2005

Qos In Cognitive Packet Networks: Adaptive Routing, Flow And Congestion Control

Pu Su

University of Central Florida

QoS in Cognitive Packet Networks: Adaptive Routing, Flow and Congestion Control

by

Pu Su
B.S., Tsinghua University, 2001
M.S., University of Central Florida, 2003

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2005

Major Professor:
Erol Gelenbe

Abstract

With the emergence of various applications that have different Quality of Service (QoS) requirements, the capability of a network to support QoS becomes more and more important and necessary. This dissertation explores QoS in Cognitive Packet Networks (CPN) by using adaptive routing, flow and congestion control.

We present a detailed description and analysis of our proposed routing algorithms based on single and multiple QoS constraints. An online estimation of packet loss rate over a path is introduced. We implement and evaluate the adaptive routing scheme in an experimental CPN test-bed. Our experiments support our claims that the users can achieve their desired best-effort QoS through this routing scheme.

We also propose a QoS-based flow and congestion control scheme that is built in the transport layer and specially designed to work with CPN to support users' QoS while remaining friendly to TCP. Theoretical models and experimental analysis are presented. Finally we experimentally demonstrate that the proposed flow and congestion control scheme can effectively control the input flows, react to the congestion and work with our proposed adaptive routing scheme to achieve users' QoS.

To my parents

Acknowledgments

I am deeply grateful to my advisor, Professor Erol Gelenbe, for giving me the opportunity to do this research. His support, encouragement, and advice have helped me with various aspects of my Ph.D work. This dissertation would not have been possible without his invaluable guidance.

I would like to acknowledge my colleagues, Michael Gellman, Ricardo Lent, Arturo Nunez, Peixiang Liu and Feng Lv for sharing their wisdom on conducting research. In particular, I am very thankful for Ben Douglass for his friendly help on revising my dissertation.

I benefited from Robert Traub and Donald Harper for their assistance with all types of technical support.

Special thanks go to my thesis committee members, Professor Mostafa Bassiouni, Professor Michael Georgiopoulos and Professor Parveen Wahid.

I would also like to thank my dear husband Zhaoming Zhu for his love, patience and care. He endured long periods of separation and uncertainty to support what I like. He and our adorable baby are constant sources of inspiration and motivation in my life.

This thesis is dedicated to my parents, who affect me with their unfailing enthusiasm and optimism about life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Research Statement

Currently the Internet offers a point-to-point delivery service, which is based on the "best-effort" delivery model. In this model, data will be delivered to its destination as quickly as possible, but it has no commitments in regards to QoS parameters, such as bandwidth or delay. Examples of best-effort services are Internet Protocol (IP), User Datagram Protocol (UDP), etc. Using protocols such as Transmission Control Protocol (TCP), the highest guarantee the network provides is just reliable data delivery. This is adequate for traditional data applications like ftp and telnet, but it is inadequate for those applications that require timeliness. For example, many multimedia applications need to communicate in real-time. They are sensitive to the delay and jitter, and can not afford the unacceptable delay introduced by the retransmission of TCP. Such real-time multimedia streaming traffic is increasing very rapidly on the Internet and is expected to occupy a significant portion of the total Internet bandwidth.

In other words, different applications have different Quality of Service (QoS) requirements, such as delay, jitter, bandwidth and loss probability, which urges the Internet Service Providers (ISPs) to provide different treatment to different users. The notion of QoS is proposed to capture the performance contract between the service provider and the user applications. Basically, providing QoS requires cooperation of many network control mechanisms, such as routing, call admission, flow and congestion control. Much work has addressed routing schemes that try to achieve desired QoS objectives. Many researchers also proposed control techniques that offer QoS guarantees to traffic generated by specific applications. Among these works, a Cognitive Packet Network (CPN) was presented to build the intelligent capabilities for routing in the packets rather than the nodes or protocols. This adaptive source routing has the potential to support QoS. However, the problems of how to acquire accurate and reliable information on the network and how to satisfy users' QoS constraints are still challenges. Therefore, in this dissertation, we study how to make adaptive routing decisions based on different QoS constraints, such as delay and loss in the CPN framework.

On the other hand, QoS is really only noticed when the best-effort service encounters congestion. That is, when the network resource is unlimited and no any congestion happens, best-effort service will certainly satisfy the users' QoS demands. However, network resources are limited and congestion is unavoidable. Fluctuations of network conditions combined with the inability of handling congestion often affect the users' desired QoS. Thus, efficient end-to-end flow and congestion control is important and necessary to limit the input flows, to respond to the network conditions and to work with the routing scheme. In particular,

real-time audio and video applications are susceptible to changes in the transmission characteristics of networks and rely on best-effort UDP. Lack of flow and congestion control in UDP jeopardizes the QoS of those competing TCP applications and can even result in congestion collapse. Therefore, we propose a QoS-based Rate Control (QRC) mechanism built in UDP to handle congestion while supporting the users' QoS. This is the second part in my dissertation work.

1.2 Contributions

This dissertation has the following contributions.

1. We introduce a method to estimate the forward packet loss ratio in real-time environment and we suggest an approach to construct composite goal functions which combine multiple QoS constraints, namely packet loss and delay, into a single goal function, which is used to complete QoS-based adaptive routing in CPN. We present a detailed description and analysis of the proposed algorithms.

2. We experimentally implement the different goal functions based on different QoS goals and we investigate the performance of the adaptive routing scheme based on users' single and multiple QoS constraints in the CPN framework. The results of our experiments demonstrated that such an adaptive routing scheme can achieve the better QoS than the most common best-effort service.

3. We propose a QoS-based Rate Control (QRC) scheme to support the user's QoS for the real-time applications. It is an end-to-end flow and congestion control scheme for unicast applications. We design and discuss two QRC schemes based on loss rate and jitter respectively. The loss-based scheme is oriented for real-time applications with the demand of high throughput and loss rate bound. The jitter-based scheme is for jitter-sensitive applications. Meanwhile, we consider the QRC's friendliness to TCP and claim that our proposed QRC can support users' QoS while being friendly to TCP based on our experimental results.

1.3 Outline of the Dissertation

The rest of this dissertation is organized as follows.

In Chapter 2 we introduce some basic concepts which will be used in the later chapters. We also conduct a general survey of mechanisms supporting QoS, such as ATM, Interv, Differv, MPLS, etc., and review the existing end-to-end flow and congestion control algorithms.

We then summarize the CPN architecture, including its packets and random neural network with reinforcement learning algorithm in Chapter 3. A detailed algorithm of estimating online packet loss rate is presented, too. This estimation is the basis of our thesis work. We use the packet loss rate to reflect the users' QoS in the network level, and control the sending rate as a congestion signal. Our proposed use of QoS constraints to explore network and

QoS-based flow and congestion control are both based on the original CPN framework and this online estimation of packet loss rate.

In Chapter 4, we formulate the QoS constraints and propose different reward functions for different QoS requirements. These reward functions are used in reinforcement learning algorithm of CPN and make routing decisions based on single or multiple metrics. We experimentally evaluate the performance of the QoS routing scheme in CPN and analyze the delay, loss rate, and jitter performance achieved by users based on different routing policies.

After that, we investigate the flow and congestion problem in Chapter 5. In design of a flow and congestion algorithm suitable for real-time applications, we take into account the friendliness with TCP and the users' QoS. We propose a QoS-based flow and congestion control for real-time applications, where we use a rate control equation based on the AIMD model to control the sending rate. The implementation details of proposed QoS-based rate control (QRC) scheme are presented in Chapter 6. We evaluate and discuss its fairness to TCP. We also observe delay, loss rate, and jitter performance respectively.

Finally, we give a summary of the dissertation and discuss a few future research issues in Chapter 7.

CHAPTER 2

PRELIMINARY

2.1 Basic Concepts

The notion of Quality of Service (QoS) is proposed to capture the qualitatively or quantitatively defined performance contract between the service provider and user applications. Thus, QoS is actually a multidimensional objective function (defined across multiple service classes) including some QoS parameters, such as end-to-end delay, jitter, loss ratio, etc. [Sch96]. We will define them and other terms that are frequently used throughout the dissertation in the following sections.

- End-to-end delay D: The one-way end-to-end delay D is the time of a packet traversing the path from the source to the destination. Formally, given a packet that leaves the source at time $t(s)$ and arrives at the destination at time $t(d)$, $D = t(d) - t(s)$. Jitter J, also called delay variance, is the difference in end-to-end delay between successive packets. $J = D_{i+1} - D_i$. The end-to-end delay and jitter are both important QoS parameters for real-time applications.

- Round Trip Time (RTT): The RTT is the time for a packet traversing the path from the source to the destination and back. Formally, given a packet that leaves the source at time $t(s)$, and arrives at the destination at time $t(d)$, and the packet's acknowledgement packet arrives at the source at time $t(a)$, $RTT = t(a) - t(s)$. RTT is useful for reflect the network condition. In particular, due to synchronization problems, it is usually difficult to determine end-to-end delay. Thus, RTT becomes an important alternate measurement parameter.

- Loss rate L: The loss rate L denotes the ratio of lost packets to sent packets. We use loss rate, loss, loss probability refer to L throughout the dissertation. We use L_f denote the forward one-way loss rate from source to destination and L_b denote the backward one-way loss rate from destination to source.

- Throughput: The throughput parameter measures the number of bits of user data transferred per second, measured over some time interval, from end-to-end. In contrast with throughput, *goodput* measures the number of data received at the destination over a period, and this term is more often used in ATM to consider the loss of some cells.

2.2 QoS with ATM, Intserv, Diffserv and MPLS

Currently there are many mechanisms that support QoS in the networks, such as ATM, Integrated Service, Differentiated Service, etc. These technologies aim to supply guaranteed

service for specific applications and meanwhile maximize the usage of the network bandwidth. In recent years, QoS has become an important consideration in network implementation and application.

Asynchronous Transfer Mode (ATM) networks were the first networks to support QoS. It appeared in the early 1990s and its primary purpose was to provide a high-speed, low-delay, low jitter multiplexing and switching environment that can virtually support any type of traffic, such as voice, data or video applications. After ATM, more and more techniques were developed to support QoS with or without ATM. Due to the popularity and dominance of TCP/IP, there is a lot of research about IP over ATM. Besides this, the Internet Engineering Task Force (IETF) has also developed Integrated Services (Intserv) architecture and Differentiated Services (Diffserv) architecture to support QoS.

Generally speaking, ATM has three main technologies to guarantee QoS: flow identification, queuing and scheduling, and intelligent path selection and bandwidth reservation. Flow identification is the mechanism by which a switch can tell one class of traffic from another, so it is the fundamental technology of supporting QoS. ATM today provides support for six service categories, each with a different set of QoS parameters. These parameters specify the requirements of an application for network service as measured by raw bandwidth, predictable delivery, and error rates. These parameters include Constant Bit Rate (CBR), non real-time Variable Bit Rate (nrt-VBR), Real-time VBR, Available Bit Rate (ABR), Unspecified Bit Rate (UBR), and Differentiated UBR (D-UBR). ATM uses UNI signalling, Private Network Network Interface (PNNI) routing and Call Admission Control (CAC) to

select path and reserve bandwidth. UNI Signaling allows hosts to communicate their QoS requirements to the ATM network. PNNI routing allows switches to maintain a dynamic view of bandwidth load on other switches, communicate QoS requirements across the ATM network, and select the best path for each connection based on its QoS requirements and the available capacity of different links. CAC is the bandwidth reservation logic by which a switch updates its internal count of available bandwidth based on each circuit on a link. As more circuits are established, the available link capacity is reduced. When the switch has insufficient capacity to support a particular new connection, the switch will refuse the attempt.

It is worthy of mentioning that ATM uses fixed-size transfer units called cells as the basic transfer unit. Because the fixed cell size, the implementation of its traffic management functions is simpler for ATM than for other technologies which deal with variable size data units. The standardization of the basic traffic management functions of ATM is completed and its more detailed description can be found in [GG98].

Since IP only provides best effort service and does not participate in resource management, it can not provide service guarantees on a per flow basis or provide service differentiation among traffic. In order to enable QoS in IP, the Internet Engineering Task Force (IETF) has developed the Integrated Service architecture (Intserv) [RFC1633] that supports differentiating services, such as best effort service, guaranteed service, and controlled load service [BQL02]. Guaranteed service provides firm end-to-end delay and it is oriented towards hard real-time applications. Controlled load service has no quantitative guarantees

but it aims to provide better performance than best effort service. It is oriented towards adaptive real-time applications. In general, this architecture requires state management in the core of the network while at the same time requiring Resource Reservation Protocol (RSVP) signaling protocol [BZB97] that applications use for setting up paths and reserving resources toward receivers before sending data. However, the reliance of RSVP on end-to-end per-flow state and per-flow processing in every router is the major drawback preventing its large-scale deployment because of scalability concerns in large networks. Here the term "flow" refers to a set of related IP packets originating from a single application at a single source belonging to the same stream (each having a unique identifier) [BQL02].

In order to support much more scalability, the IETF developed the Differentiated Services (Diffserv) architecture which does not require state management like Intserv and is directed at a method for providing differentiated services [HR00]. Unlike Intserv, which offers per-flow QoS support, the Diffserv offers aggregate-flow QoS support, which classifies packet into a small number of aggregated flows or classes, based on the Diffserv Code Point (DSCP) in the packet's IP header. The amount of state information at each router is reduced to the number of classes rather than the number of flows, and functions such as classification, marking and policing are only needed at the edge routers of the network while assuming a stateless core. Diffserv has the potential to complement IntServ rather than replacing it.

Multi-protocol Label Switching (MPLS) [RVC99] is a fast label-based switching technique that offers new QoS capabilities for large scale IP networks. In MPLS, the packets are assigned a label when they enter an MPLS network and all subsequent packet treatment

within the MPLS network is based on that label only. MPLS was originally proposed to improve the forwarding speed of routers, but now becomes a key standard technology that offers new capabilities for large-scale IP networks. MPLS uses QoS routing to arrange how traffic flows through the network and improve the utilization of the network. The inherent characteristics of MPLS make it easily support aggregated flows. Diffserv-like (DS) mechanisms [FDV99] are introduced in MPLS networks for scalable QoS support.

2.3 End-to-End Flow and Congestion Control

QoS routing alone may not be effective enough to achieve or guarantee the QoS for applications because they may exhibit unpredictable behavior, resulting in network traffic overload. Such overload traffic considerably degrades overall network throughput and makes the applications useless. Therefore, an end-to-end flow and congestion control scheme is typically needed to avoid the congestion collapse of the global Internet [FF99]. Flow control is used to properly match the transmission rate of the sender to that of the receiver and the network. Congestion control is primarily concerned with an overloaded network. Flow and congestion control is suggested to be built in the transport layer according to the end-to-end principle [SRC84] which argues that we should place the complicated functions in the hosts, not inside the network. On the other hand, the purpose of the transport layer is to enhance the QoS provided by the network layer. If the network service is impeccable, the transport layer almost need do nothing. However, if network service is poor, the transport layer has

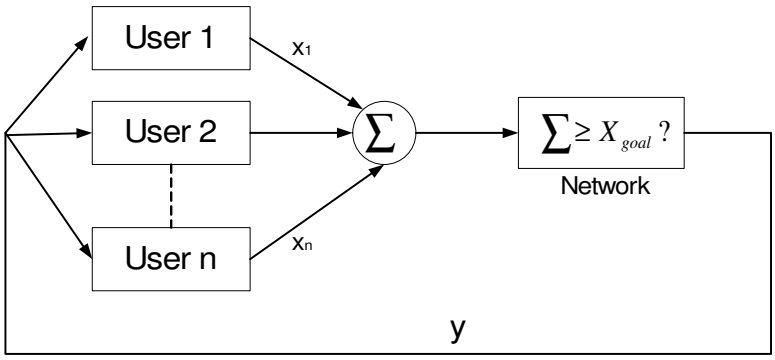

Figure 2.1: AIMD Model from Paper [DR89]

to bridge the gap between what the transport users want and what what the network layer provides. Thus, in the following sections we will introduce some classical end-to-end flow and congestion control algorithms.

2.3.1 AIMD Control Algorithm

The AIMD model was first presented by Chiu and Jain [DR89]. As shown in Figure 2.1, if during time slot t the i_{th} user's traffic load is $x(t)$, the total traffic load is $\sum$. The system will decide its load level and send a feedback y to the users. The users will cooperate with the system and response it by $u(t)$. This $u(t)$ is a function of the users' previous traffic demand and the feedback of the system. Thus, this user's traffic load at next time slot $t+1$ will decided by $x(t)$ and $u(t)$. Therefore, the problem is changed to how to find the relationship between $x(t)$ and $x(t+1)$. Chiu and Jain discussed all the cases and concluded that any additive-increase and multiplicative-decrease (AIMD)(α, β) scheme converged to fairness and

efficiency by assuming all the feedback was synchronous. This is a useful conclusion, and many congestion control mechanisms use this AIMD scheme as a theoretical basis. Basically we can use the following Equation (2.1) to express the congestion avoidance in AIMD. Here, α and β refers to increase-decrease constants.

$$W(i+1) = \begin{cases} W(i) + \alpha, & \text{No congestion} \\ \beta W(i), & \text{otherwise} \end{cases} \tag{2.1}$$

where i is a discrete time slot and every i refers to "full" round-trip time. $W(i)$ is the congestion window size at the i^{th} slot. TCP is an example of AIMD (1,1/2).

