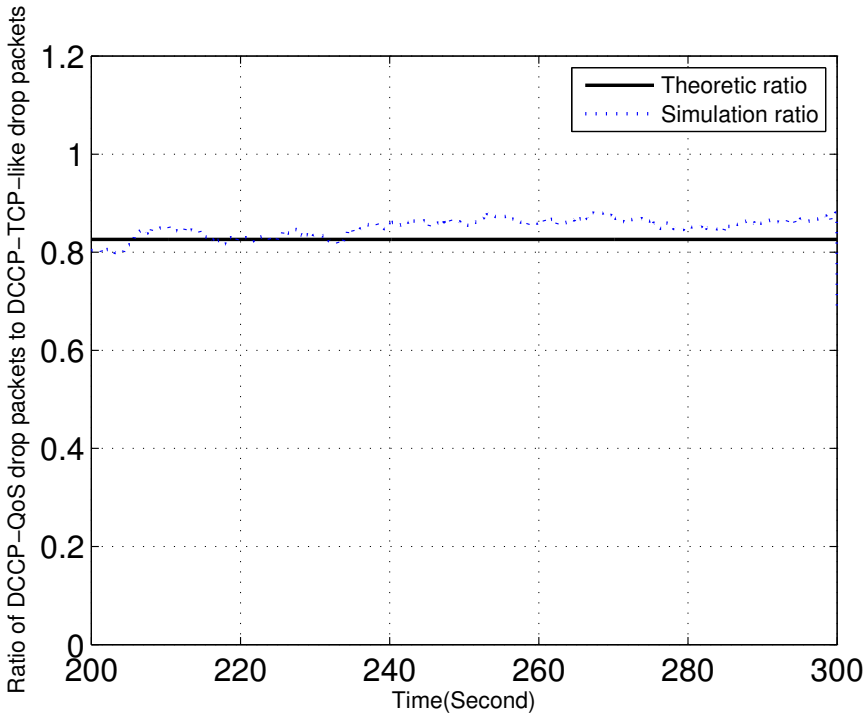

Figure 6.8. Drop Packets Proportion of QoS to TCP-like

node R1 and R2 is 100Mbps and the other links are all 200Mbps. Flow i comes from Si to Di where $1 \leq i \leq N$. All the 20 flows are symmetric with RTT 60ms.

Figure 6.9 shows the average datarates of three different type flows. It is clear that the average datarate of TCP flows and DCCP-TCP-like can achieve the stable state much faster than that of the DCCP-TFRC one. Figure 6.10 shows the average drop ratio of DCCP-TCP-like flows and DCCP-TFRC flows. The DCCP-TFRC flows can maintain a lower drop ratio than DCCP-TCP-like flows. Both of these two characteristics are due to the moderate rate changing method of DCCP-TFRC protocol. When the bandwidth is available, the DCCP-TFRC cannot increase the data transport speed fast. When there are some packets losses, the DCCP-TFRC protocol can send more data packets during the time that TCP or DCCP-TCP-like flows halve their congestion windows.

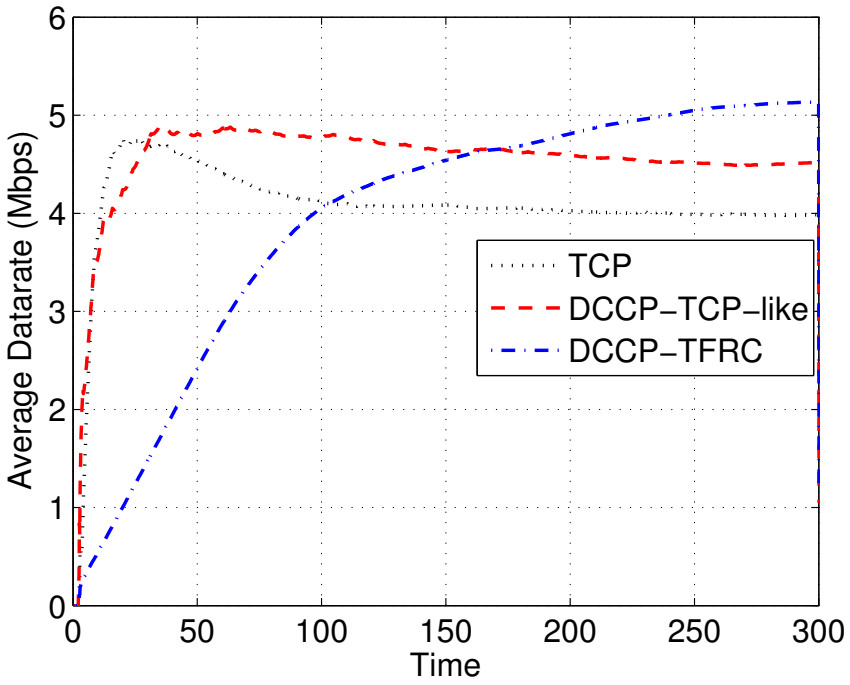

Figure 6.9. TCP, DCCP-TCP-like and DCCP-TFRC

In the simulations from Figure 6.11 to Figure 6.26, we replace three DCCP-TCP-like flows with our DCCP-QoS flows. We design our DCCP-QoS to be the protocol that can handle real-time applications such as IPTV or Voice-over-IP. These applications normally prefer to send out data in a constant datarate. So we begin to conduct the simulation with constant bit rate generator.

Firstly, we conduct a simulation by using DCCP-QoS flows with minimum guaranteed rate of 2.0Mbps and set the $r^m_{max} = 1$. It means that actually the three QoS flows have the same datarate modifying strategy as DCCP-TCP-like protocol besides they adjust sending rate based on changing STT. The data generators of these three kinds of DCCP in this simulation are all CBR (constant bit rate) with the rate of 2.0Mbps. Figure 6.11 shows that all three average datarates of DCCPs can achieve near 2.0Mbps. The minor differences

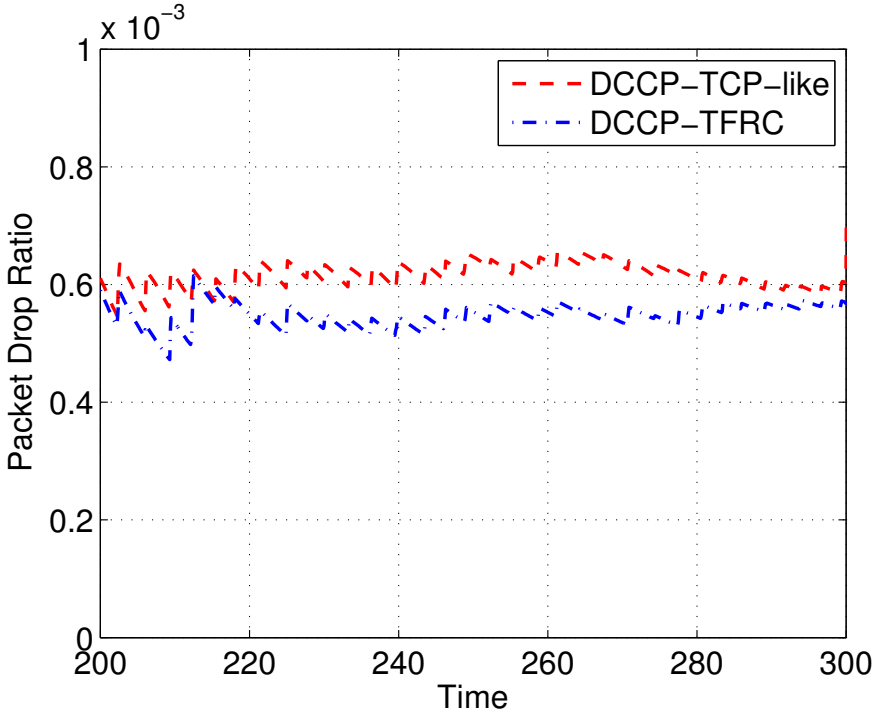

Figure 6.10. Packet Drop Ratio of DCCP-TCP-like and DCCP-TFRC flows

stem from the dropped packets in the network because we measure the datarate by the received data packets from the DCCP receivers. From Figure 6.12 the average drop ratios of DCCP-QoS and DCCP-TCP-like flows are much lower than that of DCCP-QoS flows. Because the data generator is restricted to a ”slow” 2.0Mbps, the DCCP-TFRC protocol would not slow down the sending rate on many packet-loss situations. Figure 6.13 shows that the DCCP-QoS flows can always maintain a lower loss ratio than that of DCCP-TCP-like flows.

Secondly, we conduct a simulation for DCCP-QoS with minimum guaranteed rate of 3.5Mbps and again set $r^m_{max} = 1$. And the data generators of these three kinds of DCCP are all CBR (constant bit rate) with rate of 3.5Mbps. Figure 6.14 shows that the target rate 3.5Mbps seems a little beyond the abilities of all the three kinds of DCCP flows. Figure 6.15 is quite like Figure 6.12. Figure 6.16 shows that the average drop ratios of DCCP-QoS is

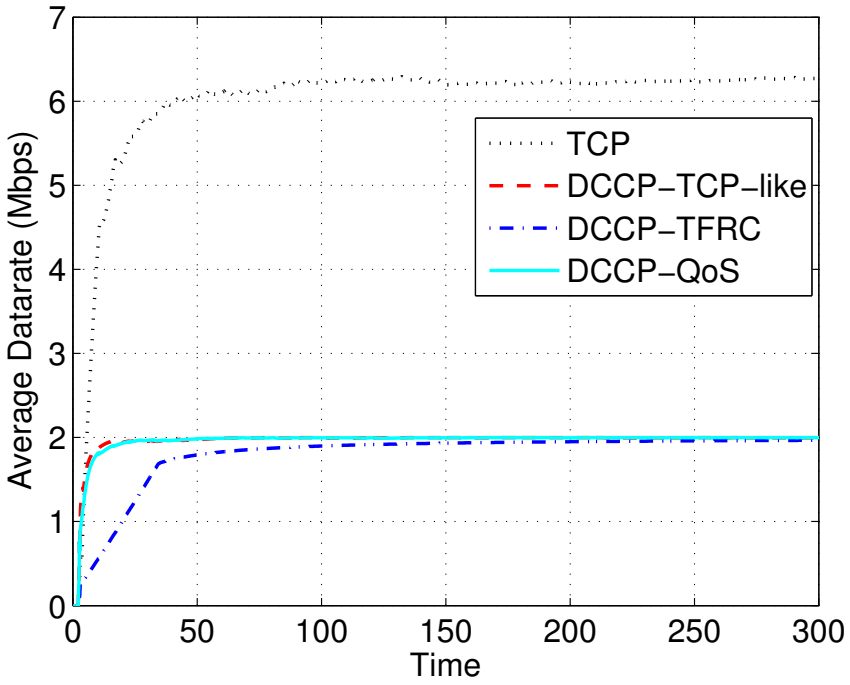

Figure 6.11. TCP, DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($r_{max} = 1$)

still a little lower than that of DCCP-QoS flows in most time period.