A TCP congestion control scheme is basically composed of slow start, AIMD, retransmit timers, and ACK-clocking algorithms [Flo01]. TCP uses a dynamic slide-window (congestion window W) to control the packets which are projected to the network. During slow start, W increases exponentially for every round-trip time (RTT). In the steady state, TCP uses AIMD $(1,\frac{1}{2})$ to react to congestion. The Acknowledgement packets (ACKs) are used to pace the transmission of packets. Therefore TCP is called as "self-clocking".

Since the appearance of TCP, many TCP variants [SKV03] have been developed. A lot of research has been done to utilize the bandwidth more adequately [Low03] while adhering to this underlying framework or present some analytic models to analyze throughput [SKV01, MR01, AR03, Nic01]. TCP Tahoe [Jac88] is the original. TCP Reno, TCP New-Reno, and TCP Vegas, etc., appeared subsequently. TCP Tahoe includes slow start, AIMD, retransmit timers, and ACK-clocking. It uses time-out and duplicate ACKs as congestion signs. As

TCP Tahoe does not deal well with multiple segments drops within a single window of data, two refinements, called fast retransmit and fast recovery were subsequently implemented in TCP Reno to recover from loss more efficiently. New-Reno modifies the fast retransmit and fast recovery, and it takes a partial ACK as an indication of another packet lost. SACK makes receiver give more information to sender about received packets so as to allow the faster recovery from multiple packet loss. Sikdar *et al.* [SKV03] studied the latency and steady-state throughput of TCP Tahoe, Reno, and SACK by an analytic model. TCP Tahoe, Reno, New-Reno and SACK all use packet loss to detect congestion. In order to achieve a better throughput and a lower loss rate, TCP Vegas was presented and it used round-trip-time (RTT) to avoid congestion. Recently, TCP Westwood [GM04b, CGM02] was proposed to modify the fast recovery by setting the value of slow start threshold and congestion window after a loss event based on available bandwidth estimate.

As for the signs of congestion, besides the loss [SKV03] and delay [MNR03], Kalampoukas *et al.* [KVR02] used delay-bandwidth product as congestion sign, and Aweya *et al.* [AOM02] proposed using ACK rate to detect congestion. In addition, a scalable TCP [Kel03] was proposed to improve performance in high-speed wide area networks by modifying the basic AIMD equation. Fast TCP [WWL05] was presented too, showing that it can provide great performance improvement over Gbps speed networks. Sastry and Lam [SL02] presented a comprehensive theoretical framework called CYRF for window-based congestion control to suit different applications and network needs. Zhang and Tsaoussidis [ZT03] derived an analytical expression of the $knee$ and $cliff$ defined in [DR89] and presented an equation

for adaptively setting an efficient window decrease ($\frac{cwnd_{knee}}{cwnd_{cliff}}$). Also, they proposed to use Coefficient of Variation (CoV) to observe the throughput smoothness experienced by flow.

2.3.2 TCP Vegas

In recent years, some studies [ZT03, LM05, SV03, MNR03] improved or emulated the TCP Vegas [BOP94] scheme, which has the congestion avoidance algorithm described by Equation (2.2). The original evaluation of TCP Vegas appeared in [ADL95]. It experimentally showed that TCP Vegas yields higher network throughput than TCP Reno. In [LPW02] authors explained and analyzed the TCP Vegas theoretically by presenting a dual model of the Vegas algorithm. It is shown in [HBG00] that TCP Vegas cannot guarantee fairness. Its congestion avoidance mechanism follows an additive increase and additive decrease scheme, which does not guarantee fairness [DR89].

$$W(i+1) = \begin{cases} W(i)+1, & \text{if } \frac{W_i}{BaseRTT} - \frac{W_i}{RTT} < \alpha_t \\ W(i)-1, & \text{if } \frac{W_i}{BaseRTT} - \frac{W_i}{RTT} > \beta_t \\ W(i), & \text{else} \end{cases} \tag{2.2}$$

where $\frac{W_i}{BaseRTT}$ refers to the expected rate and $\frac{W_i}{RTT}$ refers to the actual rate. The actual implementation estimates $BaseRTT$ by the minimum round-trip time RTT observed so far. α_t and β_t are the thresholds of throughput. Li and Ma [LM05] proposed replacing BaseRTT

with the RTT in the original Vegas rate estimation function and improved the fairness for those newcomer flows with smaller congestion window W compared with that of flows with larger W.

Low [Low03] modeled Vegas as a distributed optimization algorithm and identified the corresponding congestion window function used in TCP Reno, TCP Reno with RED, and TCP Vegas. Samios and Vernon [SV03] presented an analytical TCP Vegas model and introduced a new throughput equation under loss environment. Their non-loss throughput was given as $T = \frac{\beta}{R - BaseRTT}$, where T is the throughput when the loss is negligible, and R is the average-round-trip time for the transfer. $baseRTT$ is relatively stable throughout the flow. β is the threshold of throughput. Martin *et al.* [MNR03] concluded based on their measurement analysis that delay-based congestion control (DCA) cannot be incrementally deployed over high speed Internet paths. They also pointed out that background traffic was important to accurately assess the impact of a large deployment of DCA. Meanwhile, [WWL05] argued for fully exploited delay as a measure of congestion, argued that loss information was needed and proposed "Fast TCP" scheme.

2.4 Summary

In this chapter, we reviewed some research work related to QoS mechanisms, such as ATM, Intserv, Diffserv and MPLS. Then, we referenced a complete literature of AIMD congestion control algorithms and TCP, as AIMD is one of important factors which makes TCP

robust and reliable. Besides using loss as congestion indications, using delay as congestion indications is another active area and the classical one is TCP Vegas. In this dissertation, we propose a QoS-based flow and congestion control scheme. In order to make this scheme to be friendly to TCP flows, we also use the idea of AIMD scheme, and we will present its detailed description in Chapter 5.

CHAPTER 3

ONLINE ESTIMATION OF PACKET LOSS RATE

3.1 Overview of Cognitive Packet Networks

CPN is a store-and-forward network and its routing scheme is different from the traditional Internet Protocol(IP) scheme. The difference shows in several aspects:

- CPN has three different packets: Smart Packets (SPs), Dumb Packets (DPs), and Acknowledgement Packets (ACKs). SPs are used to exploit routing paths based on different goals and thus they do not take any user data. DPs are used to carry user data. DPs are called "dumb" because they just follow the routing paths that have already been found by SPs, instead of finding paths by themselves. SPs and DPs both have ACKs. ACKs are used to store the route followed by the original packets and collect measurement data. In contrast, IP does not have probe packets like SPs or any acknowledgement packet like ACKs. It just has one type of packet which carries data and follows the routing paths from routing tables.

- CPN's routing is adaptive and based on the different goals. SPs are used to exploit and establish the connection, while DPs and ACKs will be source routed. The routing information is stored in the mailboxes along every node of a connection, such as the delay, loss, and jitter on routing paths. The deposited information in the mailboxes depend on DPs and ACKs. SPs will be trained by RNNRL algorithms based on the feedback information to explore better paths in order to provide information for subsequent DPs' and ACKs' source routing. Therefore, these three packets affect each other and core learning algorithm is RNNRL. In contrast, IP is not an adaptive routing scheme. It compares the number of links along a connection and chooses the path which has the minimum number of links as its routing path. Therefore, the routing path is fixed once the network links are all available, which easily leads to congestion.

In general, the routing in CPN also involves two algorithms. On the one hand, it uses cognitive packets (smart packets) to explore the path and the acknowledgement packets to collect the up-to-date information; On the other hand, it employs reinforcement learning algorithm to decide the "best" path based on collected information. In order to have a more detailed and complete understanding of CPN, we will describe the lifecycle of a packet and the reinforcement learning algorithms as follows.

3.1.1 Lifecycle of a Packet

When the CPN needs to establish a connection, a SP is generated and sent out from source S to destination D. If the SP arrives at a node which has not sent out any DP for destination D and thus has no information about routing, this node will send out SPs for destination D in round-robin mode to its neighbors; otherwise, the SP is routed using reinforcement learning (as described below).

After the SP arrives at the destination D, it finishes its mission and the ACK is generated and stores the route that original SP brings. Thus the ACK being returned as a result of the SP will travel along the "reverse route" of the SP. The reverse route has to be examined and eliminate the part which forms cycle and compose a new route for the ACK. For example, if the path which SP brings back is $< a, b, c, d, a, e, f, c, g >$, the new reverse path will be $< g, c, b, a >$. ACKs deposit information in the mailboxes (MBs) of the nodes they visit.

SPs are sent out sequentially from the source until the first ACK comes back from the destination. When the first ACK comes back, all DPs of the same QoS class which carry the user data for the same destination will be sent out using the the first route that was brought back. Whenever a new ACK comes back, the DPs will use the new route. Here, we assume that the network state is changing with time and the routing must respond to this change in a timely manner. Since the routing path is fixed for DPs, DPs just collect some measurements at nodes and go to their destination.

After DPs arrive at the destination D, their ACKs are generated. Same as SPs' ACKs, the DPs' ACKs will get the sequence of nodes which are visited by the DPs and traverse their reverse route.

For more detailed and complete discussion about CPN please refer to [GSX99, GLX01, GLM02].

3.1.2 Reinforcement Learning Algorithm in CPN

Different approaches to learning could in principle be used to discover good routes in a network, including Hebbian learning, back-propagation [RM86], and reinforcement learning (RL) [Sut88]. Simulation experiments conducted on a 100 node network showed that RL is the most effective approach, and that it provides significantly better QoS than shortest-path routing [GLX00a]. In order to guarantee convergence of the RL algorithm to a single decision (e.g., selecting an output link for a given smart packet), CPN uses the random neural network (RNN) [Gel93, GMD99] which has a unique solution to its internal state for any set of weights and input variables. CPN's RL algorithm uses the observed outcome of a decision to "reward" or "punish" the routing algorithm, so that its future decisions are more likely to meet the desired QoS goal. The "goal" is the metric that characterizes the success of the outcome, such as packet delay, loss, jitter and so on.

As an example, the QoS goal G that SPs pursue may be formulated as minimizing transit delay W, or loss probability L, jitter, or some weighted combination captured in the numerical goal function G and the reward $R = 1/G$. Successive values of R, denoted by R_l, $l = 1, 2,$ ..., are computed from the measured delay and loss data (see Section 4.3), and are used to update the neural network weights.

The CPN RL algorithm [GLX01, GLM02] first updates a threshold value:

$$T_l = aT_{l-1} + (1 - a)R_l, \tag{3.1}$$

where a is a constant $0 < a < 1$, typically close to 1, and R_l is the most recently measured value of the reward. T_l is a running value that is used by the RL algorithm to keep track of the historical value of the reward and is used to determine whether a recent reward value is better or worse than the historical value.

Suppose that the l^{th} decision selected output link k, and that we received feedback from the network which measured the l^{th} reward R_l. We first determine whether R_l is greater than, or equal to the threshold T_{l-1}. If that is the case, then we conclude that the previous decision worked well; we increase the excitatory weights to the neuron k that was the previous winner (in order to reward it for its success), and make a small increase of the inhibitory weights leading to other neurons. However, if R_l is less than the threshold, we conclude that the previous decision was not so good, and moderately increase all excitatory weights leading to other and increase significantly the inhibitory weights leading to the previously selected neuron k in order to punish it for being unsuccessful:

- If $T_{l-1} \leq R_l$
  - $w^+(i,k) \leftarrow w^+(i,k) + R_l$,
  - $w^-(i,j) \leftarrow w^-(i,j) + \frac{R_l}{n-2},\ for\ all\ j \neq k.$
- Else
  - $w^+(i,j) \leftarrow w^+(i,j) + \frac{R_l}{n-2},\ for\ all\ j \neq k,$
  - $w^-(i,k) \leftarrow w^-(i,k) + R_l.$

In order to keep the weight variables within the same range of values, we normalize all the weights. First for each i we compute:

$$r_i^* = \sum_{m=1}^{n} [w^+(i,m) + w^-(i,m)], \tag{3.2}$$

and then normalize the weights using:

$$w^+(i,j) \leftarrow w^+(i,j)\frac{r_i}{r_i^*},\ \ w^-(i,j) \leftarrow w^-(i,j)\frac{r_i}{r_i^*}\ .$$

Then the probabilities q_j are computed using the RNN state equation [Gel93]:

$$q_j = \frac{\Lambda + \sum_{m=1}^{n} q_m w^+(m,j)}{\sum_{m=1}^{n}[w^+(j,m) + w^-(j,m)] + \sum_{m=1}^{n} q_m w^-(m,j)}. \tag{3.3}$$

Finally the i value with largest q_i is identified, i.e. $q_i \geq\ q_j$ for all $j = 1, ..., n$, and the smart packet is placed in the corresponding output buffer i. In CPN, every received dumb packet's acknowledgement packet (DACK) needs to deposit its information into the mailbox of the current node. This information could be any of round-trip delay, jitter, loss rate, and/or

available bandwidth, depending on the user's QoS. Based on this feedback, the node uses the RNNRL algorithm to generate the next hop for the later smart packets (SPs) carrying the same QoS. Therefore, for those applications which need to minimize loss rate, it is important to estimate the online packet loss rate at every node when a DACK arrives.

3.2 End-to-End Loss Rate Estimation

The loss rate refers to the loss over a path. However, exact determination of online loss rate is challenging. Generally speaking, packet loss rate refers to the ratio of the number of lost packets to the number of packets sent from the source during a period. It can be formulated as $L = 1 - \frac{R}{N}$, where R is the number of packets received at the destination and N is the number of packets sent from the source. Obviously it is easy to calculate the *off-line* packet loss rate. However, it is hard to get the *online* loss rate at the source due to two reasons: (1) the source does not know the *actual* number of received packets at the destination at a certain time; (2) some packets which probably will arrive at the destination successfully are still in flight at the certain time. Thus even the actual number of packets received at the destination can not reflect the real-time network performance correctly.

On the other hand, it is impractical to have the destination record a count of the number of packets received for each possible route from every possible source. We make an assumption that the forward loss rate is proportional to the reverse loss rate. This assumption is based on the fact that forward route is same with the reverse route. We are interested in

the on-line loss ratio $L^f_{t_k}$ for the DPs from the current node to the destination, where t_k is the arrival time of the k^{th} received DACK at the current node. Let us define the on-line "round-trip loss rate" (RLR) at the k^{th} DACK's arrival time at the current node. The RLR associated with the k^{th} received DACK is expressed by $L_{t_k} = 1 - \frac{A^a_{t_k}}{A^e_{t_k}}$, where t_k is the arrival time of the k^{th} received DACK at the current node, $A^a_{t_k}$ is the number of actually received DACKs from time 0 to time t_k, and the $A^e_{t_k}$ the number of expected DACKS from time 0 to current time t_k. Thus, the arrival of the k^{th} DACK indicates that the node has received K DACKs so far, and therefore $L_{t_k} = 1 - \frac{K}{A^e_{t_k}}$. Assuming that the on-line loss rate is $L^b_{t_k}$ for the DACKs from the destination to the source, we have L_{t_k}=$L^f_{t_k} L^b_{t_k}$. Further, assuming that the backward loss rate $L^b_{t_k}$ is proportional to the forward loss rate $L^f_{t_k}$, that is, $L^b_{t_k} = \beta L^f_{t_k}$, we have $L^f_{t_k} = \sqrt{\frac{1}{\beta}(1 - \frac{K}{A^e_{t_k}})}$. Thus, it is important to correctly estimate $A^e_{t_k}$ for us to get the online DPs loss rate.

In the following sections, we will first discuss the online packet loss rate in a special case, and then we will describe our proposed online packet loss rate estimation.

3.2.1 DP's Loss Rate Estimation

Let us consider the following case: the source sends dumb packets (DPs) to the destination at a constant rate of λ from time 0 to time t_e. The destination sends a acknowledgement packet (DACK) to the source after receiving a DP. The round-trip time from sending a DP to receiving its DACK at the source is D and we assume $D < t_e$. From time 0 to time t, the

total number of DPs sent out from the source is N_t and the total number of ACKs received at the source is A_t.

Assuming that there is no loss for DPs or ACKs, we can get the following expressions for N_t and A_t.

$$N_t = \begin{cases} \lambda t & (t \leq t_e) \\ \lambda t_e & (t > t_e) \end{cases} \tag{3.4}$$

$$A_t = \begin{cases} \lambda(t-D) = N_t - \lambda D & (D \leq t \leq t_e) \\ \lambda t_e = N_t & (t \geq t_e + D) \\ \lambda(t-D) = N_t - \lambda(D-(t-t_e)) & (t_e < t < t_e + D) \\ 0 & (t < D) \end{cases} \tag{3.5}$$

Further, A_t can be expressed by the following equation:

$$A_t = N_t - max(0, \lambda(D - max(0, t - t_e))) \tag{3.6}$$

If loss happens from time 0 to time t and the source only received $A_t^{'}$ DACKs from time 0 to time t, the loss possibility for DPs and DACKs from time 0 to time t is L_t, which can be expressed by:

$$L_t = 1 - \frac{A_t^{'}}{A_t} \tag{3.7}$$

Further, let us define the loss probability for the DPs from the source to the destination be L_t^f and the loss probability for the DACKs from the destination to the source be L_t^b, and we have $L_t = L_t^f L_t^b$. Assuming that the L_t^b be proportional to L_t^f, we get the forward loss rate express described by $L_t^b = \beta L_t^f$.

Then, L_t^f can be written as follows:

$$L_t^f = \sqrt{\frac{1}{\beta}(1 - \frac{A_t^{'}}{(N_t - max(0, \lambda(D - max(0, t - t_e))))}} \tag{3.8}$$

Similarly, we use the following estimated loss rate for the DPs from the current node to the destination

$$L_{t_K}^f = \sqrt{\frac{1}{\beta}(1 - \frac{K}{(N_{t_K} - max(0, \overline{\lambda_{t_K}}(D_m - T_a))}} \tag{3.9}$$

where D_m is the minimal round-trip delay experienced by the DPs.

$$\lambda_{t_K} = \frac{1}{t_N^{'} - t_{N-1}^{'}} \tag{3.10}$$

$$\overline{\lambda_{t_K}} = (1 - \alpha)\overline{\lambda_{t_{K-1}}} + \alpha\lambda_{t_K} \tag{3.11}$$

$$T_a = \begin{cases} 0 & (t_K \leq t_e) \\ t_K - t_N^{'} & (t_K > t_e) \end{cases} \tag{3.12}$$

where t_K is the arrival time of the k^{th} received DACK, and t'_N is the arrival time of the last received DP at the current node. t_e is the ending time of the flow. It is difficult for the node to know t_e in the practice, thus we tried to approximate t_e by comparing the elapsed time since the last received DP to the maximal DP's input gap at the current node. More details of the algorithm are shown in the Algorithm 1 and Algorithm 2.

Obviously, according to the the Equation (3.9), we know that in a steady state, or when the time t_K becomes infinity, the one-way forward loss rate equation reads as:

$$L_f = \sqrt{\frac{1}{\beta}(1 - \frac{K}{N_{t_K}})} \tag{3.13}$$

where β is 1 in the experiments.

Algorithm 1 Update DP's Arrival Rate

```
1: if a DP arrives then
2:    N=N+1;
3:    if N > 1 then
4:       Ingap=CurrentTime - mb→lastarrival;
5:       mb→AvgIngap=(1 - α)mb→AvgIngap + αIngap;
6:    end if
7:    mb→lastarrival=CurrentTime;
8: end if
```

Algorithm 2 Update the Loss Rate

```
1: if a DACK arrives then
2:    K=K+1;
3:    if mb→minD > currentDACK→D then
4:       mb→minD=currentDACK→D;
5:    end if
6:    idle=CurrentTime-mb→lastarrival;
7:    if idle > (1.5*mb→maxIngap) then
8:       T_a=idle;
9:    else
10:      T_a=0
11:   end if
12:   Update loss rate (N, K, T_a, mb→minD,mb→AvgIngap) according to loss rate Equation (3.9).
13: end if
```

3.2.2 Experiment Analysis on Loss Rate Estimation

All of our experiments are based on the topology shown in 3.1. The bandwidth of every link is 10 Mbps. Every packet is 1024 KBytes. Node 201 is the source and node 219 is the destination.

In order to see if our on-line DPs loss rate estimation can reflect the actual loss rate correctly, we designed the following experiments. We let the source node 201 send CBR traffic to the destination node 219 for a period of time, and every node tracks the on-line loss rate whenever it receives a DACK during this period. After the flow stopped and we waited for a time long enough to ensure all the packets which were not lost on the routes have arrived at the destination, we calculated the off-line forwarding loss rate by equation $L_f = 1 - \frac{R}{N}$ and the off-line "round-trip loss rate" by equation $L_r = 1 - \frac{A}{N}$, where R is the

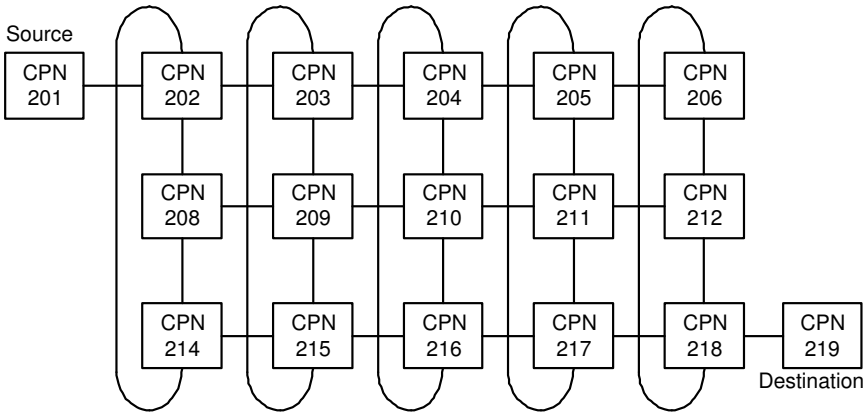

Figure 3.1: The Topology of CPN Test Bed.

number of received DPs at the destination, N is the number of DPs sent from the source, and A is the number of received DACKs at the source.

3.2.2.1 Light Traffic

In the first experiment, CBR = 3 Mbps and the total number of sent packets are 200. The on-line DPs loss rate estimation on source node 201 is shown in Figure 3.2, and the on-line DPs loss rate estimation on the intermediate node 210 is shown in Figure 3.3. It is worth noting that since the routes of CPN could be varied, the intermediate nodes and the source nodes have different A, N, and R. When rate is 2 Mbps, the off-line forwarding packet loss rate and the "round-trip loss rate" are both 0. From Figure 3.2 and Figure 3.3, we can see most of on-line DPs loss rate are 0, which is almost same with the off-line forwarding

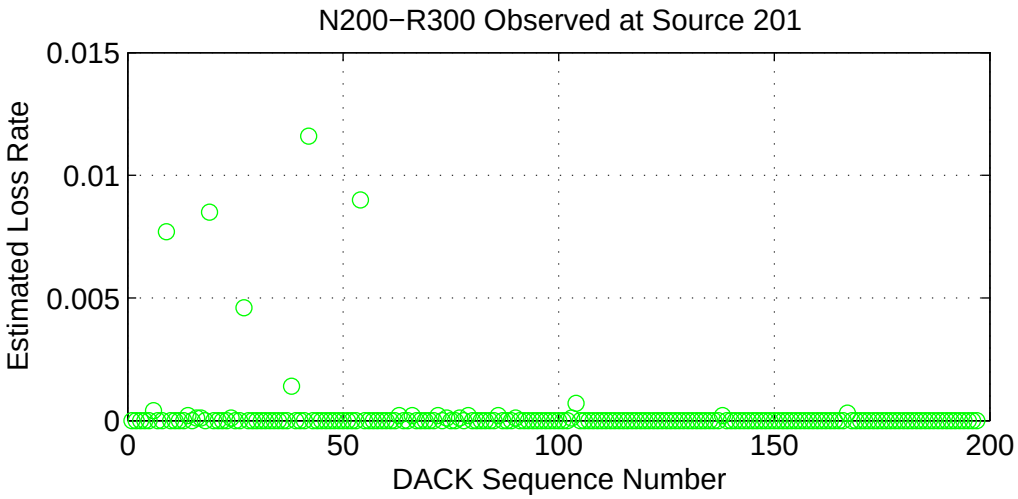

Figure 3.2: Loss Rate Observed at Source Node with 3 Mbps Traffic.

packet loss rate. Therefore, we conclude that our estimation algorithm works well in the light traffic.

3.2.2.2 Heavy Traffic

When the source node 201 sent 500 packets to the destination at a rate of 9 Mbps, the off-line forwarding packet loss rate is 0 and the square root of "round-trip loss rate" is 0.07. Here, we choose the square root of "round-trip loss rate", as we use the square root of online "round-trip packet loss rate" to estimate the online DPs loss rate in the experiment. Thus, in Figure 3.4, we compare the online DPs loss rate (circled line) with the square root of off-line round trip loss rate (solid line) observed at the source node. We found our estimation also approximate the off-line square root of round trip loss rate. We also found that *actual* DPs

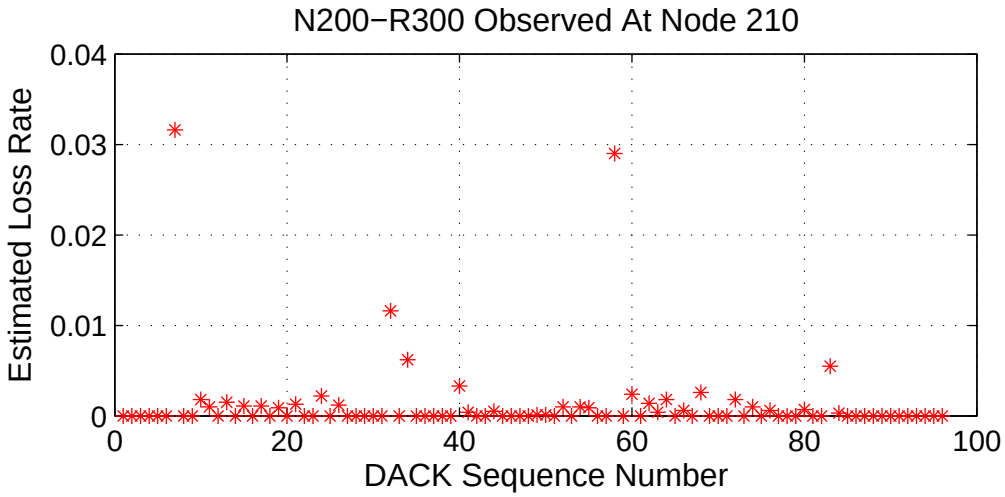

Figure 3.3: Loss Rate Observed at Immediate Node with 3 Mbps Traffic.

loss rate is 0, but our estimation is larger than 0. This can be explained by the fact that actual number of lost DACKs are bigger than the number of lost DPs, which reminds us we could change the β value in the Equation 3.13 to explore the relationship of forwarding loss and backward loss in the future. Besides, from the circled line in the Figure 3.4 we see an obvious jump of loss rate from 0 to 0.05. We think this jump is an useful information indicating congestion.

3.3 Summary

In this chapter, we introduced three types of packets in CPN and its reinforcement learning algorithm. Based on CPN framework, we proposed an online estimation of packet loss rate and discussed its equations in short-term and steady state respectively. This approach was

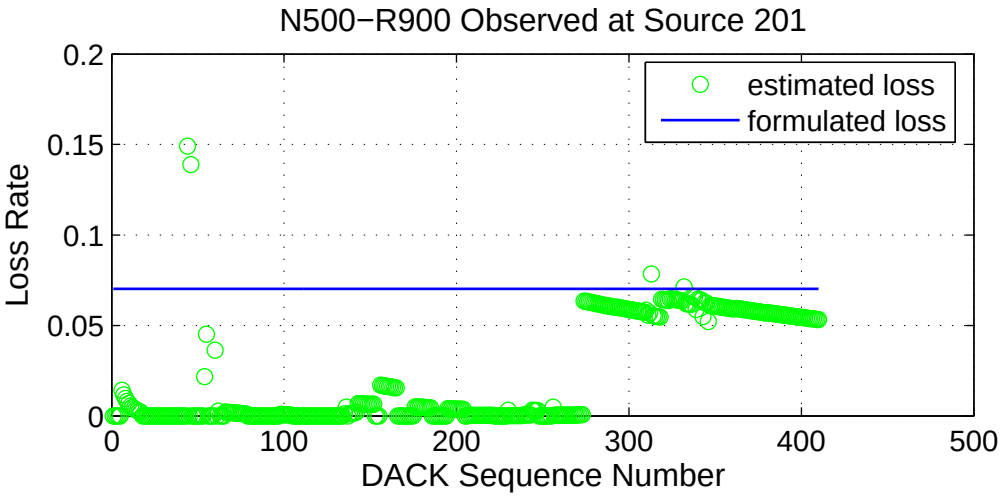

Figure 3.4: Loss Rate Observed at Source Node with 9 Mbps Traffic.

based on the assumption that the forward loss rate was same with the backward loss rate. It could estimate the source-to-destination packet loss ratio on a path which included multiple hops. We experimentally evaluated the algorithm of estimating online packet loss rate, and will use it to complete our proposed QoS routing and a QoS-based flow and congestion control scheme, which will be described in Chapter 5 and Chapter 6.

CHAPTER 4

SINGLE AND MULTIPLE-METRIC QOS ROUTING

4.1 Overview

In Chapter 3, we have already known that CPN is a store-and-forward network in which routes are chosen dynamically by SPs which belong to a connection. Thus, we can assign different goals to SPs according to the users' different QoS requirements. However, how to map the users' QoS with the goals and how to make routing decisions based on different goals are a challenge. In this chapter, we will explore using loss, delay and a combination of two as different QoS goals and making adaptive routing decisions based on these QoS goals. We use the term "goal" to indicate that there is no QoS guarantees, and that CPN provides best effort to satisfy the users' desired QoS. In addition, we will present and describe different reward functions for different QoS requirements. We will also present experimental results and compare QoS perceived by users in different routing algorithms. Before we introduce our proposed algorithms, we will review some related work first.

4.2 Related Work

Routing includes two facets. One is to gather the information about the network, and the other is to select a route for current packet based on the collected information. Traditionally, the network nodes (routers) are used to collect information and there are two mechanisms for every node in networks to collect the information of networks [Tan96]: link-state protocol, such as Routing Information Protocol (RIP), and distance-vector protocol, such as Open Shortest Path First (OSPF). The link-state protocol is to use every router to measure the delay or cost to each of its neighbors and deliver this information to all the other routers and then run Dijkstra's algorithm locally to computes the shortest path to every other router. The distance-vector protocol is to use every router to maintain and estimate the delay or average hops to every other routers and deliver distance vectors to each its neighbor every $\triangle T$ seconds. A distance vector has an entry for every possible destination, consisting of the property of the best path and the next node on the best path. These routing algorithms only consider a *single* routing metric, namely number of hops or the average path delay. Different from them, the goal of QoS routing is to find routes that satisfy multiple QoS constraints such as delay, jitter, loss rate, etc and optimize the resource utilization.

QoS routing is the process of selecting the path to be used by the packets of a flow based on their QoS requirements, such as delay and loss. In other words, QoS routing refers to algorithms that compute paths that satisfy a set of end-to-end QoS requirements. Although there are still strong concerns about the increased cost of QoS routing on the Internet, both in

terms of more complex and frequent computations and increased routing protocol overhead, many QoS routing algorithms have been proposed recently with different focuses. A good survey on QoS routing can be found in [CN98]. A feasible way to develop QoS routing protocols is based on existing routing protocols, such as OSPF and BGP. The QoS routing protocols are to provide single or multiple routes according to QoS requirements as well as network statistics. Such routing involves two algorithms. One is to decide which path is the "best" path based on collected information. The other is to collect network information and keep it up-to-date. Routing performance depends on the latter.

QoS routing solutions often globally exchange the link metrics to compute appropriate paths. This is often developed by extended versions of link state protocols, but it often results in a rather high communication overhead [CN98]. Finding a feasible path with two independent path constraints is NP-hard [GJ79]. In [NZ02] the author presented a "localized" QoS routing approach and used multiple paths between a source and a destination. Although it cannot guarantee specific service of global routing, the source node can dynamically adapt the proportions of traffic over a set of candidate paths. Some researchers [KL01] developed some algorithms to deal with fault tolerance and path restorability. The proposed schemes had direct applications in MPLS networks. On the other hand, inaccurate information is used to compute the QoS routes. In [Gue99] the authors presented the theory and algorithms analysis in this problem. In [AGK98] a performance perspective of QoS routing was presented.

IPv6 is also termed IPng or IP Next Generation. It provides a large number of enhancements over the current IPv4. Most important motivation to adopt IPv6 is that it expands the limited 32-bit address mechanism provided by IPv4 to 128-bits. Although IPv6 significantly increases the total size of the available IP address space, the challenge is whether or not this increased space sufficiently enables the continued growth of the Internet [MS99]. Two other significant components of the IPv6 protocol may provide a method to help deliver differentiated Classes of Service (CoS) [FH97]. The first component is a 4-bit priority field in the IPv6 header, which is functionally equivalent to the IP precedence bits in the IPv4 protocol specification to some extent. The priority field can be used in a similar fashion, in an effort to identify and discriminate traffic types based on contents of this field. However, IPv6 priority field does not offer any substantial improvement over the utility of the IP precedence field in the IPv4 header. The second component is a 24-bit flow label field, which was added to enable the labeling of packets that belong to particular traffic flows for which the sender might request special handling, such as best-effort service or real-time service. Thus, IPv6 tries to provide the ability to identify different flows so as to ensure QoS by providing for different classes of services. Different flows can then be monitored to ensure that the guaranteed performance was provided. However, the flow label is needed to cooperate with resource-reservation protocols [FH97], such as RSVP for IPv6. Aside from using the IPv6 flow label with RSVP, its benefit is not immediately determinable.