Again, we do the simulation for DCCP-QoS with minimum guaranteed rate of 3.5Mbps but set $r^m_{max} = 3$. Figure 6.17 shows that at this time the DCCP-QoS flows can easily reach the target rate. Figure 6.18 is nearly the same as Figure 6.15. Figure 6.19 proves that the DCCP-QoS flows can hold an average drop ratio not higher than that of DCCP-TCP-like flows with a much aggressive rate increasing strategy.

We conduct more aggressive simulation for DCCP-QoS with minimum guaranteed rate of 5.0Mbps and 10.0Mbps. And the data generators of these three kinds of DCCP are all CBR (constant bit rate) with rate of 5Mbps and 10Mbps correspondingly. Figure 6.20 and Figure 6.23 prove that the DCCP-QoS flows can still satisfy the requirement and squeeze much more bandwidth than those of DCCP-TCP-like and DCCP-TFRC. And the average

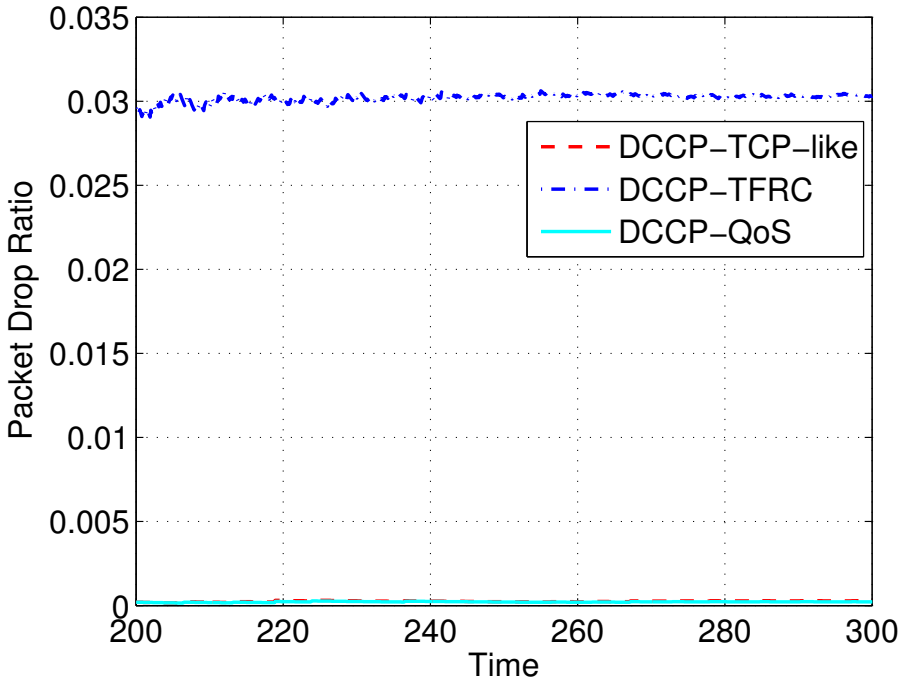

Figure 6.12. Packet Drop Ratio of DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($r_{max} = 1$) flows

drop ratios of all three kinds of flows are shown in Figure 6.21 and Figure 6.24. Figure 6.21 is like Figure 6.18 that the drop ratio of DCCP-TFRC is much higher than those of DCCP-QoS and DCCP-TCP-like. Figure 6.22 shows that the drop ratio of DCCP-QoS flows is still a little lower than that of DCCP-TCP-like. Figure 6.24 shows that DCCP-TFRC can only maintain the lowest drop ratio when the target rate is far beyond its ability.

All the previous five simulations are using CBR as data generator, this time we conduct a simulation on our DCCP-QoS with minimum guaranteed rate of 3.5Mbps but using infinite data generator FTP. Figure 6.25 shows that the average datarate of DCCP-QoS can achieve more than 5Mbps. And Figure 6.26 illustrates that the average drop ratio of DCCP-QoS flows is a little lower than that of DCCP-TCP-like flows and a little higher than that of DCCP-TFRC flows.

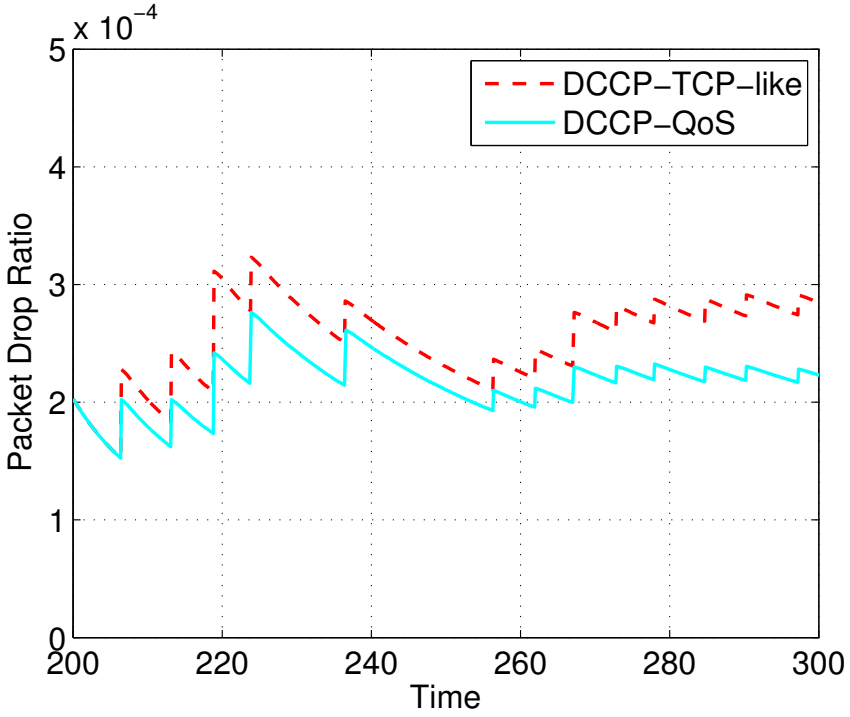

Figure 6.13. Packet Drop Ratio of DCCP-TCP-like and DCCP-QoS ($r_{max} = 1$) flows

Above simulations are all conducted on the simple topology in Figure 6.2. Next we want to see our DCCP-QoS protocol's action in a complex topology. We consider a network with 39 nodes, as shown in Figure 6.27. There are 15 persistent flows running in this network. Flow i starts at the source S_i and ends at the destination D_i. The RTTs excluding queuing delay for flows 1 to 15 are 40, 32, 32, 32, 38, 34, 38, 36, 40, 36, 44, 140, 108, 100 and 110 ms, respectively. We set $\omega = 0.3$, $\beta = 0.5$ and $\delta = 0.5$. the same as in the previous section. There are also 15 transient flows with a randomly generated set of start/stop times in the network.

In the simulations, we always run persistent flows 1-11 and all the 15 transient flows by using TCP-Reno. Firstly, we run flows 12-14 by using DCCP-TCP-like, and flows 13 and 15 by using DCCP-TFRC. Figure 6.28 shows the average throughput of flows 1, 5, 8, 9, 10,

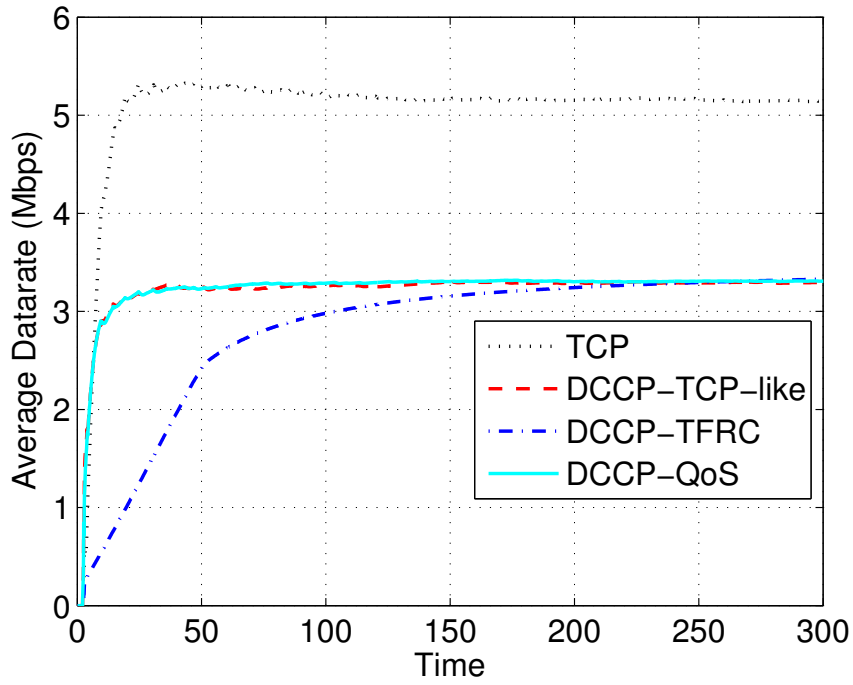

Figure 6.14. TCP, DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 3.5$ and $r_{max} = 1$)

11, 12, 13, 14 and 15. Due to the last four flows have long RTTs, the rates of flows 12-15 are around 1.2Mbps. Now we run the DCCP-QoS mechanism with $r^m_{max} = 1$ on flows 12-15. It means that our protocol sets 0 minimum guaranteed rate, these flows are NRE flows. Figure 6.29 shows that the flows using the proposed mechanism are still around 1.2Mbps, same as DCCP-TCP-like and DCCP-TFRC do. The packet drop ratio for these flows is shown in Table 6.1. From the table, we can see that the DCCP-QoS mechanism has lower packet drop ratios than those of DCCP-TCP-like and DCCP-TFRC. This shows that using STT really can quickly respond to network condition change and reduce the packet drop rate.