On the other hand, IPv6 routing is still under development. OSPF and RIP for IPv6 have been tentatively specified, but QoS-enabled OSPF is still being developed. All in all,

IPv6 does not offer any substantial QoS benefits above and beyond what already has been achieved with IPv4. A good overview of IPv6 can be found in [LLM98].

From a user's point of view, QoS is the performance of a network as perceived, such as loss probability, delay and so on. Ideally a user should choose a route suitable for its QoS. From a network's point of view, supporting QoS not only needs to maximize the utilization of network resources but also provides the corresponding service to different applications. Current IP routes all the traffic by the smallest hop number without differentiating various applications, and a route with the smallest hop number implies a minimal delay, which may not be suitable for the applications with other QoS requirements, e.g., minimizing loss rate. Therefore, in order to make users to obtain better performance of the network as specified, we investigate an adaptive routing to achieve users' QoS.

4.3 Constructing Composite QoS Goals in CPN

For an application which has QoS needs that include both loss and delay, we need to propose a form of *goal function*. Let us consider such a case: if a loss with probability $\overline{L_f}$ occurs, then the source need to wait extra time T, such as time out, and transmit the packet again. If no loss occurs, the source just uses a normal transmission time $\overline{W}$. Thus, we can estimate the average time to send a packet successfully and this average time could be regarded as

"goal". So the goal function based on both loss and delay can be written as follows:

$$G = (1 - \overline{L_f})\overline{W} + \overline{L_f}(T + G) \tag{4.1}$$

The reward function $R = 1/G$ is then given by:

$$R = \frac{1}{T\frac{\overline{L_f}}{1-\overline{L_f}} + \overline{W}} \tag{4.2}$$

In order to use R we must be able to estimate the average one-way forward delay $\overline{W}$ and loss possibility $\overline{L_f}$. However, estimation of end-to-end delay and loss is more challenging than the off-line measurements because of clock synchronization problem. In Chapter 3, we have described the features of CPN. CPN routes the DPs that carry the data and the DACKs by the same path, thus we can use a simple method to measure the delay. That is, when an ACK for a packet of class C that is going from S to D enters some node N from node M, the difference between the local time-stamp and the time-stamp stored in the ACK for this particular node is computed and divided by two. The resultant value is the estimated forward delay for a packet of QoS class C going to destination D via node M as next hop. We use $W(C, D, M)$ to denote it and this quantity is inserted in the goal function described above. Similarly the estimated forward loss rate $L(C, D, M)$ is needed too. We use the loss rate Equation (4.3) in steady state derived in Chapter 3 to estimate the forward loss rate.

$$L_f = \sqrt{1 - \frac{K}{N}} \tag{4.3}$$

where L_f is the loss rate over a path from the source S to the destination D through next hop M, K is the number of DP's ACKs arrived at the source, and N is the number of DPs sent from the source.

4.4 Reward Functions Based on Single and Multiple QoS Constraints

If users need to minimize their delay, the path with smallest delay should be selected. We can construct the reward R based on delay as follows:

$$R = \frac{1}{\beta \overline{W}}, \tag{4.4}$$

where W is the average one-way forward delay experienced by DPs, and β is a constant. In the experiment, we choose $\beta = 0.75$. We use low-pass filter method to smooth the average delay and loss rate.

If routing only selects paths which offer the lowest packet loss, there are several ways by which we can construct the reward R. One approach is to set $\overline{W} = 0$ in the expression (4.2):

$$R = \frac{1 - \overline{L_f}}{T \overline{L_f}}, \tag{4.5}$$

so that $1/T$ acts as a constant multiplier. In practice, since we do not want R to be infinite when $L_f = 0$, we set:

$$R = \frac{1 - \overline{L_f}}{T(\overline{L_f} + \epsilon)}, \tag{4.6}$$

where ϵ is a constant representing some minimal value for the loss. A simpler approach is to use R of the form:

$$R = \frac{\alpha}{\overline{L_f} + \epsilon}, \tag{4.7}$$

which relates loss directly to the reward. This is the approach we have taken in our experiments when we just deal with loss (rather than loss and delay). In our experiments, we have used the following numerical values of the constants: $\epsilon = 10^{-5}$ and $\alpha = 0.5$.

4.5 Experiments and Results

The measurements were performed under a variety of conditions on the network test-bed as shown in Figure 3.1. All tests were conducted using a flow of UDP packets entering the CPN network with constant bit rate (CBR) traffic from the source node 201 to the destination node 219 and a packet size of 1024 bytes. All CPN links used 10-Mbps point-to-point Ethernet. The CPN protocol was run with paths discovered by SPs, while DPs were used to carry users' payload. We designed two sets of experiments to evaluate our algorithms. One was to test the network performance without background traffic. The other was to observe its performance under the background traffic. Without background traffic, usually the high

sending rate is a main factor to bring up congestion, as it leads to buffer overflowing. Under background traffic, we can select lower sending rate lower and thus guarantee our algorithm to have enough time to learning and make a decision.

4.5.1 Performance Evaluation without Background Traffic

We first conducted experiments without background traffic. As shown in Figure 4.2, using the goal function based only on loss results in the lowest observed loss rates when the traffic rate is smaller than 11 Mbps. When the sending rate is very high, exceeding the bandwidth of the link, congestion happened and no one can get low loss rate. The curves in the Figure 4.1 show that the lowest delay is obtained by using only delay in the goal function. That is, when the users' QoS requirements is to minimize the end-to-end delay, they obtain the smallest delay by the routing based on only delay. Both Figure 4.2 and 4.1 show the goal function based on both loss and delay obtains medium delay and loss compared to the goal function based only on delay and only on loss. Besides, we also observe the jitter performance on the varied input rate shown in Figure 4.3. We find that using the goal function based only on delay or based on both delay and loss results in the smaller and smoother jitter than the goal function based only on loss.

In order to see if the end user is indeed obtaining the outcome in QoS that is requesting. We used Figure 4.4 and Figure 4.6 to show the reward function values at the end user. The reward is computed using the formulas given in equations (4.2) and (4.7), using the *measured*

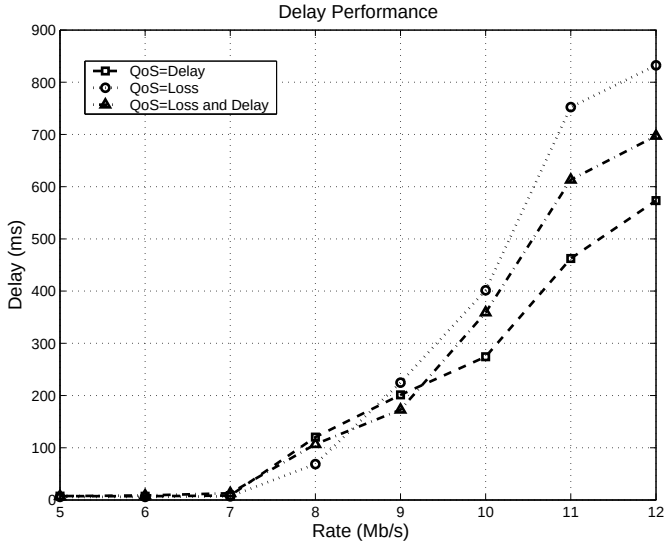

Figure 4.1: Delay Performance without Background Traffic.

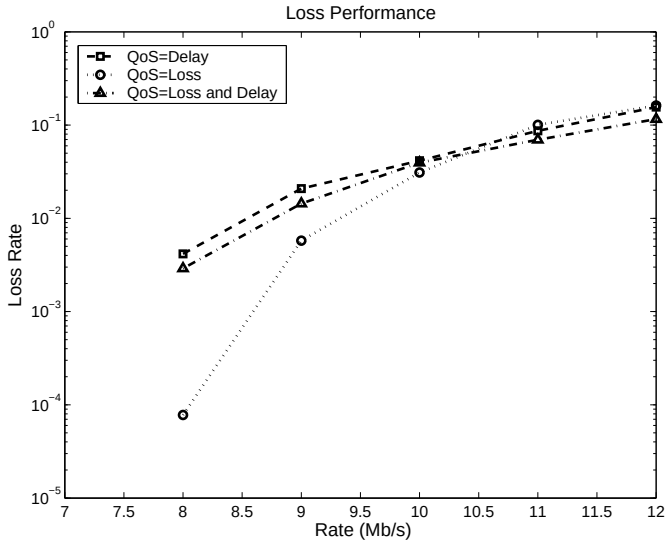

Figure 4.2: Loss Performance without Background Traffic.

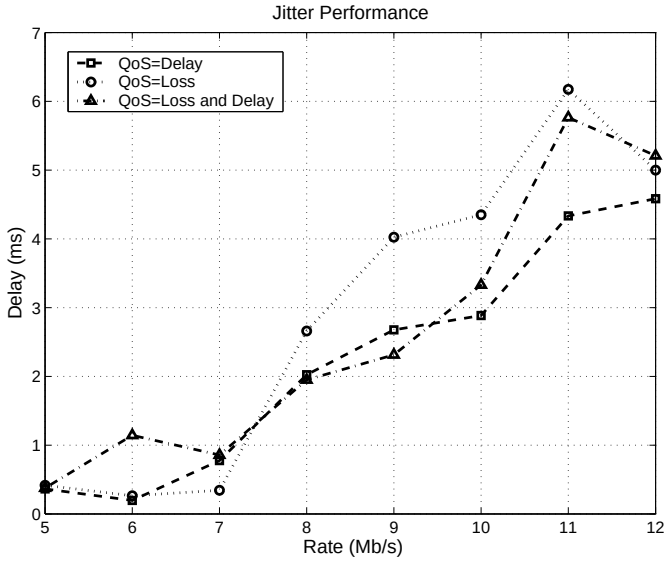

Figure 4.3: Jitter Performance without Background Traffic.

values of loss and delay with the constants as indicated in Section 4.4. In Figure 4.4, we can see the reward values based on only delay are very close using three different routing policies. Thus, we enlarge the part with higher transmission rate and observe that using delay as QoS goal or using delay and loss as QoS goals will provide the users who request minimizing delay with the best reward at higher traffic. In Figure 4.6, we can see very clearly that the highest reward, that is, the best reward is obtained if we use only loss in the goal function at medium traffic. The reason is that at the low traffic, all the goal functions result in an negligible effect of loss rate, and at the high traffic, congestion happens and all the goal functions can not avoid packet loss. This in further proved that users achieved their desired QoS.

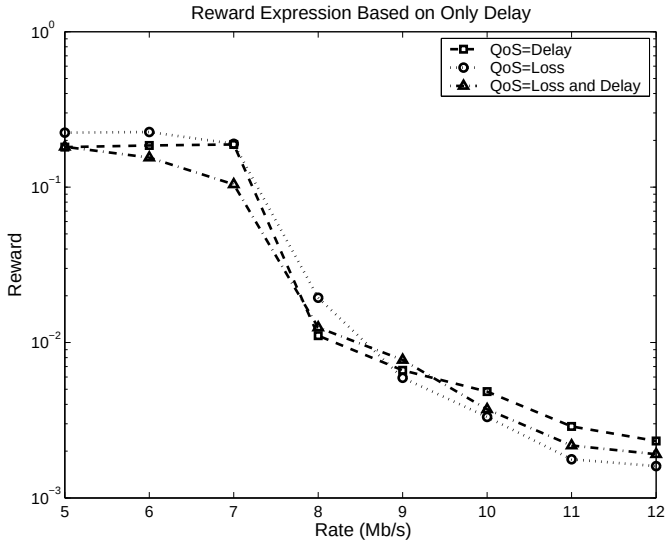

Figure 4.4: Reward Expression Based on Only Delay.

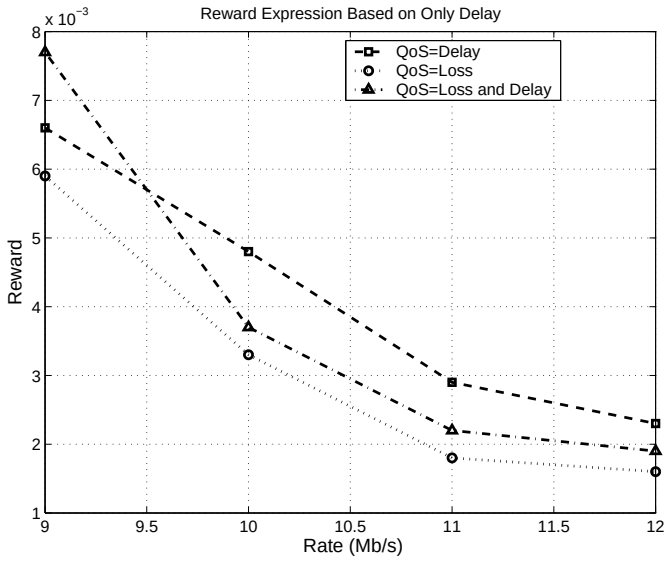

Figure 4.5: Reward Expression Based on Only Delay with High Transmission Rate.

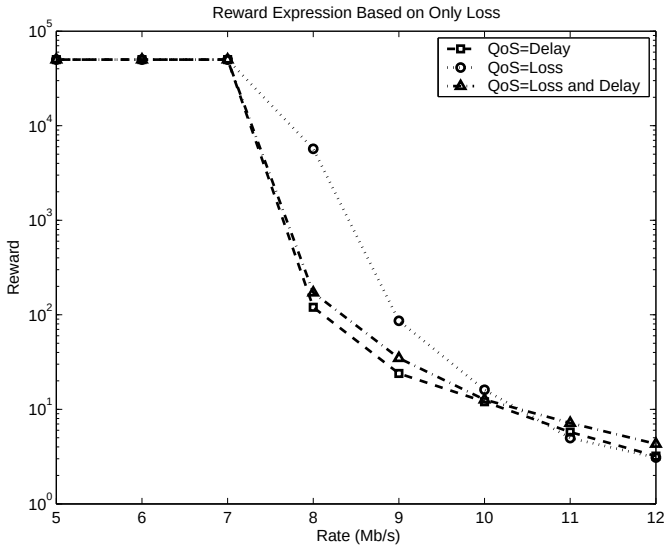

Figure 4.6: Reward Expression Based on Only Loss Traffic.

From the Figure 4.4 and Figure 4.6, we find the curve based on both loss and delay is similar with the curve based only on delay. Thus, we try to explain the reason and draw the relationship among reward, loss and delay shown in Figure 4.7. We can see that the reward is more sensitive to delay than loss rate. That is why the curves based on both loss and delay are close to curves based on only delay.

4.5.2 Performance Evaluation with Background Traffic

We have known that if the sending rate is so large that the DPs will be sent out before SPs get enough time to learn the route conditions, then the performance of CPN will not be so good as expected. Therefore, we limit the rate of UDP traffic at a lower rate. However, if

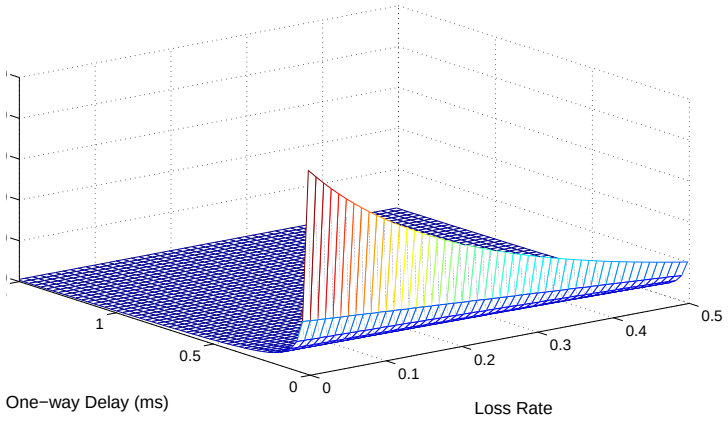

Figure 4.7: 3-D Reward Expression Based on Both Loss and Delay.

the rate is low and meanwhile there is only one UDP flow in the network, obviously the flow will not experience the congestion and will not lose packets in the wired network. Thus, we proposed to set up some background traffic so that we could observe the performance based on our different goal functions. The background traffic is UDP traffic. Every nodes send 6 Mbps UDP traffic to its neighbors while forwarding the regular packets. Again, as shown in Figure 4.9, using the goal function based only on loss results in the lowest observed loss rates. Using the goal function based only on delay or based on both delay and loss obtain the lower delay in Figure 4.8, and obtain smaller and smoother jitter in Figure 4.10 than the goal function based only on loss. The curve based only on delay and the curve based on both loss and delay are nearly identical. The reason is same with the one we explained in the performance without background traffic.