Now we conduct the simulations by using DCCP-QoS with $r^m_{max} = 3$ on flows 12 to 15, and set the minimum guaranteed rate as 2.5 Mbps for flows 12 and 13, 3 Mbps for flows 14 and 15. Figure 6.30 shows that the flows using the proposed DCCP-QoS can achieve their

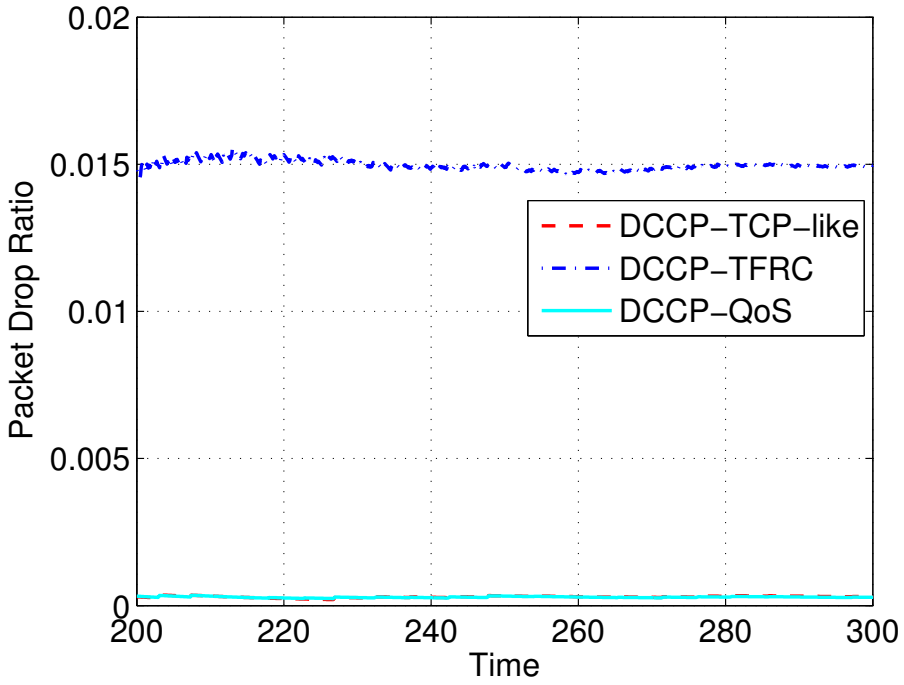

Figure 6.15. Packet Drop Ratio of DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 1$) flows

Table 6.1. Drop ratio ($\times 10^{-3}$)

	Flow 12	Flow 13	Flow 14	Flow 15
DCCP-TCP-like	4.2	-	6.75	-
DCCP-TFRC	-	5.2	-	7.93
DCCP-QoS	3.7	4.75	5.95	7.45

targeted rates and can maintain their rates stably through the heavy load period of network causing by the transient flows. Figure 6.31 shows that the packet drop ratio of these four flows. The drop ratios increase a little comparing those in case of $r^m_{max} = 1$, but they are still in the same order, close to these in DCCP-TCP-like and DCCP-TFRC. The results indicate that the derived DCCP-QoS control mechanism can really guarantee some requested rate while maintaining a small packet drop ratio.

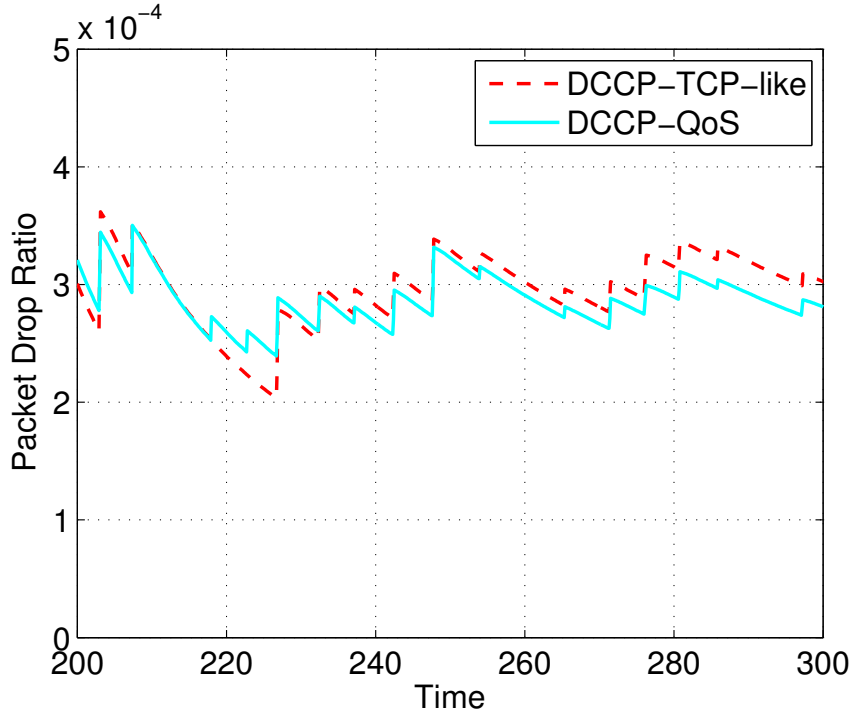

Figure 6.16. Packet Drop Ratio of DCCP-TCP-like and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 1$) flows

6.6 Summary

DCCP protocol is designed to provide congestion control for unreliable applications. It can support multiple control mechanisms and allow an application to choose the most suitable one. But the existing congestion control mechanisms of DCCP cannot support QoS features.

This chapter aims at designing an end-to-end QoS-aware CCID for DCCP to support QoS features for multimedia streaming. To achieve this goal, we redefined the congestion indicator as the STT over the threshold. We also introduced STT into the scalar function so that the control mechanism can quickly respond to the network condition changes. The proposed DCCP-QoS possesses several provable properties, including friendliness to TCP, stability, and optimality. We used fluid model to derive an upper bounded minimum guaranteed rate that can be achieved by the DCCP-QoS. The extensive simulations showed that the

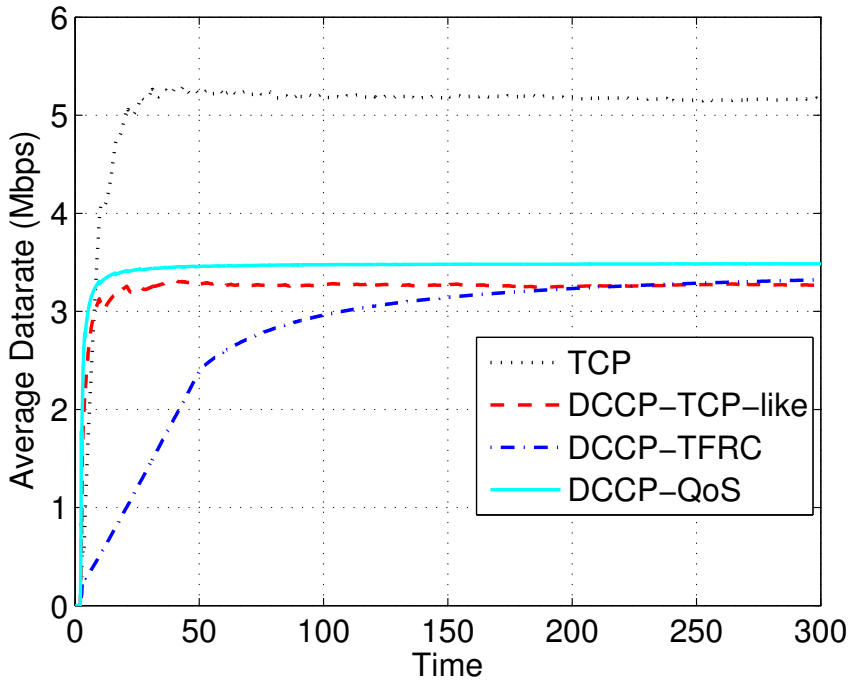

Figure 6.17. TCP, DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 3$)

proposed DCCP-QoS can provide QoS features and maintain a low packet loss ratio.

Our work can provide insights on the possible design of an optimal, end-to-end QoS transport layer protocol to support low-bandwidth Internet multimedia streaming which can utilize unreliable flows, such as live streaming media and IPTV.

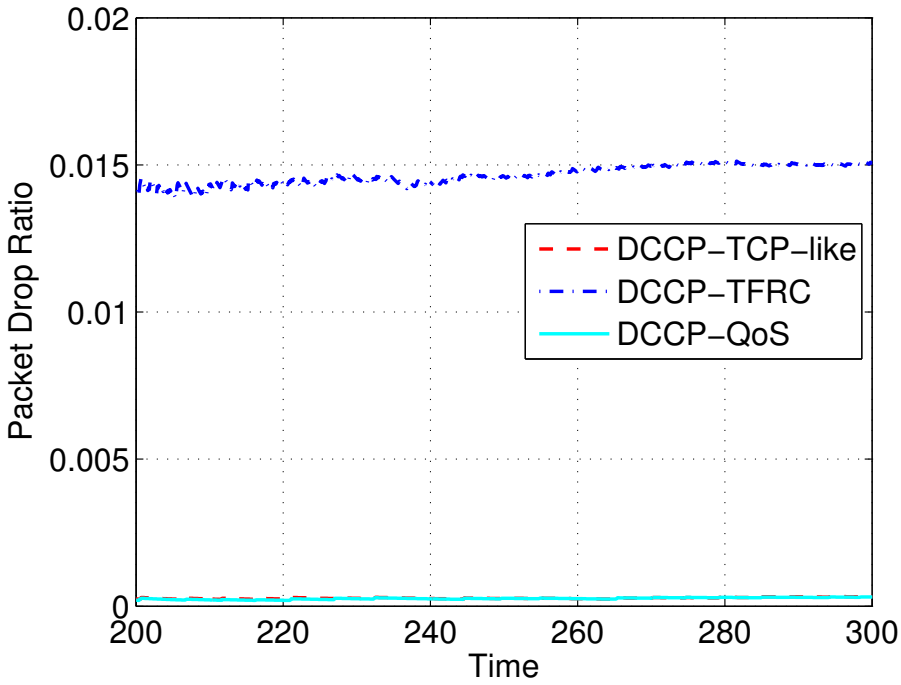

Figure 6.18. Packet Drop Ratio of DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 3$) flows

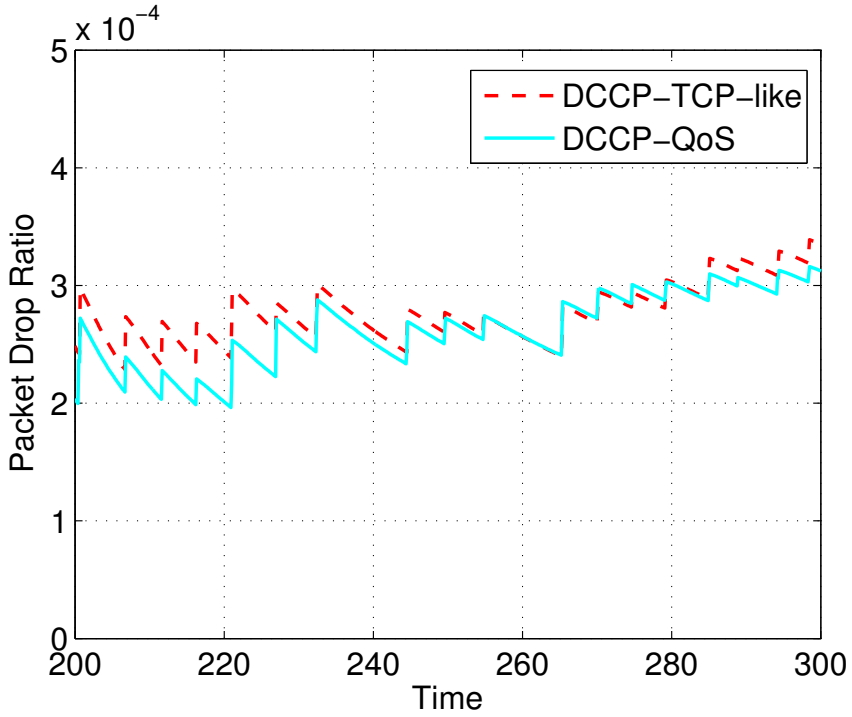

Figure 6.19. Packet Drop Ratio of DCCP-TCP-like and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 3$) flows