Again, Figures 4.11 and 4.12 summarize measurements from the perspective of QoS perceived by the user with background traffic. The purpose of these figures is to see whether the

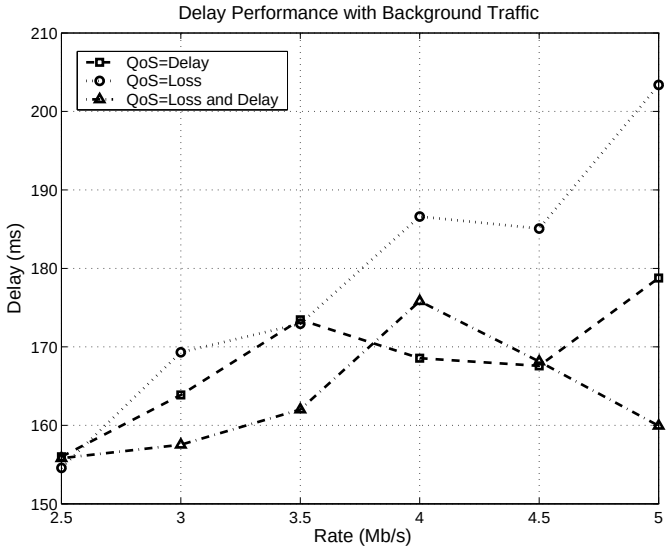

Figure 4.8: Delay Performance with 6 Mbps Background Traffic.

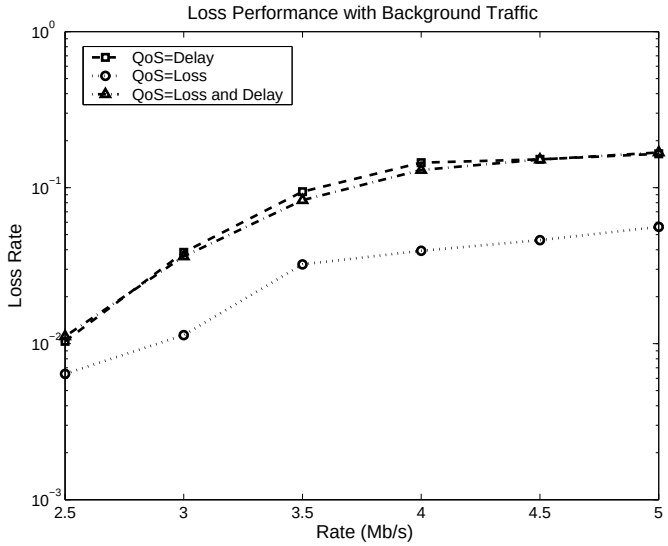

Figure 4.9: Loss Performance with 6 Mbps Background Traffic.

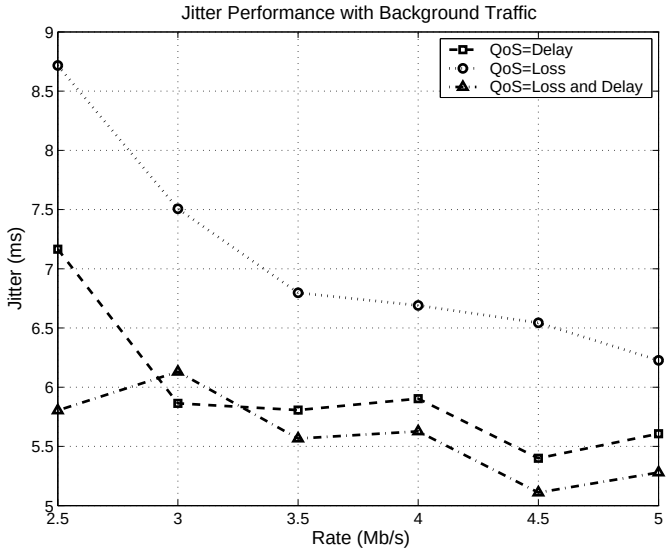

Figure 4.10: Jitter Performance with 6 Mbps Background Traffic.

end user is indeed obtaining the outcome in QoS that it is requesting under the background traffic just like what they get without background traffic. On the y-axis of the figures we show the reward function value at the end user (and not the numerical value that is used by the SPs, which operate at the lower network level). Similarly, the reward value is computed using the previous reward functions and using the measured values of loss and delay got from applications. In the Figure 4.12, we show the "loss based" reward for the user, as a function of user traffic. We see very clearly that the highest (therefore the best) reward is obtained if we use *only loss* in the goal function, while using both *delay* and *loss and delay* are worse but equivalent from the user's perspective. In Figure 4.11, we see that using *delay* as the QoS goal or using *loss and delay* as QoS goal provide the user with the greater rewards than using *only loss* as the QoS goal. In general, we do see that using a goal function which is

specifically tuned to the user's needs is justified since in some cases it will have a definite effect, while in other cases it will not be detrimental to user perceived QoS.

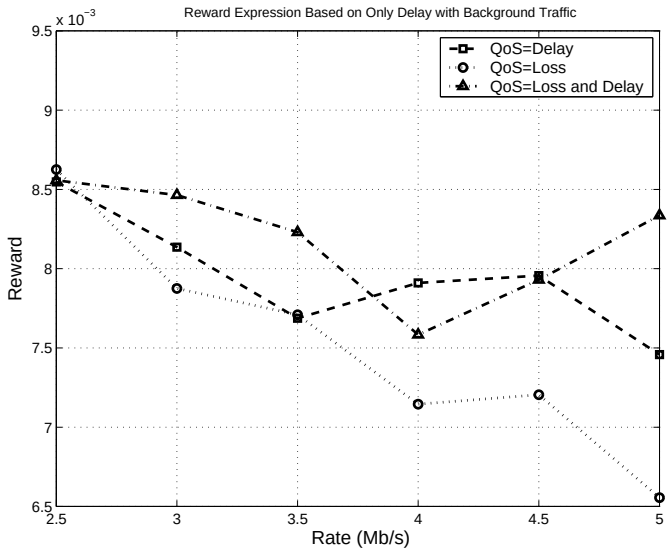

Figure 4.11: Reward Expression Based on Only Delay with Background Traffic.

4.6 Summary

In this chapter, we we have proposed an approach to construct composite goal functions which combine packet loss and delay into a single goal function used for CPN routing. We also presented different reward functions for applications with different QoS requirements. In addition, we have conducted experiments on CPN test-bed.

In a set of measurements, we have observed that when congestion happens, the use of delay by itself in the routing goal function results in better *delay* and *jitter* characteristics

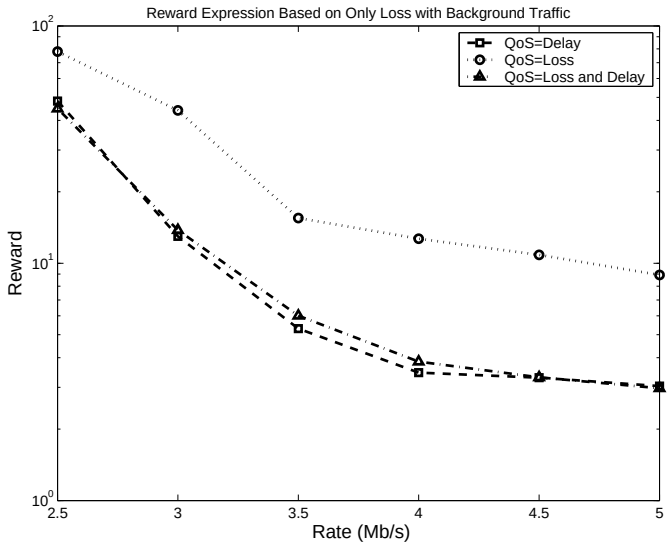

Figure 4.12: Reward Expression Based on Only Loss with Background Traffic.

for DPs. Similarly, using loss alone in the goal function provides the smallest loss rate for DPs. Nonetheless, when the network is saturated, we see that all goal functions (with or without both metrics) essentially result in very high delay and loss rate. Also we observed using delay alone or combination of loss and delay in the goal function does not seem to have a significant difference. In the experiments with background traffic, the similar results were obtained. Using delay alone results in smallest delay and jitter, and using loss alone results in the smallest loss. Thus our experiments support the claim that there is a good correspondence between the QoS goal that the SPs use to find paths and the resulting QoS observed by the users' payload.

CHAPTER 5

QOS-BASED RATE CONTROL (QRC) SCHEME

5.1 Overview

In this chapter, we will describe our proposed QoS-based Rate Control (QRC) scheme. It is built in the transport layer and specially designed to work with the Cognitive Packet Network (CPN) [GLX00b] and to support users'QoS while being TCP-friendly. It is a flow and congestion control scheme for unicast applications. We explore two different rate control functions to match applications with different QoS requirements, e.g. limiting a loss bound or jitter bound. In contrast with the existing congestion control schemes we will introduce in Section 5.2, these new schemes use new rate equations and deploy loss rate and jitter as congestion signal respectively. We this describe our proposed flow and congestion control algorithms in depth in Section 5.3 through 5.6.

5.2 Related Work

5.2.1 TCP-friendly Flow and Congestion Control

TCP congestion control is appropriate for applications such as bulk data transfer, but it is not well-suited for real-time audio and video applications, because the fact that is it inherently conservative and experiences abrupt changes in transfer rate can reduce the perceived quality for streaming multimedia applications. Therefore, various TCP-friendly [FHP00] congestion control mechanisms have been proposed for these applications. We will introduce some related work regarding these TCP-friendly flow and congestion control schemes in the following sections.

We can divide the TCP-friendly congestion controls into two classes: window-based and rate-based schemes. The window-based congestion control uses a slide window like the one used by TCP to control the sending rate. Its advantage is that it can provide the window-oriented fairness to TCP, but it is also easy to cause packet bursts and rate fluctuation. Based on some control theoretical results from [Mas99], Grieco *et al.* proposed a new slide window control function [GM04a] to react to congestion for TCP-friendly flows. They also proposed a quick probing function corresponding the TCP slow-start phase, and a gentle probing function corresponding the TCP congestion avoidance phase. The simulation results showed the ARC remarkably improves the goodput with respect to TFRC [FHP00] and Reno TCP in the presence of wireless lossy links.

5.2.2 Rate-based TCP-friendly Congestion Control

Much research has aimed to use the exact rate to control the sending rate, but the challenge of these rate-based congestion controls is to find a suitable target rate. A lot of schemes have been explored for the target rate from different views. These schemes can be divided into 3 classes: AIMD, throughput equation, and bandwidth. AIMD mimics the TCP AIMD behavior by additive increase and multiplicative decrease of its sending rate. Obviously this scheme keeps the TCP drawback of abruptly backing off in response to a single congestion indication. Some authors investigated the control rate function by probing the available bandwidth. Liu *et al.* [LH03] proposed to use bi-weighted moving average to detect trends in packet size and one-way delay so as to estimate the available bandwidth based on those trends. Among all the throughput-based TCP-friendly congestion controls, TFRC [FHP00] is one of the most widely investigated due to its fairness and relative smoothness. TFRC uses a TCP throughput equation [PFT98] in a steady state to control its sending rate in congestion avoidance. It also introduces the concept of a "loss event" to react to the multiple congestion indications. Many simulations and experiments have been done to evaluate the performance of TFRC.

The TCP-friendly Rate Control Protocol (TFRC) proposes to use TCP throughput r_{TCP} in steady state [PFT98] shown in Equation (5.1) as a target rate to adjust its transmission

rate so as to obtain a fair throughput compared to a TCP connection.

$$r_{TCP} = \frac{MTU}{RTT\sqrt{\frac{2p}{3}} + T_o min(1, 3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{5.1}$$

Where MTU is the packet size, p is the steady state *loss event rate*, T_o is the TCP retransmission timeout value, and RTT is the round trip delay.

The performance of TFRC is investigated in a lot of papers [BFS02, HK, YKL03], and many simulation results claim that TFRC has good fairness and relative smoothness. In [Aka04], the window-based and rate-based congestion control schemes are compared and it is concluded that rate-based schemes are more appropriate for real-time applications.

5.2.2.1 Performance of Evaluation on TFRC

The performance of TFRC over a 2.5G network was experimentally evaluated in [BFS02]. Hassan and Kara proposed a metric called rate-uniformness to describe transfer rate variation and evaluated the performance of TFRC with RED configuration for transferring high-quality video in the Internet by simulation in [HK]. Coefficient of variation (CoV) was used to quantify the smoothness and fairness of protocols in [YKL03], and researchers evaluated via simulation the smoothness, fairness and responsiveness of TCP and three TCP-friendly congestion control protocols: GAIMD, TFRC and TEAR by network environment changes. Bansal *et al.* [BBF01] investigated the behavior of slowly-responsive TCP-friendly congestion control algorithms under more realistic dynamic network conditions and concluded that most

of the TCP-compatible algorithms they studied are safe for deployment by analyzing the persistent loss rates, long- and short-term fairness properties, bottleneck link utilization, and smoothness of transmission rates.

5.2.2.2 Support for QoS

We have discussed the TFRC and some related research which aimed to improve the fairness and smoothness. With the emergence of different multimedia applications, more and more applications have different QoS requirements. Thus, much research begins to consider the factor of QoS. In [BRS02], the jitter factor is considered, and the authors propose to modify the RAP [RHE98] and TFRC to satisfy the Voice over IP (VoIP) applications's QoS . A extended TFRC supporting users' QoS is presented in [WMM00]. The authors argue that the original fair share of link bandwidth pursued by TFRC can not lead to the fairness in terms of applications-level QoS. In other words, the application-level qualities of service which users directly perceive are not necessarily identical even if they use the same amount of bandwidth. Therefore, the authors propose to estimate the application-level QoS from the estimated throughput, and then determine the sending rate.

5.2.2.3 Other TCP-friendly congestion control

Several congestion control protocols made improvements based on TFRC. An adaptive TCP-friendly rate control protocol (ATFRC) [CWL] was proposed dividing the loss indication into triple duplicate ACKs without timeouts and timeout occurrence to reflect the transient behavior of TCP, and it claimed that this approach of using different sending rate control functions for the two loss events indications can provide faster responsiveness than original TFRC. In [LSL03] authors discussed the effects of loss event rate on the sending rate in TCP friendly Equation (5.1) and argued that TFRC was very sensitive to low packet loss rate, and therefore proposed an logarithm-based TFRC (L-TFRC) rate control mechanisms to improve the smoothness of multimedia streaming. Also, based on the TCP steady throughput Equation (5.1), a smooth and fast rate adaption mechanism (SFRAM) was presented, and it was experimentally demonstrated that it was able to adjust its transmission rate in a very smooth manner by applying proposed weighted RTT, RTO and loss rate schemes[KKK04].

In [BJC03, BJC02] authors proposed a TCP-friendly rate adaptation algorithm based on loss (TRABOL) for real-time radar application with high bandwidth demand, where it adjusted the sending rate by emulating the AIMD scheme, and detected congestion by comparing the loss rate calculated at the receiver to the maximal loss value. Their experiments showed that the TRABOL increased the average throughput of the application during lightly loaded conditions while being friendly to TCP. A transport protocol called rate-control snoop [SS03] was proposed for heterogeneous networks with wired and wireless links. It uses the

rate-based transmission control to prevent packet bursts, and uses the ratio of the observed sending rate at the receiver to the actual sending rate at the sender as a primary metric to detect congestion and linearly adjust the sending rate. It showed better throughput than TCP Reno by simulation.

5.3 Design Considerations

Our proposed QRC scheme is designed for bandwidth- and jitter-sensitive applications. According to the characteristics of these applications and advantages of CPN, we address some concerns of designs as follows:

- **Rate-based** approaches are more suitable than window-based approaches for applications that transmit audio or video. Real-time applications are usually rate-based traffic. Also, window-based updates may cause packet burst, which is not desirable for real-time applications and results in short-time congestion.

- **QoS constraints** should be considered in the control. Applications with different QoS requirements should be treated differently. For some real-time applications which can tolerate low loss rate, low delay and low jitter, we are concerned about their throughput. For jitter-sensitive applications, we should monitor their jitter performance.

- **Fair bandwidth** should be distributed among competing connections. In particular, the traffic based on this adaptive control scheme should be friendly to TCP traffic.

- **Smooth rate** adaption should be deployed. Packet burst not only causes short-term congestion but also increases the end-to-end delay and jitter so as to affect the perceived quality by users. Transport protocols should adjust the sending rate smoothly and project packets into the network as evenly spaced as possible.

- **New congestion signs** should replace "duplicate ACKs" as indication of packet loss in CPN. Most of existing flow and congestion control protocols use duplicate ACKs as an indication of packet loss, which is suitable for the packets that take the same route. However, the routes for packets of same flows in CPN are always changed and packets are easily unordered, and thus duplicate ACKs may not indicate packet loss.

- **Retransmission** should only be done when necessary. Conservative retransmission could lead to unacceptable end-to-end delay. In addition, real-time applications can bear a low loss rate. It is preferable to retransmit the data only when it is required, which also can help reducing the overall load on the network.