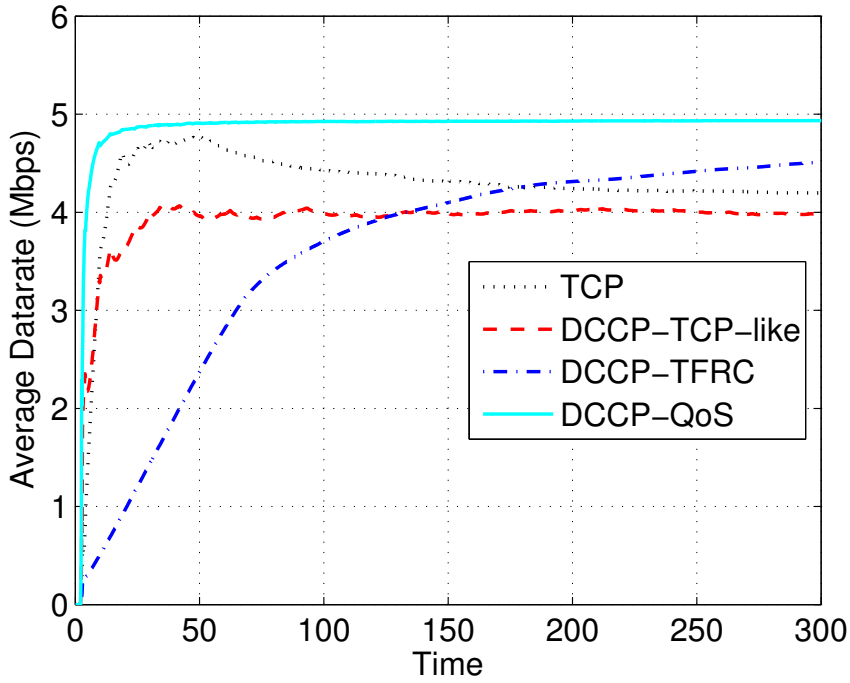

Figure 6.20. TCP, DCCP-TCP-like, DCCP-TFRC and DCCP-QoS($\theta = 5.0Mbps$ and $r_{max} = 3$)

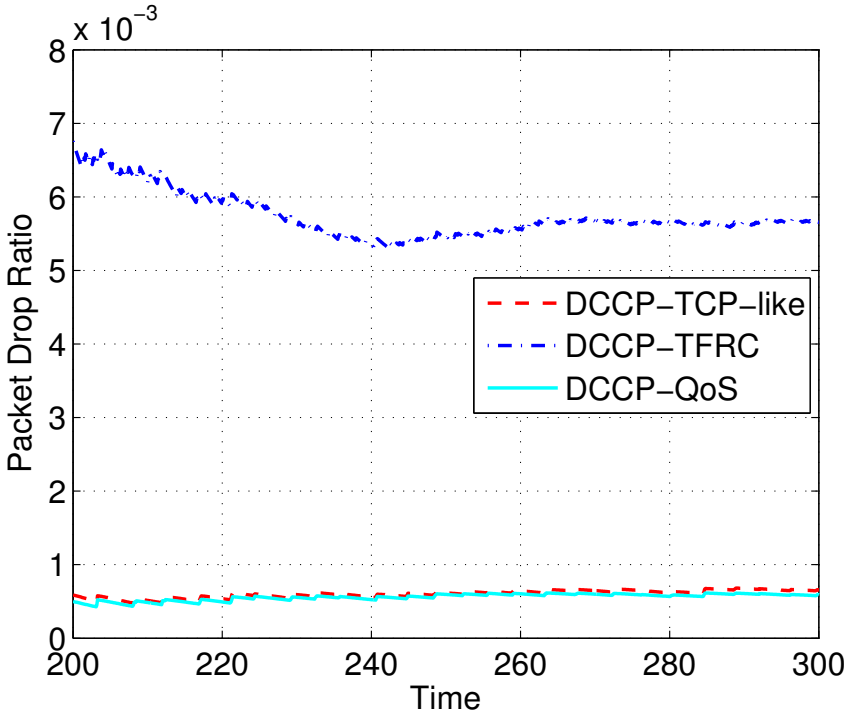

Figure 6.21. Packet Drop Ratio of DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 5.0Mbps$ and $r_{max} = 3$) flows

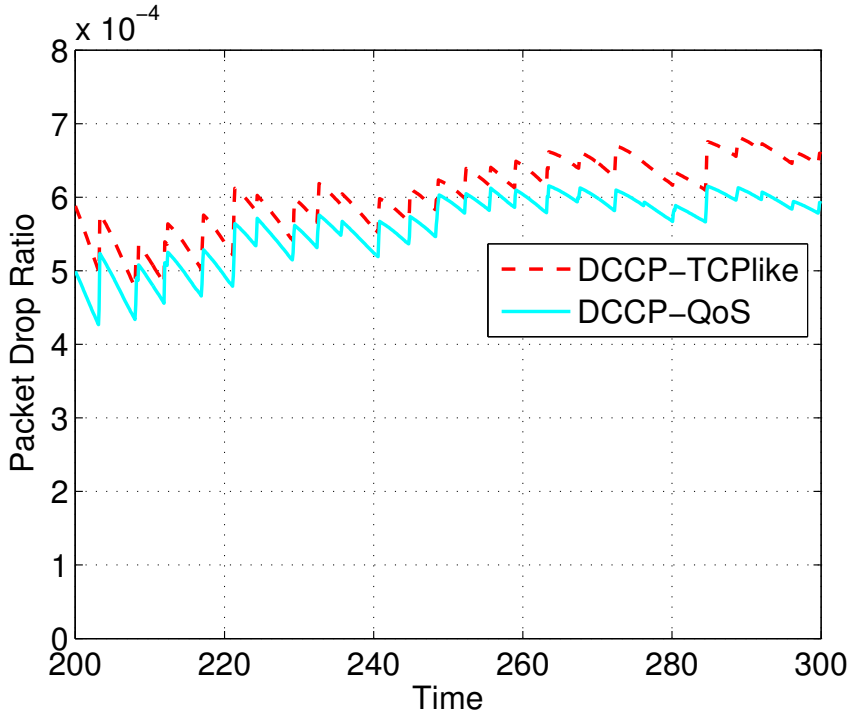

Figure 6.22. Packet Drop Ratio of DCCP-TCP-like and DCCP-QoS ($\theta = 5.0Mbps$ and $r_{max} = 3$) flows

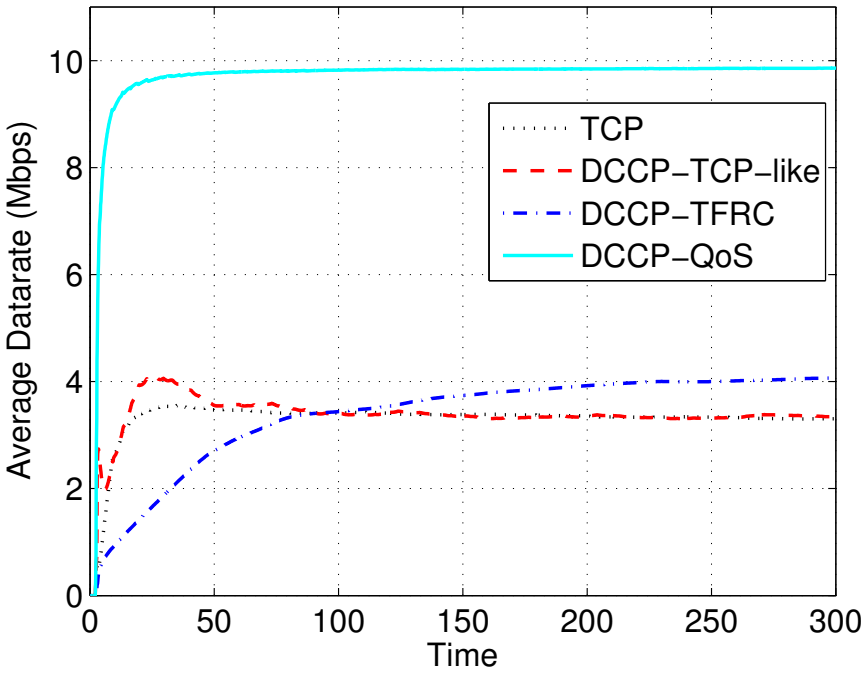

Figure 6.23. TCP, DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 10.0Mbps$ and $r_{max} = 3$)

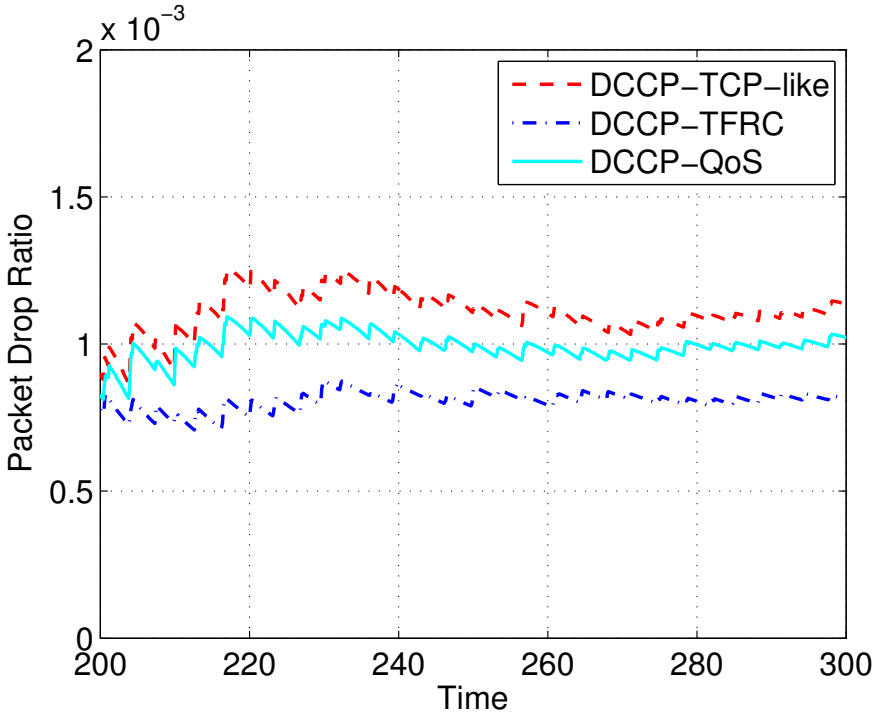

Figure 6.24. Packet Drop Ratio of DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 10.0Mbps$ and $r_{max} = 3$) flows