5.4 Feedback Mechanism

The congestion detection scheme has to depend on feedback information. Traditional IP just provides the best-effort connectivity service and has no any feedback information to its upper layer. Thus, the transport layer, such as TCP and UDP, has to build their individual feedback mechanism. TCP has its ACK segments and can detect loss through duplicate

ACKs. However, UDP does not have such ACK packets, which brings some difficulties for UDP source to detect congestion. Therefore, several congestion control schemes for UDP [KHF04] were proposes to introduce ACKs to UDP, which is used for congestion control purpose only.

CPN provides an adaptive routing mechanism different from the traditional IP routing as we described in Chapter 3. Thus, the transport layer can use the information from routing layer in CPN. Whenever a DP's ACK arrives at the DP's source node, its corresponding flow information, such as forwarding loss rate, delay or jitter, will be estimated and stored according to the flow identification in the packet header. These values can be accessed by the transport layer. At the routing level, QoS information, such as loss and delay, is collected with three constraints: source node, next hop and destination node. At the transport level, we are only concerned about the end-to-end information. Thus, we need a way to formulate the feedback information from the routing level and provide congestion parameters for its transport level.

5.4.1 Congestion Degree

TCP uses *duplicate ACKs* as indication of congestion. It is good for applications that need reliable service, but it is too conservative for the loss-tolerant real-time applications. In addition, TCP halves its congestion window size whenever it gets duplicate ACKs, which is too abrupt for real-time applications that need smooth transmission rate. Therefore,

TFRC proposes a *loss event rate* to formulate the congestion and the loss event rate is often very low, thus TFRC can avoid changing its rate when only one loss event occurs. Besides introducing the concept of loss event rate, TFRC employs the TCP throughput equation in long-term to control the source sending rate. By this approach, the transmission rate can be very smooth in the steady state where RTT and loss rate have no big changes. However, once the loss rate is high, the throughput of TFRC applications is very low.

In this chapter, we introduce a new concept, *congestion degree*. We use this variable to describe the degree of congestion in the network. We use loss, delay, or jitter to formulate the congestion degree. Different flows with different QoS requirements can select different congestion degree formulation to adapt their packet rate projection to the network. We explore the end-to-end loss rate and jitter as QoS parameters in rate control functions, and we use the Equation (5.2) and low-pass filter approach shown in Equation (5.3) to estimate the end-to-end forward loss rate and the smoothed loss rate, respectively. The detailed deductions of Equation (5.2) is described in Chapter 3. In addition, when a flow starts, we use the Equation 3.9 to measure the loss rate.

$$L_f = \sqrt{1 - \frac{K}{N}} \tag{5.2}$$

where L_f is the loss rate from the source to the destination, which reflects the *congestion degree* that packets of a flow experienced, K is the number of DP's ACKs arrived at the

source, N is the number of DPs sent from the source and β is a constant.

$$\overline{L_f} \leftarrow \alpha\overline{L_f} + (1-\alpha)L_f \tag{5.3}$$

where $\overline{L_f}$ is an average forward loss rate and α is a constant.

Jitter is also called delay variance. That is, J_i=$D_{i+1} - D_i$. However, these simple delay difference values may not be good to reflect *real* congestion in the network. Thus, we first use a low-pass filter shown in Equation (5.4) to obtain the smoothed delay $\overline{D}$, and then define the jitter J as shown in Equation (5.5) to describe the *congestion degree*.

$$\overline{D} \leftarrow \alpha\overline{D} + (1-\alpha)D \tag{5.4}$$

where $\overline{D}$ is the smoothed end-to-end delay from the source to the destination, D is an end-to-end delay experienced by that ACK's corresponding DP, and α is a constant.

$$J_{i+1} = \frac{\overline{D_{i+1}} - \overline{D_i}}{\overline{D_i}} \tag{5.5}$$

where $\overline{D_{I+1}}$ and $\overline{D}$ are two consecutive smoothed delay, and J_{i+1} is an end-to-end redefined jitter.

Here it is worthy of noting that we deploy loss rate and jitter as two different approaches to reflect congestion degrees with which applications with different QoS requirements are concerned. For those real-time applications with high-bandwidth requirements, we use loss rate as congestion degree indications, and we wish to limit their loss rate to a value below the maximal loss rate which they can tolerate. For those jitter sensitive applications, we also give the *threshold* value for the jitter that they can bear. In the following sections, we will describe the congestion control rate equations in details.

5.5 Loss-based QRC Scheme

We explore a congestion control mechanism for the real-time applications based on the loss rate feedback. The goal of this scheme is to achieve as much throughput as possible and meanwhile to be friendly to TCP for those real-time applications which can bear low loss rate. This loss-based QRC scheme (QRC-L) uses additive increase and multiplicative decrease (AIMD) algorithm to adapt the rate, and is mainly implemented at the source. Window-based congestion control approach uses the ACKs to trigger the changes of congestion window size so as to adjust its sending rate. However, the similar way can not work on rate-based control mechanisms. Rate-based control approach uses the packet interval calculated through a rate equation to trigger sending packets. On the other hand, considering the loss rate we estimate is a cumulative value [SG04], we can not simply increase the sending rate when congestion degree is within a bearable value and decrease the sending rate otherwise. Different from TCP's AIMD two state policy, we introduce a triple-state control policy to adaptively change the sending rate.

5.5.1 Slow Start

At the beginning of a new connection, the source does not know the suitable rate to transmit the data. Therefore, the source first use a conservative low transmission rate R_o to probe the bandwidth. Usually the R_o is selected to be very low, thus it could not lead to congestion.

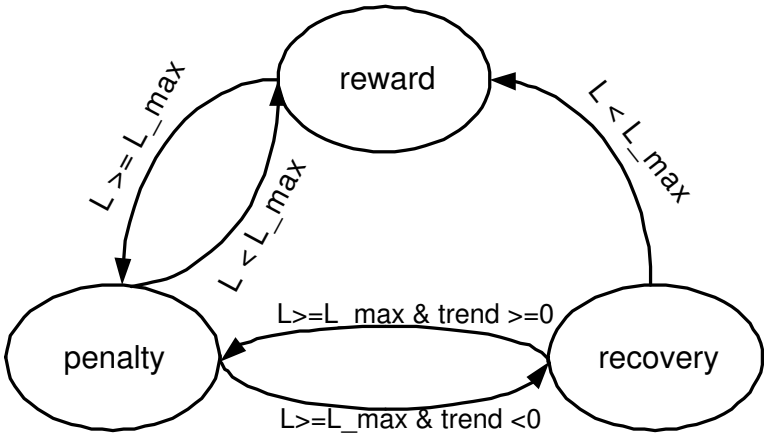

Figure 5.1: QRC-L State Transition Diagram.

Similarly to TCP, then the source *periodically double* increase the sending rate until it reaches a threshold value of sending rate R_s. Then the source enters our triple states.

5.5.2 Triple-state Control Policy

As shown in Figure 5.1, QRC-L source is a state machine model with three states: reward, penalty, recovery. In the following sections, we will describe its corresponding behavior in each of the above state.

Reward State Behavior

Assume the maximum loss rate value be L_{max}. When the packet loss rate is less than the maximum loss rate L_{max}, the source enters the reward state. Thus, according to the additive increase (AI) scheme, the transmission rate is *periodically* increased by factor K_a in a step-like fashion which is similar with RAP [RHE98]. However, we dynamically determine

the additive increase rate Equation (5.6) here. The smaller the current loss rate, the bigger its reward sending rate.

$$R_{i+1} \leftarrow R_i + K_a(1 - L_i) \tag{5.6}$$

Penalty State Behavior

When the source is in the reward state and suddenly the packet loss rate is larger than or equal to the maximum loss rate, the source will *immediately* enter penalty state and decrease its sending rate according to Equation (5.7). However, if the source is in the penalty state and the packet loss rate is larger than or equal to the maximum loss rate, the source will *not have to* decrease its sending rate again. As we are using a cumulative loss rate L as a congestion indication, such memory-based value can not reflect whether the network condition is becoming better. Therefore, a new variable, loss trend $\triangle$ will be considered. We define loss trend $\triangle$ in Equation (5.8). If $\triangle$ is larger than or equal to 0, we think loss rate is continuing to increase, which implies previous penalty probably does not work, so the transmission rate will be *periodically* decreased again; otherwise, it enters the loss recovery state.

$$R_{i+1} \leftarrow R_i * \frac{L_{max}}{L_i} \tag{5.7}$$

$$\triangle = \begin{cases} < 0, & \text{if } L_i < L_{i-1}; \\ \geq 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

Recovery State Behavior

Recovery state refers to the state where current loss rate L is still larger than or equal to the maximum value L_{max}, but it has a loss rate decreasing trend, which implies the network condition is becoming better or the previous penalty works. Therefore, the *greedy* source begins to probe the available bandwidth again by *periodically* increasing the sending rate *very slowly*. We use decision policy shown in Equation (5.9) where $K_r \ll K_a$:

$$R_{i+1} \leftarrow R_i + K_r(1 - L_i) \tag{5.9}$$

5.6 Jitter-based QRC Scheme

For those jitter-sensitive real-time applications, jitter is an important factor. Similarly we concern two problems when we design the scheme. On the one hand, we want the new scheme to minimize the jitter and adapt its sending rate according to the jitter condition in the network. In paper [SG04] we have known that the routing based on delay can achieve smaller end-to-end delay than the routing based on loss rate. When we minimize the delay, we also minimize the jitter. Therefore we use routing based on delay to minimize the jitter in the routing layer, and in the transport layer we use a jitter-based QRC scheme (QRC-J) to control the rate. On the other hand, we want the scheme to be TCP-friendly.

Our proposed congestion control uses an AIMD increase/decrease algorithm similar to TCP. Assuming the maximum jitter is ξ, we use a rate control Equation (5.10). The idea is that when the packet jitter is less than the maximum value ξ, the transmission rate is *periodically* increased by factor $\frac{\alpha}{D_{i+1}}$ where D_{i+1} denotes the one-way delay. The increase factor is deducted from AI scheme of TCP. As the TCP's window size increases by $W_{i+1} = W_i + 1$, we can get $\frac{W_{i+1}}{RTT} = \frac{W_i}{RTT} + \frac{1}{RTT}$. That is, R_{i+1} is approximately equal to the sum of R_i and $\frac{\alpha}{D_{i+1}}$. If the packet jitter is larger than or equal to the maximum value, the transmission rate is *immediately* decreased.

$$R_{i+1} = \begin{cases} R_i + \frac{\alpha}{D_{i+1}} & \text{if } J_{i+1} < \xi; \\ \\ \frac{\xi}{J_{i+1}} R_i & \text{otherwise} \end{cases} \tag{5.10}$$

where D_i is the estimated forward one-way delay, R_i is the sending rate at the source node, ξ is the jitter threshold, and α is constant.

5.7 Summary

We summarize QRC-L rate control function as follows:

$$R_{i+1} = \begin{cases} R_i + K_a(1 - L_i) & \text{if } L_i < L_{max}; \\ \\ R_i * \frac{L_{max}}{L_i} & \text{if } L_i \geq L_{max} \text{ and } \triangle \geq 0; \\ \\ R_i + K_r(1 - L_i) & \text{otherwise} \end{cases} \tag{5.11}$$

Here, i denotes every step. L_i denotes the estimated forward loss rate. R_i denotes the sending rate at the source node. The maximum loss rate L_{max} is the loss rate threshold. Other parameters have been described in the above text.

In the previous section, we have already described the rate equation of QRC-J. Both QRC-L and QRC-J use rate-based control function so as to be suitable for those rate-based real-time applications. Basic idea is to use AIMD model to be friendly to TCP. In addition, we consider the QoS metrics for real-time applications that are delay sensitive.

CHAPTER 6

QRC PERFORMANCE EVALUATION

6.1 Overview

We implement the schemes QRC-L and QRC-J in Linux 2.4. Figure 6.1 shows the flow of the process. Whenever a user sends a data from application layer, the system and socket write functions will be invoked. Then, it will call sendmsg() function to check the waiting time. The waiting time is calculated by the rate control function. If waiting time is equal to X units, then this thread will wait for X units. Here, the unit is equal to 10^{-5} second. The parameters will be passed to cpn-xmit-skb() function after X units and thus the routing layer begins to handle the packet. After a packet leaves the routing layer, its departure time will be stored.

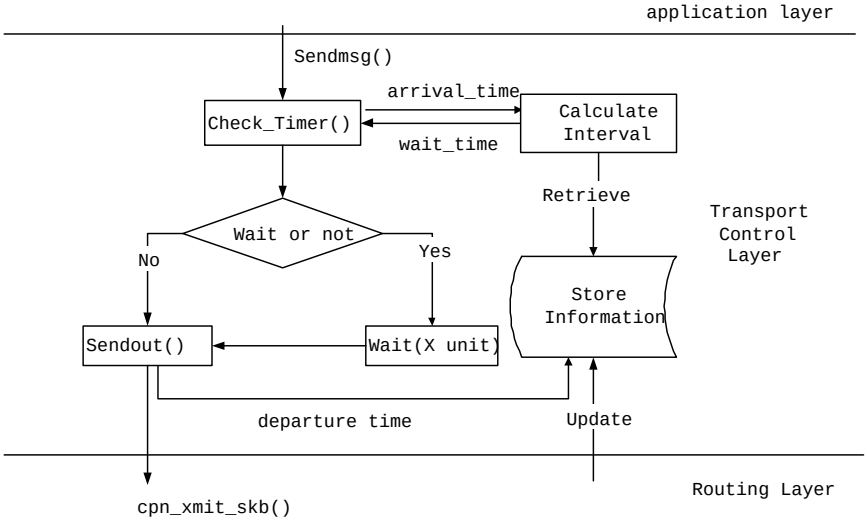

Figure 6.1: Flow Chart of Congestion Control

6.2 Interval Control Functions

Rate control is actually a "interval" control. This means the real implementation is to control sending rate by adjusting the interval of successive packets. In the Linux kernel, the smallest time unit is 10^{-5} second. Thus, for convenience we use 10^{-5} second/packet as a new unit and get the interval Equations (6.1) and (6.2). Here, I_i is the interval of two successive packets and its unit is 10^{-5} second/packet.

$$I_{i+1} = \begin{cases} max(I_{min}, I_i \frac{1}{1+I_i K_a (1-L_i)*10^{-5}}), & \text{if } L_i < L_{max}; \\ min(I_{max}, I_i \frac{L_i}{\alpha L_{max}}), & \text{if } L_i \geq L_{max} \text{ and } \triangle \geq 0; \\ max(I_{min}, I_i \frac{1}{1+I_i K_r (1-L_i)*10^{-5}}), & \text{otherwise} \end{cases} \tag{6.1}$$

$$I_{i+1} = \begin{cases} max(I_{min}, I_i \frac{1}{1+I_i \frac{\alpha}{D_{i+1}} * 10^{-5}}), & \text{if } J_{i+1} < \xi; \\ min(I_{max}, I_i \frac{J_{i+1}}{\xi}), & \text{otherwise} \end{cases} \tag{6.2}$$

We also list the parameters we used in our experiments in Table 6.1.

Table 6.1: Parameters in QRC Experiments

Variables	Description	Values in Experiments
L_{max}	Maximal Loss Rate	0.1
ξ	Maximal Jitter Degree	0.1
R_{min}	Minimal Sending Rate	1 Mbps
R_{max}	Maximal Sending Rate	10 Mbps
K_a	Increase Factor(reward)	0.5
K_r	Increase Factor(recovery)	0.2
α	Delay Adjust Factor (QRC-J)	1

6.3 Experiments Across One Hop

In order to evaluate our proposed QoS-based flow and congestion control protocol, we have done some experiments on a simple topology with one bottleneck link only shown in Figure 6.3 and on a complex topology with 26 nodes totally shown in Figure (6.6). All the links use 10 Mbps point-to-point Ethernet and the packet size is 1024 KByte. We use QRC-L, QRC-J to denote the network frame consisting of the proposed loss-based QRC scheme and loss-based routing scheme, and the network frame consisting of proposed jitter-based QRC scheme and delay-based routing, respectively. Besides, we use "iperf" [ipe] to generate TCP traffic and measure its throughput. In general, the system structure is shown in Figure 6.2.

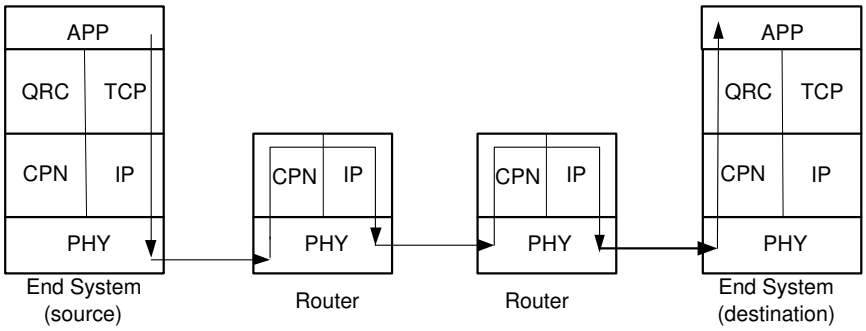

Figure 6.2: System Structure

In our first set of experiments, our goal is to (1) test if QRC-L and QRC-J can probe as much bandwidth as possible when they are alone in the network, respectively, and to (2) observe if QRC-L and QRC-J flows can be friendly to TCP when they share the single bottleneck bandwidth with TCP flows, respectively. We construct a simple topology as shown in Figure 6.3.