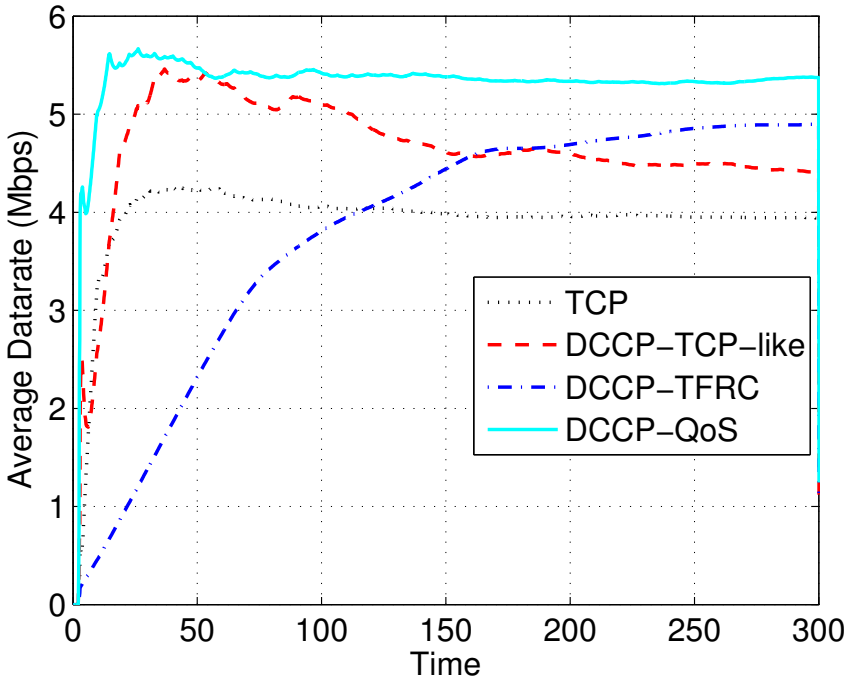

Figure 6.25. TCP, DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 3$) without CBR

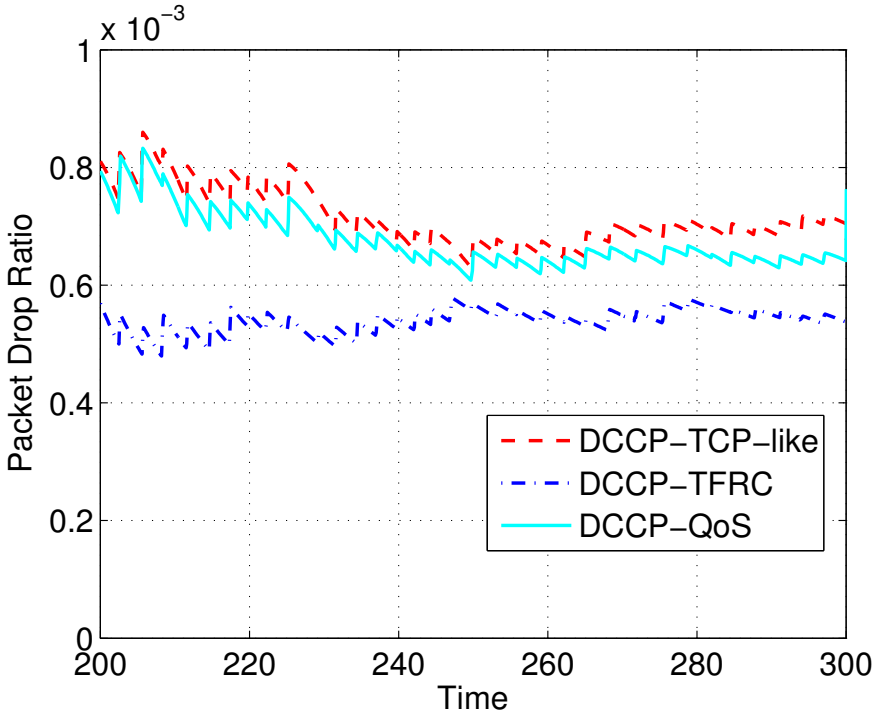

Figure 6.26. Packet Drop Ratio of DCCP-TCP-like, DCCP-TFRC and DCCP-QoS ($\theta = 3.5Mbps$ and $r_{max} = 3$) flows without CBR

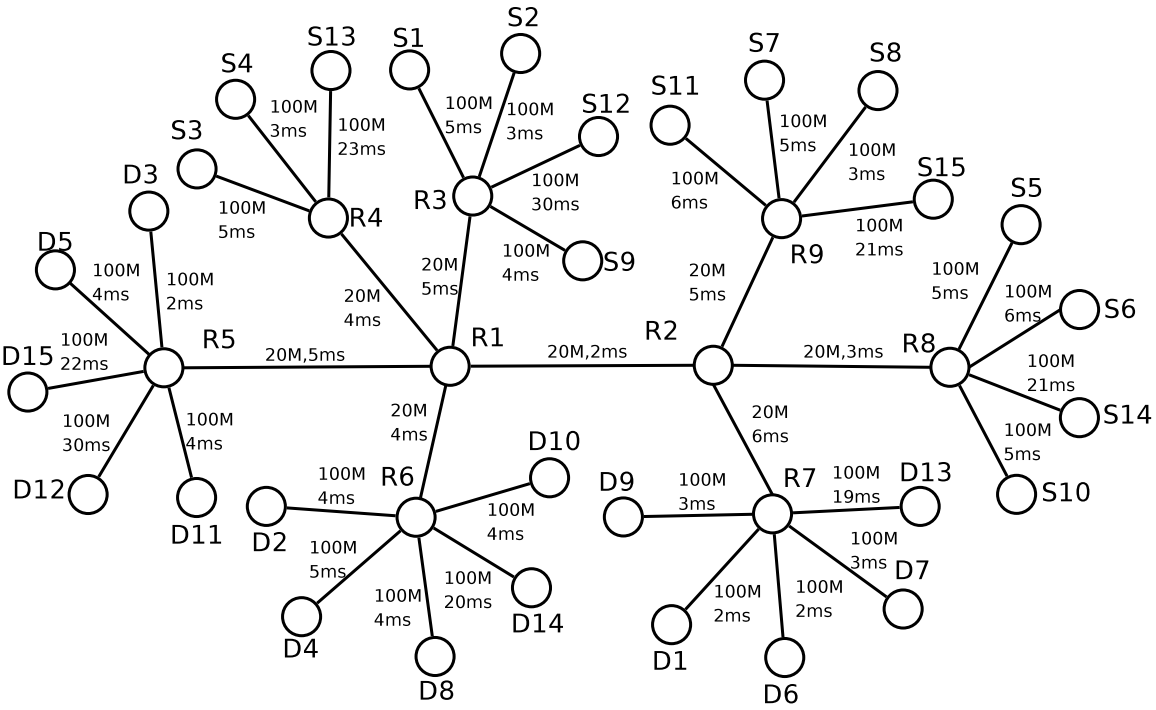

Figure 6.27. Network Topology 5.2

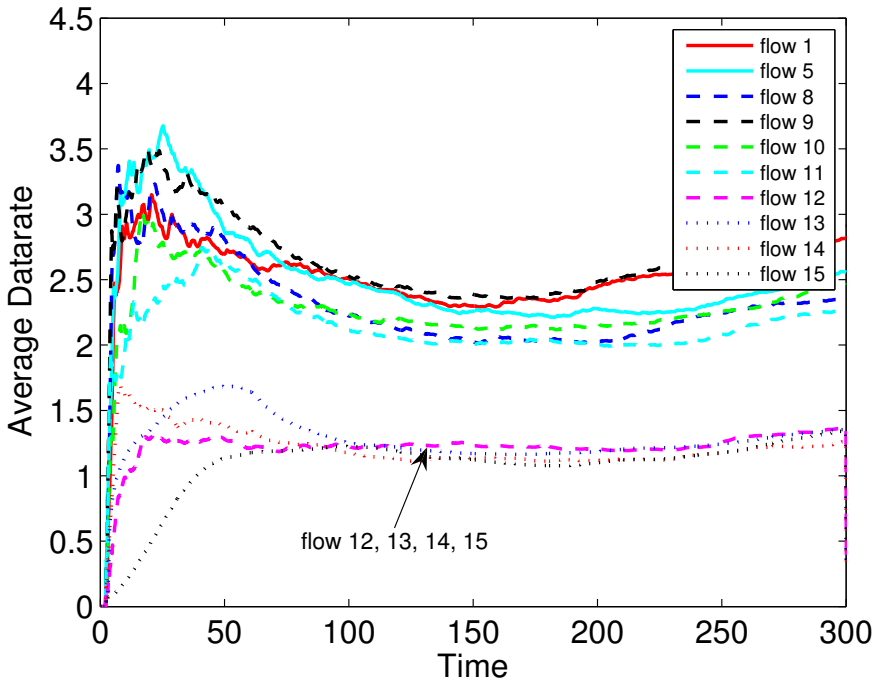

Figure 6.28. TCP, DCCP-TCP-like and DCCP-TFRC

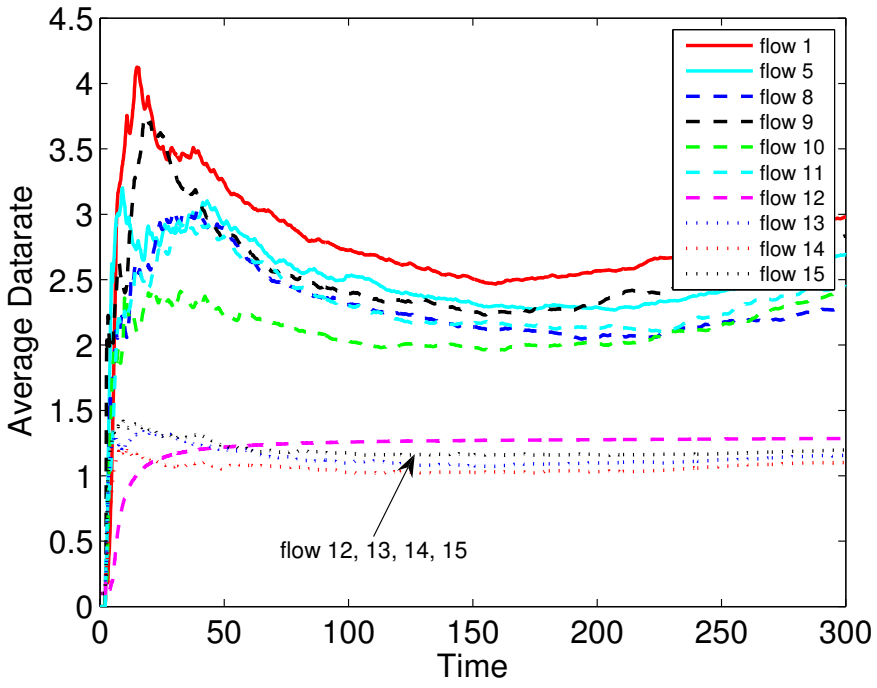

Figure 6.29. TCP and DCCP-QoS, with $r_{max} = 1$

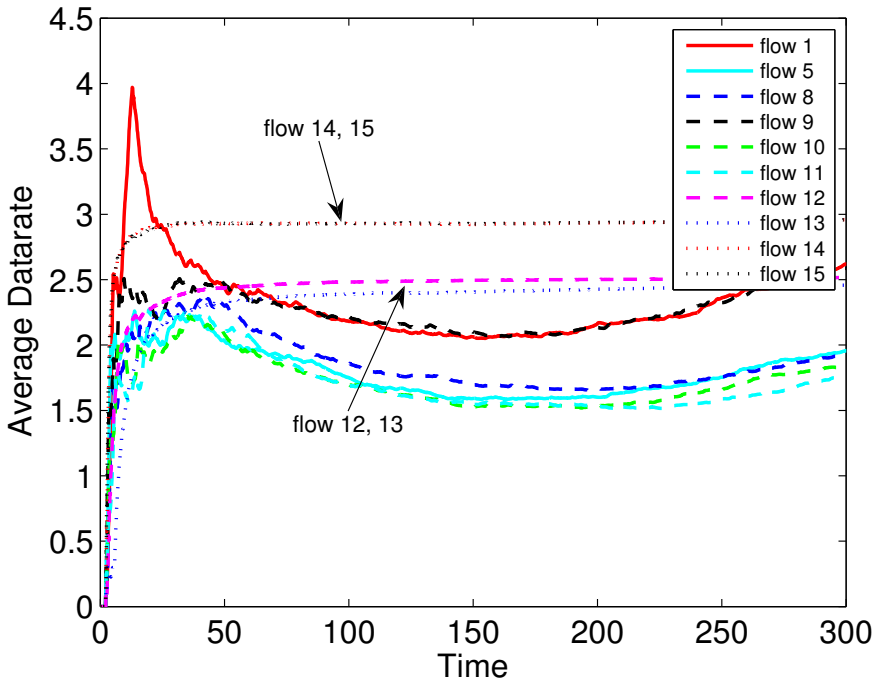

Figure 6.30. TCP and DCCP-QoS, with $r_{max} = 3$

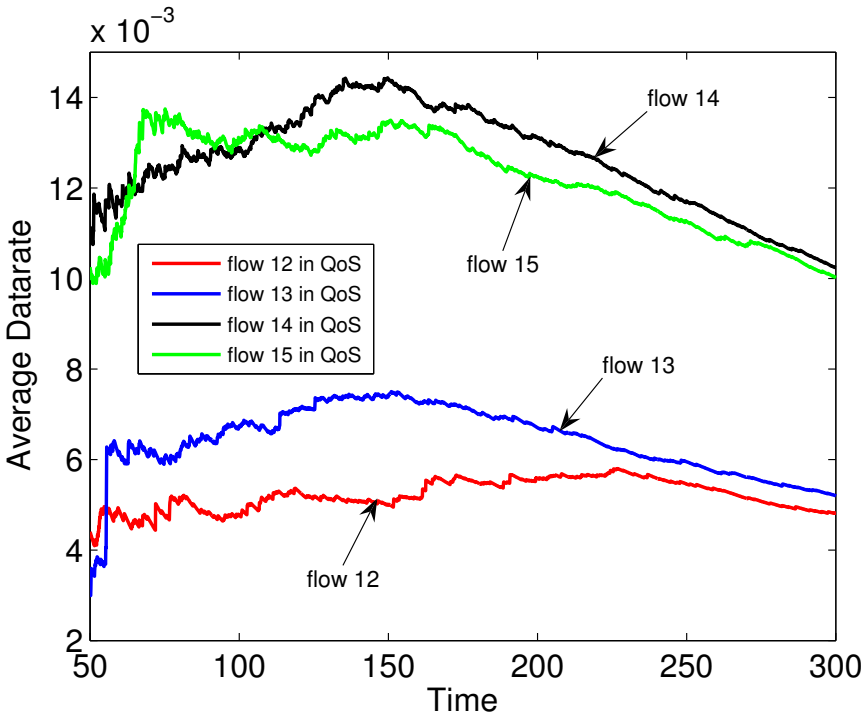

Figure 6.31. Packet drop ratio of DCCP-QoS, with $r_{max} = 3$

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Nowadays Internet, for a large part, can only provide the best effort service to the users. There are more and more real-time applications requiring Quality of Service (QoS). Many existing QoS solutions need from help of the network core nodes, which make them difficult to scale to whole Internet. Our research focuses on designing a family of optimization-based, end-to-end transport layer protocols to support various QoS requirements for the real-time applications.

In this thesis, we proposed a new optimal utility-based protocol design methodology TERSE. By reverse engineering control laws of TERSE to the TCP congestion control protocol, we derived the utility function of TCP. Based on TERSE and TCP utility function, we designed the unified congestion control protocols for both the reliable applications and unreliable applications respectively.

In this thesis, a new utility based protocol design methodology TERSE is developed and used to design new transport protocols which possess some desirable, provable properties, including fairness, stability, and optimality. The TERSE design methodology expresses the inelastic part of real-time applications as flow-level constraints, which addressed the fundamental problems of the traditional utility-based optimization protocol design methods. Second, a global utility function and the corresponding optimal control law, known as TCP control law, which maximizes the global utility, are derived. The TCP utility function is

Table 7.1. Comparison of QoS aware Congestion Control Protocols

	Reliable/Unreliable	Network Dependency	Fairness
Harks's second order optimization	Reliable	AQM	Utility proportional
Wang's application-oriented flow control	Reliable	Link price feedback	Proportional/Max-min
Resilient overlay networks	Reliable	Fully support	Not specified
Service overlay network	Reliable	Service layer agreement	Not specified
MATE	Reliable	MPLS	Not specified
AIMD DCCP	Unreliable	No need	Not specified
Fast recovery DCCP	Unreliable	No need	TCP fairness
Retransmission DCCP	Unreliable	No need	Not specified
Our TERSE	Reliable & Unreliable	No need	TCP fairness

linear in the slow start phase and approaches the logarithm form in the congestion avoidance phase. Third, based on TERSE methodology and the proposed TCP utility function, a family of QoS aware congestion control protocols are designed for reliable service. The family of QoS aware congestion control protocols can support a set of class of services including AF, MRG, UBR, and MRGUBS. The protocols are verified by NS-2 and implemented in the real Internet. Finally, a QoS aware DCCP congestion control mechanism for unreliable service is derived. It can be used for multimedia streaming. By using STT, we designed the DCCP-QoS which can increase the flow rate and in the mean time decrease the drop ratio.

At last, we compare our proposed protocols based on TERSE to other researchers' work in Table 7.1. It is clear that the family of our proposed protocols is the only solution that can provide end to end QoS aware services and is designed with TCP fairness.

7.2 Future Work

Our research in this thesis mainly focused on developing the end-to-end QoS aware transport layer protocol to support various real-time applications. In the future, we plan to develop an end-to-end QoS aware protocol specified for wireless networks. And also we plan to develop end-to-end cross-layer mechanism to support multimedia streaming.

With the rapid growing of the wireless networks, such as wireless LAN, 3G, WiMax

and etc., there are more and more demand for them to support real-time applications. It calls for the QoS support for wireless networks. Due to the special characteristics of wireless networks like lossy medium and mobility, the traditional QoS mechanisms and even TCP may not work well in them. Hence we need to develop specified end-to-end QoS aware protocol for wireless networks.

Nowadays there are many existing adaptive encoding multimedia scheme/framework that can vary the bit rate of video/sound representation, such as MPEG-4 framework. It is possible for us to employ our TERSE methodology and cross-layer design idea together to create a framework that can adapt modulate the encoding multimedia streaming bit rate based on the minimum rate guaranteed by the end-to-end QoS aware protocol. The framework should have the mechanism to estimate the throughput and the optimal strategy to choose suitable encoding scheme.

In the future, we will study these problems and develop new protocols to provide QoS for wireless networks and adaptive encoding multimedia systems.

REFERENCES

[1] T. Ahmed, A. Mehaoua, R. Boutaba, and Y. Iraqi, "Adaptive Packet Video Streaming Over IP Networks: A Cross-Layer Approach", *IEEE Journal on Selected Area in Communications*, vol. 23, no. 2, pp. 385-401, Feb. 2005.

[2] T. Alpcan, and T. Basar, "A Utility-Based Congestion Control Scheme for Internet-Style Networks with Delay", *Proceedings of IEEE INFOCOM 2003*, pp. 2039-2048, San Francisco, CA, April, 2003.

[3] C. An, and T. Q. Nguyen, "Analysis of Utility Functions for Video", *Proceedings of IEEE International Conference on Image Processing*, vol. 5, pp. 89-92, 2007.

[4] D. Andersen, H. Balakrishnan, F. Kasshoek, and R. Morris, "Resilient Overlay Networks", *Proceedings of ACM Symposium on Operating Systems Principles*, vol. 35, no. 5, pp. 131-145, December, 2001.

[5] S. Athuraliya, and S. H. Low, "Fluid models: An Empirical Validation of a Duality Model of TCP and Queue Management Algorithms", *Proceedings of the 2001 Winter Simulation Conference*, pp. 1269-1274, 2001.

[6] M. A. Azad, R. Mahmood, and T. Mehmood, "A Comparative Analysis of DCCP Variants (CCID2, CCID3), TCP and UDP for MPEG4 Video Applications", *Proceedings of International Conference on Information and Communication Technologies 2009*, pp. 40-45, August, 2009.

[7] V. Basto, and V. Freitas, "Distributed QoS Multimedia Transport", *Proceedings of International Conference on Distributed Frameworks for Multimedia Applications*, pp. 15-21, February, 2005.

[8] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", *RFC 2998, IETF*, November, 2000.

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

[10] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin "Resourse ReSerVation Protocol (RSVP) – Version 1 Functional Specification", *RFC 2205, IETF*, September 1997.

[11] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", *RFC 1633 IETF*, June, 1994.

[12] L. Brakmo, and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465-1480, 1995.

[13] L. Cai, X.S. Shen, J.W. Mark, and J. Pan, "QoS Support in Wireless/Wired Networks Using the TCP-Friendly AIMD Protocol", *IEEE Transactions on Wireless Communications*, vol. 6, no. 2, pp. 469-480, February, 2006.

[14] C. Caini, and R. Firrincieli, "TCP Hybla: a TCP Enhancement for Heterogeneous Networks", *Proceedings of International Journal of Satellite Communications and Networking*, vol. 22 , no. 5, pp. 547-566, September, 2004.

[15] Z. Cao, and E. W. Zegura, "Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme", *Proceedings of IEEE INFOCOM 1999*, vol. 2, pp. 793-801, March, 1999.