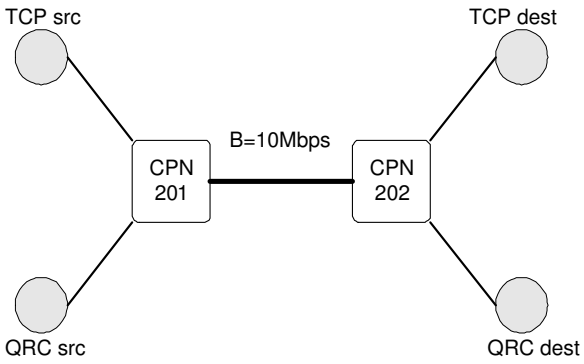

Figure 6.3: One Hop Topology

6.3.1 Throughput Comparison of TCP, QRC-L and QRC-J

This experiment examines if QRC-L only or QRC-J only can probe as much bandwidth as possible when it is an unique flow in the network. A single TCP flow first runs from the source node 201 to the destination node 202 as shown in Figure 6.3. From 10 seconds to 25 seconds, we observe that the throughput of this TCP flow keeps around 6.6 Mbps. Then, we introduce QRC-L and QRC-J flows respectively to the network from node 201 to 202 as we do for TCP flow. It is worthy of noting that here every QRC-L or QRC-J or TCP flow does not share bandwidth with others. Our goals is to examine how much bandwidth they can achieve when they are dominant in the network respectively. Here, we observe that the throughput of QRC-L is around 8 Mbps and loss rate is 0, and that the throughput of QRC-J is around 7.6 Mpbs , a little lower than QRC-L, at different transmission period from 10 seconds to 25 seconds. The detailed throughput results are shown in Table 6.2, which indicates both QRC-L and QRC-J flow can achieve better throughput than TCP flow when they are alone in the network.

Table 6.2: Throughput Comparison of TCP and QRC Alone

Time (s)	Throughput (Mbps)		
	TCP	QRC-L	QRC-J
10	6.704	7.893	7.513
12.5	6.528	7.898	7.619
15	6.720	8.000	7.565
17.5	6.670	8.001	7.605
20	6.600	8.003	7.608
22.5	6.613	8.005	7.596
25	6.624	8.007	7.606

6.3.1.1 TCP-friendliness of "UDP"

In this experiment we observe why UDP is not friendly to TCP. A TCP flow and a TCP or UDP flow are conducted to the network shown in Figure 6.3 to share the single bottleneck link and to compete bandwidth. First, we use *iperf* track the throughput of two TCP flows which run at same time and keep for the same time period. Their throughput results are recorded in Table 6.3, and we can see these two flows have very close throughput values (4 Mbps), which implies these two TCP flows are allocated fair bandwidth as the ethernet link is 10 Mbps. The reason is that these two TCP flows have congestion control scheme and the scheme is effective.

Table 6.3: Throughput of TCP and TCP Flows Sharing a Bottleneck

Time (s)	Throughput (Mbps)	
	TCP(1)	TCP(2)
10	4.19	3.91
12.5	4.10	3.90
15	4.05	3.85
17.5	4.06	3.92
20	4.03	3.93
22.5	4.07	3.93
25	3.99	3.91

Then, we observe the throughput of a TCP flow sharing bandwidth with a competing UDP flow. Here, TCP flows traversers IP layer and UDP flows traversers CPN layer. The non-congestion-controlled UDP flow tries to send out packets as many as possible and its transmission rate is around 10 Mbps. As it has no any response to congestion, it keeps the same sending rate even if it experiences high loss rate (e.g. 0.18) shown in Table 6.4.

Meantime TCP flow reacts to congestion and finally ends up with very poor throughput (less than 1 Mbps).

Table 6.4: Throughput of TCP and UDP Flow Sharing a Bottleneck.

time (s)	TCP	UDP	
	throughput (Mbits/s)	throughput (Mbits/s)	loss rate
10	0.523	10	0.18
12.5	0.319	10	0.17
15	0.245	10	0.17
17.5	0.213	10	0.17
20	0.185	10	0.18
22.5	0.174	10	0.18
25	0.191	10	0.18

We use "fairness index" f defined in Eq. (6.3) to quantize the fairness and compare the fairness index of non-congestion-control UDP flow (non-CC) with TCP flow in Figure 6.4.

$$f = \frac{(\sum_{i=1}^{n} x)^2}{n \sum_{i=1}^{n} x^2} \tag{6.3}$$

Here x is the throughput of every flow which exists with TCP flows.

6.3.2 TCP-friendliness of QRC-L and QRC-J

The second set of experiments examines the TCP-friendliness of QRC-L and QRC-J. At the same time, QRC-L or QRC-J and TCP traffic were introduced from node 201 to node 202. We observe the throughput at the different transmission period. Table 6.5 summarizes the throughput, loss rate and jitter of QRC-L flow and the throughput of its competitive

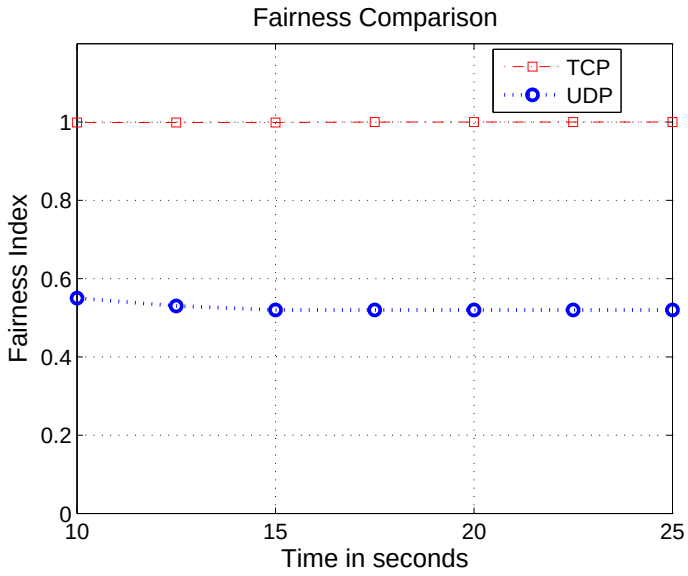

Figure 6.4: Fairness Indexes of TCP and UDP

neighbor TCP flow. Table 6.6 summarizes the throughput, loss rate and jitter of QRC-J flow and the throughput of its competitive neighbor TCP flow. We can see that QRC-L and TCP share the bandwidth fairly. So do QRC-J and TCP. Although QRC-L has some loss (loss rate >0), it does not exceed the maximal loss rate ($L_{max} = 0.1$). In contrast, QRC-L adjusts its sending rate in time reacting to the congestion, which leads to that QRC-L traffic and TCP traffic share the bandwidth fairly. The jitter achieved by QRC-J is a little lower than the jitter achieved by QRC-L, but both of their jitters are very low (2-2.5ms). In addition, from the Figure 6.5 we can see that the fairness index is mostly close to 1 when the TCP and QRC-L or TCP and QRC-J are neighbors, which is much better than TCP and UDP.

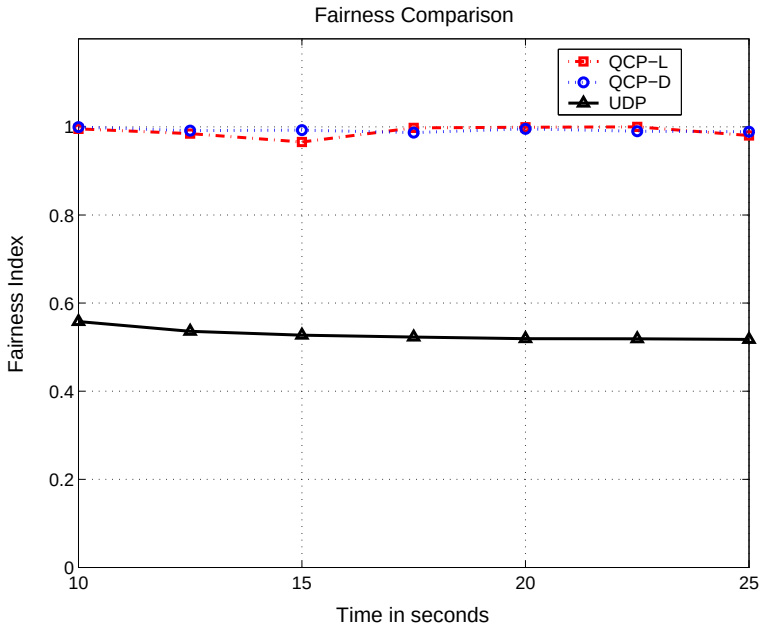

Figure 6.5: Fairness Indexes of QRC-J, QRC-L and UDP

Table 6.5: Performance of Network Shared by TCP and QRC-L Flows

time (s)	TCP	QRC-L		
	throughput (Mbits/s)	throughput (Mbits/s)	loss rate	Jitter (ms)
10	3.73	4.26	0	2.25
12.5	3.45	4.43	0	2.21
15	3.23	4.14	0.0014	2.7
17.5	3.77	3.40	0.001	2.46
20	3.77	5.86	0.005	2.55
22.5	3.86	3.86	0.004	2.64
25	3.35	4.45	0.002	2.20

Table 6.6: Performance of Network Shared by TCP and QRC-J Flows

time (s)	TCP	QRC-J		
	throughput (Mbits/s)	throughput (Mbits/s)	loss rate	Jitter (ms)
10	4.08	3.924	0	2.1
12.5	3.63	4.363	0	2.4
15	3.62	4.306	0	2.3
17.5	3.54	4.456	0	2.4
20	3.64	4.168	0	2.3
22.5	3.59	4.385	0	1.98
25	3.50	4.329	0	1.96

6.4 Experiments Across CPN Test Bed

Across the small test-bed with only hop, we have already observed that both QRC-J and QRC-L behave friendly with TCP. Besides, they have both very low loss rate (e.g. close to 0) and low jitter (2-2.5 ms). This is also because the source adjusts its sending rate timely so that it avoids the congestion.

Next we conduct the experiment based on the larger test-bed shown in Figure 6.6. Again, the source node is 201 and the destination node is 219. For TCP traffic the routing path is $201 \rightarrow 202 \rightarrow 203 \rightarrow 204 \rightarrow 205 \rightarrow 206 \rightarrow 212 \rightarrow 218 \rightarrow 219$. However, QRC traffic uses CPN routing to route packets. Thus, the routing paths are dynamically selected. However, TCP and QRC traffic have same source and destination. These two nodes also probably cause congestion. As the routing paths are different, we expect QRC-J and QRC-L show different end-to-end performance. We observe QRC-J and QRC-L both obtain zero loss rate. Then we evaluate their delay shown in Figure 6.7 and jitter shown in Figure 6.8 and we find QRC-J achieved both lower delay and jitter than QRC-L.

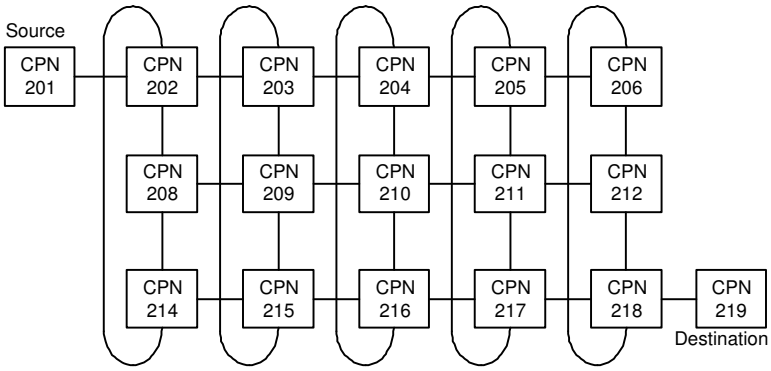

Figure 6.6: Test Bed Topology of QRC and CPN Framework

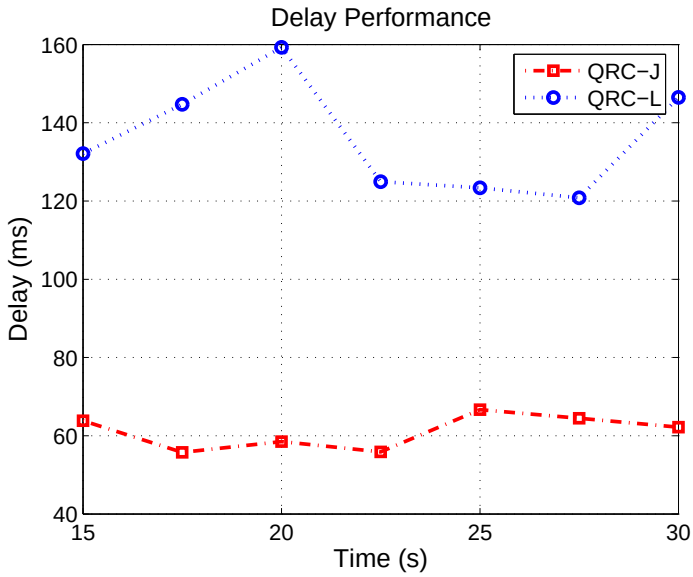

Figure 6.7: Delay based on different rate control schemes on large test-bed

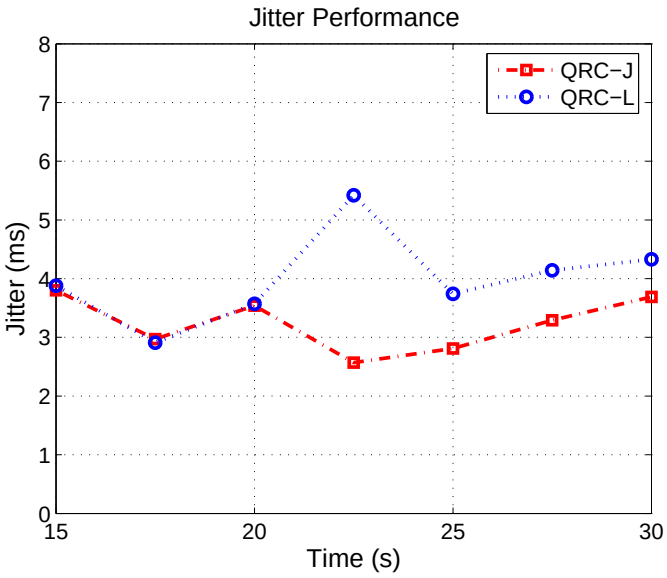

Figure 6.8: Jitter based on different rate control schemes on large test-bed

6.5 Summary

In this chapter, we conducted a set of experiments that show that the flows based on proposed congestion control schemes (QRC-L and QRC-J) can achieve better throughput when they are dominant in the network than TCP, and that both QRC-L and QRC-J flows show friendliness to TCP flows. In the small test-bed where only one hop exists, QRC-L and QRC-J get similar low loss rate and low jitter, but in a larger test-bed, QRC-J achieves lower delay and lower jitter than QRC-L, which again support our previous claims that our proposed routing based on delay minimizes the delay and jitter. Low loss rate, low jitter and friendliness also supports our claims that QRC scheme supports real-time applications' QoS and meantime is friendly to TCP.

CHAPTER 7

SUMMARY AND FUTURE WORK

In this chapter, we will summarize our dissertation work and suggest some further areas of research.

7.1 Summary

In this dissertation, we explored two schemes: routing, flow and congestion control to achieve QoS in the Cognitive Packet Network (CPN). Usually a router is concerned about how to select one adjacent router from its neighbors to forward the packets. Shortest path routing is to select an adjacent router which can send the packets from current router to get it as quickly as possible to its eventual destination. QoS routing is to find a route satisfying the given QoS constraints, which could minimize the end-to-end delay, loss probability or maximize the bandwidth. We introduced some existing QoS architectures on the Internet and analyzed the end-to-end flow and congestion control schemes in Chapter 2.

The design and implementation of CPN architecture have been presented by our colleagues. CPN uses SPs to explore the routes based on different goals, and DPs to carry payload by following the routes which are found by SPs. RNNRL algorithm is another core of CPN. A series of experimental results have shown that CPN's adaptive routing can dynamically select an adjacent node based on different goals. We made a simple introduction about the architecture and reinforcement learning algorithm of CPN in Chapter 3. Based on the CPN framework, we investigated a few approaches to achieve QoS.

We addressed an approach to estimate the online loss rate, and we discussed the online loss rate in short-term and steady state respectively. Usually synchronization is an obstacle for measuring online loss rate. In our approach, we avoided the synchronization on different nodes, and estimated a memory-based online loss rate by assuming the forward loss rate was proportional to the backward loss rate. This assumption was based on the special features of CPN. The proposed approach was used in CPN adaptive routing and flow and congestion control.