[16] K. Chandrayana, and S. Kalyanaraman, "Uncooperative Congestion Control", *Proceedings of Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 258-269, 2004.

[17] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility Maximization in Peer-to-Peer Systems", *Proceedings of Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 169-180, 2008.

[18] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures", *Proceedings of the IEEE*, vol. 95, no. 1, pp. 252-312, 2007.

[19] D. M. Chiu, "Some Observations on Fairness of Bandwidth Sharing", *Proceedings of Fifth IEEE Symposium on Computers and Communications*, pp. 125-131, July, 2000.

[20] S. H. Choi, and M. Handley, "Fairer TCP-friendly Congestion Control Protocol for Multimedia Streaming Applications", *Proceedings of International Conference On Emerging Networking Experiments And Technologies*, Article no. 54, pp. 303-309, 2007.

[21] M. Dai, Y. Zhang, and D. Loguinov, "A Unified Traffic Model for MPEG-4 and H.264 Video Traces", *IEEE Transaction on Multimedia* vol. 11, no. 5, pp. 1010-1023, August, 2009.

[22] S. Deb, A. Ganesh, and P. Key, "Resource Allocation Between Persistent and Transient Flows", *IEEE/ACM Transaction on Networking*, vol. 13, no. 2, pp. 302-315, April, 2005.

[23] S. Deb, and R. Srikant, “Congestion Control for Fair Resource Allocation in Networks With Multicast Flows”, *IEEE/ACM Transaction on Networking*, vol. 12, no. 2, pp. 274-285, 2004.

[24] S. Deb, and R. Srikant, “Rate-Based versus Queue-Based Models of Congestion Control”, *Proceedings of Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 246-257, 2004.

[25] Z. Duan, Z. Zhang, and Y. T. Hou, “Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning”, *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp 870-883, 2003.

[26] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja, “Mate: Mpls Adaptive Traffic Engineering”, *Proceedings of IEEE INFOCOM 2001*, vol. 3, pp. 1300-1309, April, 2001.

[27] M. Fazel, and M. Chiang, “Network Utility Maximization With Nonconcave Utilities Using Sum-of-Squares Method Decision and Control”, *Proceedings of 2005 and 2005 European Control Conference. CDC-ECC ’05. 44th*, pp. 1867-1874, December, 2005.

[28] J. Feng, and L. Xu, “TCP-Friendly CBR-Like Rate Control”, *Proceedings of IEEE International Conference on Network Protocols*, pp. 177-186, October 2008.

[29] S. Floyd, “Connections with Multiple Congested Gateways in packet-Switched Networks, Part 1: one-way Traffic”, *ACM Computer Communications Review*, vol. 21, no.5, pp. 30-37, October, 1991.

[30] S. Floyd, “RED: Discussions of Setting Parameters”, available on http://www.icir.org/floyd/REDparameters.txt, November 1997.

[31] S. Floyd, “Recommendation on using the ”gentle_” variant of RED”, available on http://www.icir.org/floyd/red/gentle.html March 2000.

[32] S. Floyd, T. Henderson, and A. Gurtov, “The NewReno Modification to TCP’s Fast Recovery Algorithm”, *RFC 3782, IETF*, April, 2004.

[33] S. Floyd, and J. Kempf. “IAB concerns regarding congestion control for voice traffic in the internet”. *RFC 3714, IETF*, March 2004.

[34] S. Floyd, and E. Kohler. “Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control”, *RFC 4341, IETF*, March 2006.

[35] S. Floyd, and E. Kohler. “TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant”, *RFC 4828, IETF*, April 2007.

[36] S. Floyd, E. Kohler, and J. Padhye. “Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)”, *RFC 4342*, March 2006.

[37] S. J. Golestani, and S. Bhattacharyya, “A Class of End-to-End Congestion Control Algorithms for the Internet”, *Proceedings of IEEE International conference on Network Protocols (ICNP)*, pp. 137-150, October, 1998.

[38] B. Gorkemli, and M. R. Civanlar, “ Streaming SVC Coded Video over DCCP”, *Proceedings of IEEE 15th. Signal Processing and Communications Applications* pp. 1-4, June, 2007.

[39] S. Ha, I. Rhee, and L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant”, *ACM SIGOPS Operating System Review*, vol. 42, no. 5, pp. 64-74, July, 2008.

[40] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, “Overlay TCP for Multi-path Routing and Congestion Control”, *Proceedings of ENS-INRIA ARC-TCP Workshop*, pp. 1-24, France, 2003.

[41] P. Hande, S. Zhang, and M. Chiang, "Distributed Rate Allocation for Inelastic Flows", *IEEE/ACM Transaction on Networking*, vol. 15, no. 6, pp. 1240-1253, December, 2007.

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

[43] T. Harks, and T. Poschwatta, "Congestion Control in Utility Fair Network", *Computer Networks: The International Journal of Computer and Telecommunications Networking* vol. 52, no. 15, pp. 2947-2960, 2008.

[44] T. Harks, and T. Poschwatta, "Priority Pricing in Utility Fair Networks", *Proceedings of 13th IEEE International Conference on Network Protocols (ICNP)*, pp. 311-320, November, 2005.

[45] T. Harks, and T. Poschwatta, "Utility Fair Congestion Control for Real-Time Traffic", *Proceedings of IEEE INFOCOM 2005*, vol. 4, pp. 2786-2791, March, 2005.

[46] T. R. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On Improving the Fairness of TCP Congestion Avoidance", *Proceedings of IEEE Globe Telecommunications Conference*, vol. 1, pp. 539-544, 1999.

[47] H. Hisamatsu, H. Ohsaki, and M. Murata, "Fluid-Based Analysis of a Network with DCCP Connections and RED Routers", *Proceedings of the International Symposium on Applications on Internet*, pp. 22-29, 2006.

[48] P. Hurley, J. L. Boudec, and P. Thiran, "A Note on the Fairness of Additive Increase and Multiplicative Decrease", *Proceedings of 16th International Teletraffic Congress*, pp. 467-478, 1999.

[49] A. Huszk, and S. Imre, "DCCP-based Multiple Retransmission Technique for Multimedia Streaming", *Proceedings of the 6th International Conference on Advances in*

Mobile Computing and Multimedia, pp. 21-28, 2008.

[50] iPlayer, available from http://www.bbc.co.uk/iplayer.

[51] S. Jin, L. Guo, I. Matta, and A. Bestavros, "A Spectrum of TCP-friendly Window-Based Congestion Control Algorithms", *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 341-355, June, 2003.

[52] V. Jocobson, "Congestion Avoidance and Control", *Proceedings of ACM SIGCOMM*, pp. 273-288, 1998.

[53] S. K. Jonathan, D. Towsley, and J. Kurose, "Optimization-Based Congestion Control for Multicast Communications", *IEEE Communications Magazine*, pp. 90-95, September 2002.

[54] S. Kandula, D. Katabi, B. Davie, and A. Charney, "Walking the Tightrope: Responsive yet Stable Traffic Engineering", *Proceedings of ACM SIGCOMM*, pp. 253-264, August, 2005.

[55] K. Kar, S. Sarkar, and L. Tassiulas, "A Simple Rate Control Algorithm for Max Total User Utility", *Proceedings of IEEE INFOCOM 2001*, vol. 1, pp. 133-141, 2001.

[56] F. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability", *Journal of the Operational Research Society*, vol. 49, pp. 237-252, 1998.

[57] R. Koenen, "Overview of the MPEG-4 Standard", *ISO/IEC JTC1/SC29/WG11 N4668*, available on http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm, March, 2002.

[58] E. Kohler, S. Floyd, and A. Sathiaseelan. "Faster Restart for TCP Friendly Rate Control (TFRC)", *Internet draft proposed to IETF*, July, 2007.

[59] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: Congestion Control Without Reliability", *Proceedings of ACM SIGCOMM*, pp. 27-38, August, 2006.

[60] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", *RFC 4340, IETF*, March, 2006.

[61] S. Kunniyur, and A. Srikant, "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks", *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 689-702, October, 2003.

[62] R. J. La, and V. Anantharam, "Window-Based Congestion Control with Heterogeneous Users", *Proceedings of IEEE INFOCOM 2001*, pp. 1320-1329, 2001.

[63] R. J. La, and V. Anantharam, "Utility-Based Rate Control in the Internet for Elastic Traffic", *IEEE Transactions on Networking*,vol. 10, no. 2, pp. 272-286, April, 2002.

[64] C. M. Lagoa, H. Che, and B. A. Movsichoff,"Adaptive Control Algorithms for Decentralized Optimal Traffic Engineering in the Internet", *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 415-428, June, 2004.

[65] Y. C. Lai, "DCCP Congestion Control with Virtual Recovery to Achieve TCP-Fairness", *IEEE Communications Letters*, vol. 12, no. 1, pp. 50-52, January, 2008.

[66] T. V. Lakshman, and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth Delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, vol. 5, no.3, pp. 336-350, June, 1997.

[67] J. Lee, R. Mazumdar, and M. Shroff, "Non-convexity Issues for Internet Rate Control with Multi-class Services: Stability and Optimality", *Proceedings of IEEE INFOCOM 2004*, vol. 1, pp. 1-34, 2004

[68] J. Lee, A. Tang, J. Huang, M. Chiang, and A. R. Calderbank, "Reverse-Engineering MAC: A Non-Cooperative Game Model", *IEEE Journal on Selected Areas in Communications* vol. 25, no. 6, pp. 1135-1147, 2007.

[69] R. Li, L. Ying, A. Eryilmaz, and N. B. Shroff, "A Unified Approach to Optimizing Performance in Networks Serving Heterogeneous Flows", *Proceedings of IEEE INFOCOM 2009*, pp. 253-261, 2009.

[70] Y. Li, M. Chiang, and A. R. Calderbank, "Congestion Control in Networks with Delay Sensitive Traffic", *Proceedings of IEEE Global Communications Conference 2007*, pp. 2746-2751, 2007.

[71] S. Linck, E. Mory, J. Bourgeois, E. Dedu, and F. Spies, "Video quality estimation of DCCP streaming over wireless networks", *Proceedings of 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* , pp. 405-412, February, 2006.

[72] E. Lochin, G. Jourjon, and L. Dairaine, " Study and Enhancement of DCCP over DiffServ Assured Forwarding Class", *Proceedings of Fourth European Conference on Universal Multiservice Networks*, pp. 250-262, February, 2007.

[73] D. Loguinov, and H. Radha, "End-to-End Rate-Based Congestion Control: Convergence Properties and Scalability Analysis", *IEEE/ACM Transaction on Networking*, vol. 11, no. 4, pp. 564-577, August, 2003.

[74] S. H. Low, "A Duality Model of TCP and Queue Managemnet Algorithms", *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525-536, 2003.

[75] S. H. Low, and D. Lapsley,"Optimization flow control, I: Basic algorithm and convergence", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861-874, 1999.

[76] S. H. Low, F. Paganini, and J. C. Doyle, "Internet Congestion Control", *IEEE Control System Magazine*, vol. 22, no. 1, pp. 28-43, February, 2002.

[77] K. Ma, R. Mazumdar, and J. Luo, "On the Performance of Primal/Dual Schemes for Congestion Control in Networks with Dynamic Flows", *Proceedings of IEEE INFOCOM 2008*, pp. 960-967, 2008.

[78] L. Massoulie, and J. Robert. "Bandwidth Sharing: Objectives and Algorithms". *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 320-328, June, 2002.

[79] K. Miller, and T. Harks, "Utility Max-Min Fair Congestion Control with Time-Varying Delays", *Proceedings of IEEE INFOCOM 2008*, pp. 968-976, 2008.

[80] V. Misra, W. Gong, and D. Towsley, "Fluid-Based Analysis of a Network of AQM routers supporting TCP Flows with Application to RED", *Proceedings of ACM SIGCOMM*, pp. 151-160, August, 2000.

[81] J. Mo, and J. Warland, "Fair End-to-End Window-Based Congestion Control". *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556-567, October, 2000.

[82] B. A. Movsichoff, C. M. Lagoa, and H. Che,"Decentralized optimal traffic engineering in connectionless networks", *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 293-303, February, 2005.

[83] B. A. Movsichoff, C. M. Lagoa, and H. Che, "End-to-End Optimal Algorithms for Integrated QoS, Traffic Engineering, and Failure Recovery", *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 813-823, August, 2007

[84] Network Simulator 2, available on http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[85] F. Nivor, “Experimental Study of DCCP for Multimedia Applications”, *Proceedings of International Conference On Emerging Networking Experiments And Technologies*, pp. 272-273, 2005.

[86] N. Ozbek, B. Gorkemli, A. M. Tekalp, and T. Tunali, “Adaptive Streaming of Scalable Stereoscopic Video Over DCCP”, *Proceedings of IEEE International Conference on Image Processing, 2007*, vol. 6, pp. 489-492, September, 2007.

[87] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. “Modeling TCP Throughput: A Simple Model and its Empirical Validation”, *Proceedings of ACM SIGCOMM*, pp. 303-314, October, 1998.

[88] F. Paganini Z. Wang, J. Doyle, and S. H. Low, “Congestion Control for High Performance, Stability and Fairness in General Networks”, *IEEE/ACM Transactions on Networking*, vol. 13, no. 11, pp. 43-56, February, 2005.

[89] J. Papandriopoulos, S. Dey, and J. Evans, “ Optimal and Distributed Protocols for Cross-Layer Design of Physical and Transport Layers in MANETs”, *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp.1392-1405, 2008.

[90] C. Perkins, and L. Gharai, “RTP and the Datagram Congestion Control Protocol”, *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 1521-1524, July, 2006.

[91] J. Pongsajapan, and S. H. Low, “Reverse Engineering TCP/IP-like Networks Using Delay-Sensitive Utility Functions”, *Proceedings of IEEE INFOCOM*, pp. 418-426, May, 2007.

[92] J. Postel, ”Transmission Control Protocol”, *RFC 793, IETF*, September, 1981.

[93] J. Postel, “User Datagram Protocol”, *RFC 768, IETF*, August, 1980.

[94] PPlive, available on http://www.pplive.com .

[95] M. C. Ranasingha, M. N. Murthi, and K. Premaratne, "A Congestion Control Method to Support Coordinated Bandwidth Allocation", *Proceedings of 40th Conference on Information Sciences and Systems*, pp. 597-602, 2006.

[96] P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear Instabilities in TCP-RED", *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 1079-1092, December, 2004.

[97] P. Ranjan, R. J. La, and E. H. Abed, "Global Stability Conditions for Rate Control With Arbitrary Communication Delays", *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 94-107, February, 2006.

[98] E. Rosen, A. Visvanathan, and R. Callon, "Multiprotocol Label Switching Architecture", *RFC 3031, IETF*, January, 2001.

[99] V. Rosolen, O. Bonaventure, and G. Leduc, "A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic", *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 3, pp. 23-43, July, 1999.

[100] V. Rosolen, O. Bonaventure, and G. Leduc, "Impact of cell discard strategies on TCP/IP in ATM UBR networks", *Proceedings of the 6th Workshop on Performance Modelling and Evaluation of ATM Networks (IFIP ATM'98)*, pp. 82-91, Ilkley, UK, July, 1998.

[101] A. Sathiaseelan, and G. Fairhurst, "Performance of VoIP using DCCP over a DVB-RCS Satellite Network", *Proceedings of IEEE International Conference on Communications*, pp. 13-18, June, 2007.

[102] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacboson, "RTP: A Transport Protocol for Real-Time Applications", *RFC 3550, IETF*, July, 2003.

[103] N. Seddigh, B. Nandy, and P. Pieda. “Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network”, *Proceedings of IEEE Global Communications Conference*, pp. 6-12, Rio De Janeiro, Brazil, December 1999.

[104] S. Shakkottai, and R. Srikant, “Deterministic Fluid Models of Congestion Control in High-speed Networks”, *Proceedings of the 2001 Winter Simulation Conference*, pp. 1275-1281, 2001.

[105] B. Shao, M. Mattavelli, D. Renzi, M.T. Andrade, S. Battista, S. Keller, G. Ciobanu, and P. Carvalho, “A Multimedia Terminal for Adaptation and End-to-End QoS Control”, *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 433-436, 2008.

[106] S. Shenker, “Fundamental Design Issues for the Future Internet”, *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176-1188, 1995

[107] W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, *RFC 2001, IETF*, January, 1997.

[108] J. S. Stidham, “Pricing and Congestion Management in a Network with Heterogeneous Users”, *IEEE Transactions on Automatic Control*, vol. 49, no. 6, pp. 976-981, June 2004.

[109] R. Srikant, “Models and Methords for Analyzing Internet Congestion Control Algorithms”, *In Advances in Communication Control Networks in the series ”Lecture Notes in Control and Information Sciences(LCNCIS)”, C.T. Abdallah, J. Chiasson and S. Tarbouriech (eds.)*, Springer-Verlag, 2004.

[110] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, “Over QoS: Offering Internet QoS Using Overlays”, *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 11-16, 2002.

[111] J. Takahashi, H. Tode, and K. Murakami, "QoS Enhancement Methods for MPEG Video Transmission on the Internet", *Proceedings of IEEE Symposium on Computers and Communications*, pp. 106-113, July, 2002.

[112] A. Tang, J. Wang, and S. H. Low, "Counter-Intuitive Throughput Behaviors in Networks Under End-to-End Control", *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 355-368, April, 2006.

[113] A. Tang, J. Wang, S. H. Low, and M. Chiang, "Equilibrium of Heterogeneous Congestion Control: Existence and Uniqueness", *IEEE Transaction on Networking*, vol. 15, no. 4, pp. 824-837, August, 2007.

[114] A. Tang, D. Wei, S. H. Low, and M. Chiang, "Heterogeneous Congestion Control: Efficiency, Fairness and Design", *Proceedings of 14th IEEE International Conference on Network Protocols* pp. 127-136, 2006.

[115] Q. T. Tong, H. Koga, K. Iida, and Y. Sakai, "Improved TCP Fairness in DCCP Flow Control for Bursty Real-Time Applications", *Proceedings of IEICE/IEEK/IEEE 1st International Conference on Communications and Electronics*, pp. 66-71, October, 2006.

[116] H. V. Balan, L. Eggert, S. Niccolini, and M. Brunner, "An Experimental Evaluation of Voice Quality over the Datagram Congestion Control Protocol" *Proceedings of IEEE INFOCOM 2007*, pp. 2009-2017, May, 2007.

[117] T. Voice, "A Global Stability Result For Primal-Dual Congestion Control Algorithms With Routing", *ACM SIGCOMM Computer Communications Review*, vol. 34, no. 3, pp. 35-41, July, 2004.

[118] M. Vojnovic, J. Boudec, and C. Boutremans, "Global Fairness of Additive-increase and Multiplicative-decrease with Heterogeneous Round-trip Times", *Proceedings of IEEE INFOCOM 2000*, vol. 3, pp. 1303-1312, March, 2000.

[119] J. V. Velthoven, B. V. Houdt, and C. Blondia, "Performance of Constant Quality Video Applications Using the DCCP Transport Protocol", *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pp. 511-512, 2006.

[120] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study", *Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 908-915, New York, 2004.

[121] C. Wang , H. Wang , Y. Lin, S. Chen, and C. Wu, "Charge-Based Low Priority Congestion Control", *Proceedings of IEEE International Conference on Comunication Technology Proceedings*, pp. 550-553, 2008.

[122] J. Wang, and Q. Wu, "Porting VoIP Applications to DCCP", *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, pp. 49-54, 2008.

[123] W. H. Wang, M. Palaniswami, and S. H. Low, "Application-Oriented Flow Control: Fundamentals, Algorithms and Fairness", *IEEE/ACM Transaction on Networking*, vol. 14, no. 6, pp. 1282-1291, December, 2006.

[124] X. Wang, and H. Schulzrinne, "Pricing Network Resources for Adaptive Applications", *IEEE/ACM Transactions on Networking*, vol. 14, no. 3, pp. 506-519, June, 2006.

[125] A. Watanabe, T. Takien, M. Murata, and H. Miyahara "Comparisons of QOS from User's Perspective; Which Provides Better Utility to Users, Best Effort or Reservation-Based Services", *Communications, APCC/OECC '99*, vol. 2, pp. 1170-1175, 1999.

[126] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246-1259, December, 2006.

[127] J. Widmer, R. Denda, and M. Mauve, "A Survey on TCP-friendly Congestion Control", *IEEE Network*, vol. 15, no. 3, pp. 28-37, May 2001.

[128] B. Wydrowski, and M. Zukerman, "MaxNet: A Congestion Control Architecture for Maxmin Fairness," *IEEE Communications Letters*, vol. 6, pp. 512-514, Nov. 2002.

[129] C. Xu, J. Liu, and C. Zhao, " Performance analysis of transmitting H.263 over DCCP", *Proceedings of IEEE VLSI Design and Video Technology*, pp. 328-331, May, 2005.

[130] L. Ye, Z. Wang, H. Che, H. B.C. Chan, and C. M. Lagoa, "Utility Function of TCP", *Elsevier Computer Comunications*, vol. 32, no. 5, pp. 800-805, 2009.

[131] N. Zhang, G. M. Dimirovski, Y. Jing, and S. Zhang, "AQM Algorithm Based on Kelly's Scheme Using Sliding Mode Control", *Proceedings of American Control Conference*, pp. 1575-1579, June, 2009.

[132] H. Zhou, "STT-vegas, A Simple Single-Trip Time Based Modification of Vegas", *Proceedings of IEEE Consumer Communications and Networking Conference*, vol. 1, pp. 453-457, Las Vegas, January, 2006.