We proposed and formulated the users' QoS requirements in CPN goal functions. On the one hand, these different goal functions reflected different users' QoS needs. On the other hand, they were effectively cooperated with the reinforcement learning algorithm in CPN to calculate suitable rewards for making routing decisions. We investigated using end-to-end delay or loss rate as single QoS constraint to make adaptive routing decisions, and explored using both loss and delay as 2-dimension QoS constraints to route packets. A series of experiments were done and supported our claim that there was a good correspondence

between the QoS goal that the SPs use to find paths, and the resulting QoS observed by the users' payload.

We also proposed and implemented a QoS-based Rate Control (QRC) scheme. It is an end-to-end flow and congestion control scheme in UDP transport layer to support users' QoS requirements for real-time video and audio applications. It is known that TCP existing flow and congestion control is very robust and efficient for those applications which need reliable services, but it is poor for those real-time applications. According the QoS requirements of these real-time applications, we designed QRC scheme that is built in UDP transport layer to support users' QoS requirements. Our QRC scheme was designed specially for CPN framework. Thus, we addressed and discussed some specific design considerations compared with existing schemes. We considered two different QoS constraints, jitter and loss rate. According to these two constraints, we proposed two rate control equations. Experimentally we showed that the proposed algorithms can probe more available bandwidth to satisfy the users' QoS than the the CPN without flow and congestion control and meantime be friendly to TCP.

7.2 Future Work

Although CPN has shown attractive performance compared with the traditional packet switching network, there are still some aspects that can be further improved and evaluated:

Considering the self-adaptation of CPN, we can explore more QoS constraints and map them into different goal functions to support QoS routing in CPN. In addition to minimizing the loss rate or end-to-end delay that have already been presented in this dissertation, other important constraints, such as bandwidth, the available queue length of every router, the remaining power at every router in wireless network, can be investigated further. Also, we can extend our approach to more general multi-dimensional QoS goals, such as the need to maintain the QoS of a connection within a defined boundary defined by values of several different metrics such as loss, delay, jitter, and including the effect of power limitations. We will also consider the use of priority schemes for packets within CPN routers. These extensions will provide a much broader framework within which network self-monitoring and self-awareness can be exploited dynamically to obtain best-effort QoS to users.

In our proposed QoS-based Rate Control (QRC) schemes, we used the loss rate and jitter that a flow experienced in the network as congestion signs. For example, in the loss-based QRC scheme, we considered the cumulative loss rate as a QoS bound and used the loss trend to judge the network conditions to adjust the sending rate. In the jitter-based QRC scheme, we considered the defined jitter as a QoS boundary and adjusted the transmission rate. Both loss rate and jitter reflected the attributes of flows in a network. While these algorithms at the transport endpoints can ensure the network capacity be not exceeded by limiting the input numbers of flows, they cannot tell the congestion loss with the error loss, and they cannot ensure fair sharing of that capacity. Only in routers, at the convergence of flows, is there enough information to control sharing and fair allocation. Thus, we can explore the

explicit notification from the network, rather than only depending the information collected at transport endpoints.

Furthermore, we can expand our design to multicast QoS routing in the future. In our current design and implementation, we only studied the unicast QoS routing. It is a promising area to explore multicast QoS routing in large-scale, wide-area networks. In order to avoid the unacceptable delay brought by retransmission, our current rate control does not include error control. However, we should investigate error probabilities and refine the network loss rate in the future.

LIST OF REFERENCES

[ADL95] Jong Suk Ahn, Peter B. Danzig, Zhen Liu, and Limin Yan. "Evaluation of TCP Vegas: emulation and experiment." In *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 185–195, New York, NY, USA, 1995. ACM Press.

[AGK98] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi. "Quality of Service Based Routing: A Performance Perspective." In *ACM SIGCOMM*, pp. 17–28, 1998.

[Aka04] B. Akan. "On the throughput analysis of rate-based and window-based congestion control schemes." *Comput. Networks*, **44**(5):701–711, 2004.

[AOM02] James Aweya, Michel Ouellette, and Delfin Y. Montuno. "A self-regulating TCP acknowledgment (ACK) pacing scheme." *Int. J. Netw. Manag.*, **12**(3):145–163, 2002.

[AR03] Alhussein A. Abouzeid and Sumit Roy. "Stochastic modeling of TCP in networks with abrupt delay variations." *Wirel. Netw.*, **9**(5):509–524, 2003.

[BBF01] Deepak Bansal, Hari Balakrishnan, Sally Floyd, and Scott Shenker. "Dynamic behavior of slowly-responsive congestion control algorithms." In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 263–274, New York, NY, USA, 2001. ACM Press.

[BFS02] D. Beaufort, L. Fay, C. Samson, and A. Teil. "Measured performance of TCP friendly rate control protocol over a 2.5G network." In *Vehicular Technology Conference, Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, volume 1, pp. 563 – 567, Sept. 2002.

[BJC02] S.L. Bangolae, A.P. Jayasumana, and V. Chandrasekar. "TCP-friendly congestion control mechanism for an UDP-based high speed radar application and characterization of fairness." In *Communication Systems, 2002. ICCS 2002. The 8th International Conference on*, volume 1, pp. 164 – 168, 25-28 Nov. 2002.

[BJC03] Sangeetha L. Bangolae, Anura P. Jayasumana, and V. Chandrasekar. "Performance Evaluation of a Memory-Based TCP-friendly Rate Adaptation Algorithm

for a Real-time Radar Application." In *28th Annual IEEE Conference on Local Computer Networks (LCN 2003), 20-24 October 2003*, pp. 319–322, 2003.

[BOP94] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance." In *SIGCOMM*, pp. 24–35, 1994.

[BQL02] R. Banerjee, J. Quemeda, P. Lorenz, T. Braun, and B. Martinez:. "Mechanisms for Attaining QoS in IPv6-based Multimedia Internetworks." Technical report, 2002. NGNi project deliverable D2.

[BRS02] F. Beritelli, G. Ruggeri, and G. Schembra. "TCP-friendly transmission of voice over IP." In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 2, pp. 1204 – 1208, 28 April-2 May 2002.

[BZB97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. "Resource Reservation PRotocol (RSVP) Version 1: Functional Specification.", 1997.

[CGM02] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. "TCP westwood: end-to-end congestion control for wired/wireless networks." *Wirel. Netw.*, **8**(5):467–479, 2002.

[CN98] S. Chen and K. Nahrstedt. "An overview of Quality-of-Servicerouting for the next generation high-speed networks: Problems and Solutions." *Network Magazine*, 1998.

[CWL] Seongho Cho, Heekyoung Woo, and Jong won Lee. "ATFRC: Adaptive TCP Friendly Rate Control.".

[DR89] D.Chiu and R.Jain. "Analysis of the increase and decrease algorithmfor congestion avoidance in computer networks." *Journal of Computer Networks*, **17**(1):1–4, 1989.

[FDV99] F. Le Faucheur, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. "MPLS Support of Differentiated Services." Technical report, 1999.

[FF99] S. Floyd and K. Fall. "Promoting the use of end-to-end congestion control in the Internet." In *IEEE/ACM Transactions on Networking*, Aug. 1999.

[FH97] P. Ferguson and G. Huston. *Quality of Service:Delivering QoS on the Internet and in Corporate Networks*. Wiley Computer Publishing, 1997.

[FHP00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. "Equation-based congestion control for unicast applications." In *SIGCOMM 2000*, pp. 43–56, Stockholm, Sweden, August 2000.

[Flo01] S. Floyd. "A Report on Some Recent Developments in TCP Congestion Control." *To appear in IEEE Communications Magazine*, 2001.

[Gel93] E. Gelenbe. "Learning in the recurrent random neural network." *Neural Computation*, **5**(1):154 – 164, 1993.

[GG98] N. Giroux and S. Ganti. *Quality of Service in ATM networks: state-of-art traffic management*. Prentice Hall PTR, 1998.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman and Co., 1979.

[GLM02] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu. "Cognitive packet petworks: QoS and Performance." In *Proc. IEEE MASCOTS Conference*, San Antonio, TX, 2002.

[GLX00a] E. Gelenbe, R. Lent, and Z. Xu. "Towards networks with cognitive packets." In *Proc. IEEE MASCOTS Conference*, pp. 3–12, San Francisco,CA, 2000.

[GLX00b] E. Gelenbe, R. Lent, and Z. Xu. "Towards networks with cognitive packets." In *Proc. IEEE MASCOTS Conference*, pp. 3–12, San Francisco,CA, 2000.

[GLX01] E. Gelenbe, R. Lent, and Z. Xu. "Measurement and performance of Cognitive Packet Networks." *Comp. Networks*, **37**:691–701, 2001.

[GM04a] L.A. Grieco and S. Mascolo. "Efficiency, fairness and friendliness evaluation of TFRC and ARC rate-based congestion control." In *Control, Communications and Signal Processing, First International Symposium on 2004*, pp. 353 – 356, 2004.

[GM04b] Luigi A. Grieco and Saverio Mascolo. "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control." *SIGCOMM Comput. Commun. Rev.*, **34**(2):25–38, 2004.

[GMD99] E. Gelenbe, Z.H. Mao, and Y. Da-Li. "Function approximation with spiked random networks." *IEEE Transasctions on Neural Networks*, **10**(1):3–9, 1999.

[GSX99] E. Gelenbe, E. Seref, and Z. Xu. "Towards networks with intelligent packets." In *Proc. IEEE-ICTAI Conference on Tools for Artificial Intelligence*, Chicago, 1999.

[Gue99] R. A. Guerin. "QoS Routing in Networks with Inaccurate Information: Theory and Algorithms." *IEEE/ACM Transcations onNetworking*, **7**(3), 1999.

[HBG00] Urs Hengartner, Jurg Bolliger, and Thomas Gross. "TCP Vegas Revisited." In *INFOCOM (3)*, pp. 1546–1555, 2000.

[HK] S. Hassan and M. Kara. "Performance evaluation of end-to-end TCP-friendly video transfer in the Internet." In *Proceedings. Ninth IEEE International Conference on 10-12 Oct. 2001*, pp. 56– 61.

[HR00] E. Horlait and N. Rouhana. "Differentiated Services and Integrated Services Use of MPLS." In *Fifth IEEE Symposium on Computers and Communications*, Antibes, France, 2000.

[ipe] "http://dast.nlanr.net/Projects/Iperf/.".

[Jac88] V. Jacobson. "Congestion Avoidance and Control." *ACM Computer Communication Review*, **18**(4):314–329, 1988.

[Kel03] Tom Kelly. "Scalable TCP: improving performance in highspeed wide area networks." *SIGCOMM Comput. Commun. Rev.*, **33**(2):83–91, 2003.

[KHF04] E. Kohler, M. Handley, and S. Floyd. "Datagram congestion control protocol (DCCP)." *IETF Internet-Draft, draftietf- dccp-spec-08.txt*, 2004.

[KKK04] Young-Gook Kim, JongWon Kim, and C. C. Jay Kuo. "TCP-friendly Internet video with smooth and fast rate adaptation and network-aware error control." *IEEE Trans. Circuits Syst. Video Techn.*, **14**(2):256–268, 2004.

[KL01] M. Kodialam and T.V. Lakshman. "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information." In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pp. 376–385, Anchorage, AK , USA, 2001.

[KVR02] Lampros Kalampoukas, Anujan Varma, and K. K. Ramakrishnan. "Explicit window adaptation: a method to enhance TCP performance." *IEEE/ACM Trans. Netw.*, **10**(3):338–350, 2002.

[LH03] Qiang Liu and Jeng-Neng Hwang. "End-to-end available bandwidth estimation and time measurement adjustment for multimedia QoS." In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 3, pp. III – 373–6, 6-9 July 2003.

[LLM98] D.C. Lee, D.L. Lough, S.F. Midkiff, N.J. Davis, and P.E. Benchoff. "The Next Generation of the Internet: Aspects of the Internet Protocol Version 6." *IEEE Network*, **12**(1), 1998.

[LM05] Jung-Shian Li and Chung-Wen Ma. "Improving fairness of TCP Vegas." *Int. J. Netw. Manag.*, **15**(1):3–10, 2005.

[Low03] Steven H. Low. "A duality model of TCP and queue management algorithms." *IEEE/ACM Trans. Netw.*, **11**(4):525–536, 2003.

[LPW02] Steven H. Low, Larry L. Peterson, and Limin Wang. "Understanding TCP Vegas: a duality model." *J. ACM*, **49**(2):207–235, 2002.

[LSL03] Zhen Li, Guobin Shen, Shipeng Li, and E.J. Delp. "L-TFRC: an end-to-end congestion control mechanism for video streaming over the Internet." In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 2, pp. II – 309–12, 6-9 July 2003.

[Mas99] S. Mascolo. "Congestion control in high-speed communication networks using the Smith principle." In *Automatica, Speical Issue on Control methods for communication networks*, volume 35, pp. 1921–1935, December 1999.

[MNR03] Jim Martin, Arne Nilsson, and Injong Rhee. "Delay-based congestion avoidance for TCP." *IEEE/ACM Trans. Netw.*, **11**(3):356–369, 2003.

[MR01] Michael Mitzenmacher and Rajmohan Rajaraman. "Towards More Complete Models of TCP Latency and Throughput." *Journal of Supercomputing*, **20**(2):137–160, 2001.

[MS99] J. Mambretti and A. Schmidt. *Next Generation Internet*. Whiley Computer Publishing, 1999.

[Nic01] David M. Nicol. "Fluid simulation: discrete event fluid modeling of TCP." In *WSC '01: Proceedings of the 33nd conference on Winter simulation*, pp. 1291–1299, Washington, DC, USA, 2001. IEEE Computer Society.

[NZ02] S. Nelakuditi and Z. Zhang. "A localized adaptive proportioning approach to QoS routing granularity." *IEEE Comm.*, **46**(2):66–71, 2002.

[PFT98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. "Modeling TCP throughput: a simple model and its empirical validation." In *SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 303–314, New York, NY, USA, 1998. ACM Press.

[RHE98] R. Rejaie, M. Handely, and D. Estrin. "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet." In *Proc. IEEE Infocom'99*, 1998.

[RM86] D. E. Rumelhart and J. L. McClelland. *Parallel distributed processing Vols. I and II*. Bradford Books and MIT Press, 1986.

[RVC99] E. Rosen, A. Viswanathan, and R. Callon. "Multiprotocol label switching Architecture." Technical report, 1999.

[Sch96] M. Schwartz. "Broadband Integrated Networks.", 1996.

[SG04] P. Su and M. Gellman. "Using adaptive routing to achieve quality of service." *Performance Evaluation*, **57**:105–119, 2004.

[SKV01] B. Sikdar, S. Kalyanaraman, and K. Vastola. "Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK." In *IEEE GLOBECOM*, pp. 1781–1787, November 2001.

[SKV03] Biplab Sikdar, Shivkumar Kalyanaraman, and Kenneth S. Vastola. "Analytic models for the latency and steady-state throughput of TCP tahoe, Reno, and SACK." *IEEE/ACM Trans. Netw.*, **11**(6):959–971, 2003.

[SL02] Nishanth R. Sastry and Simon S. Lam. "A Theory of Window-Based Unicast Congestion Control." In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pp. 144–154, Washington, DC, USA, 2002. IEEE Computer Society.

[SRC84] Jerome H. Saltzer, David P. Reed, and David D. Clark. "End-To-End Arguments in System Design." *ACM Transactions on Computer Systems*, **2**(4):277–288, 1984.

[SS03] Young-Joo Song and Young-Joo Suh. "Rate-control snoop : a reliable transport protocol for heterogeneous networks with wired and wireless links." In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 2, pp. 1334 – 1338, March 2003.

[Sut88] R. S. Sutton. "Learning to Predict by the Methods of Temporal Differences." *Machine Learning*, **3**:9–44, 1988.

[SV03] Charalampos (Babis) Samios and Mary K. Vernon. "Modeling the throughput of TCP Vegas." In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 71–81, New York, NY, USA, 2003. ACM Press.

[Tan96] A. S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, 3 edition, 1996.

[WMM00] N. Wakamiya, M. Murata, and H. Miyahara. "On TCP-friendly video transfer with consideration on application-level QoS." In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 2, pp. 843 – 846, 30 July-2 Aug. 2000.

[WWL05] J. Wang, D. X. Wei, and S. H. Low. "Modeling and stability of FAST TCP." In *IEEE Inforcom, Miami, FL*, Miami, FL, 2005.

[YKL03] Yang Richard Yang, Min Sik Kim, and Simon S. Lam. "Transient behaviors of TCP-friendly congestion control protocols." *Computer Networks*, **41**(2):193–210, 2003.

[ZT03] Chi Zhang and Vassilis Tsaoussidis. "Improving TCP smoothness by synchronized and measurement-based congestion avoidance." In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pp. 131–140, New York, NY, USA, 2003. ACM Press.