MODELING AND ESTIMATION TECHNIQUES FOR UNDERSTANDING HETEROGENEOUS TRAFFIC BEHAVIOR

A Dissertation

by

ZHILI ZHAO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2004

Major Subject: Computer Engineering

MODELING AND ESTIMATION TECHNIQUES FOR UNDERSTANDING HETEROGENEOUS TRAFFIC BEHAVIOR

A Dissertation

by

ZHILI ZHAO

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

A. L. Narasimha Reddy
(Chair of Committee)

Dvahg Swaroop
(Member)

Riccardo Bettati
(Member)

Pierce E. Cantrell, Jr
(Member)

Chanan Singh
(Head of Department)

May 2004

Major Subject: Computer Engineering

ABSTRACT

Modeling and Estimation Techniques
for Understanding Heterogeneous Traffic Behavior. (May 2004)

Zhili Zhao, B.S., East China Normal University;
M.S., Shanghai Institute of Technical Physics, Chinese Academy of Sciences
Chair of Advisory Committee: Dr. A. L. Narasimha Reddy

The majority of current internet traffic is based on TCP. With the emergence of new applications, especially new multimedia applications, however, UDP-based traffic is expected to increase. Furthermore, multimedia applications have sparkled the development of protocols responding to congestion while behaving differently from TCP. As a result, network traffic is expected to become more and more diverse. The increasing link capacity further stimulates new applications utilizing higher bandwidths of future. Besides the traffic diversity, the network is also evolving around new technologies. These trends in the Internet motivate our research work.

In this dissertation, modeling and estimation techniques of heterogeneous traffic at a router are presented. The idea of the presented techniques is that if the observed queue length and packet drop probability do not match the predictions from a model of responsive (TCP) traffic, then the error must come from non-responsive traffic; it can then be used for estimating the proportion of non-responsive traffic. The proposed scheme is based on the queue length history, packet drop history, expected TCP and queue dynamics. The effectiveness of the proposed techniques over a wide range of traffic scenarios is corroborated using NS-2 based simulations. Possible applications based on the estimation technique are discussed.

The implementation of the estimation technique in the Linux kernel is presented in order to validate our estimation technique in a realistic network environment.

Adapted from the NS-2 implementation, the Linux implementation is tuned to produce accurate estimates under a realistic testbed. The correctness of our Linux implementation is corroborated through tests.

In order to investigate the feasibility of an approach to regulate the volume of incoming non-responsive traffic on an aggregate level by utilizing the estimation information, the performance evaluation of heterogeneous traffic under different queue management schemes and network configurations is studied. Our evaluation considers the aggregate bandwidth of different classes of traffic and the delays observed at the router. Our NS-2 based evaluation shows that queue management schemes (without per-flow or per-class state) do not provide significant control over the traffic mix consisting of both long-term and short-term traffic.

To my parents and my family

ACKNOWLEDGMENTS

I am indebted to my advisor, Dr. A. L. Narasimha Reddy. I am enlightened and benefited every time Dr. Reddy discusses problems with me. He has shared with me a great deal of his profound knowledge and professional manner of conducting research. Dr. Reddy supports me, gives me great advice in all aspects, and helps me to overcome obstacles all the way through my academic years at Texas A&M University.

I would like to express my gratefulness to Dr. Darbha Swaroop. I benefited greatly from Dr. Swaroop's academic support. He helped me to apply control theory to my research of network traffic estimation and regulation. He has been kind enough to give his time to discuss my research problems with me. I appreciate so much his kindness and greatest help to make the completion of my research work possible.

I am very thankful to Dr. Riccardo Bettai for giving me technical advice for my research work. Attending both of his challenging classes – Real-time System and Distributed Processing System – prepared me well to be confident in my research work.

It is a great honor for me to have Dr. Pierce Cantrell on my committee. His contributions in the networking area have extended beyond my reach. Although he is very busy everyday, as the associate provost for IT at Texas A&M University, when I request a meeting with him, he always arranges to see me at the earliest possible time. I am grateful for his enlightening academic and career guidance.

Special thanks goes to my friends and fellow graduate students. I would like to thank Jayesh Ametha for his initial work contributing to this research work; Yong Liu and I discuss academic issues often. He is giving me many valuable suggestions on my research work; Phani Acbhanta helped me with his rich Linux experience;

Zhuo Li discusses various mathematical issues with me. I would also like to thank Sumitha Bhandarkar for her academic support and valuable suggestions.

I owe a great debt of gratitude to my family. They encourage me, support me, help me, and are always there for me whenever I am down or succeed. Their support gives me a strong desire to excel. Because of them, I feel confident, happy, and strong enough to conquer any obstacle ahead of me and to succeed.

I would like to express my appreciation to all the staff in my department that gave me various help and support during these years.

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Since the first network, ARPANET, that was set up in 1969, the network technology has attracted significant interest. The increasing interest from researchers, along with higher demands from users, further stimuates the inventions of new physical layer technologies and the expansion of the network. Today, the Internet has reached hundreds of millions of users compared to only 4 hosts when it was first set up and the backbone capacity has grown from only 50kbps to over 10Gbps [1]. Various new applications, which may require higher bandwidth and different levels of Quality-of-Service, tend to grow. Those applications further fuel the evolution of the network. So the diversities of both network technologies and the network traffic are expected to increase.

With the observation of current statuses and trends of the network traffic and the network, our research focuses on understanding heterogeneous traffic behavior with the modeling methodology and applying developed models for practical uses.

A. Current Status and Trend of the Heterogeneous Network Traffic

Recent measurements shows that current network traffic consists of 10% - 20% UDP traffic and 80% - 90% TCP traffic [2]. While most UDP traffic is long-term, TCP traffic can be further classified as long-term or short-term. Short-term TCP flows(STFs), usually referred to as "web mice", occupy about 30% of total network traffic. Long-term flows, UDP or TCP, still contribute the majority of the load in current networks.

However, with the emergence of new applications, the usage of protocols other than TCP is expected to increase. For instance, several multimedia applications

The journal model is *IEEE Transactions on Automatic Control.*

rely on UDP to transport packets. Recently, there has been an increased interest in developing protocols that respond to congestion differently from TCP and provide smoother bandwidth to end applications [3, 4]. Furthermore, increases in bandwidth and computation power are expected to fuel the growth of multimedia applications that do not rely on TCP. These trends point to an increased diversity of network protocols and changes in the distribution of bandwidth among flows employing these diverse protocols in the future. Those protocols may not respond to the congestion as TCP does. The non-responsiveness of those protocols can be utilized to stage Denial-of-Service (DoS) attacks on end hosts and the network by pumping large amounts of non-responsive flows into the network. Some of the recent DoS attacks have used such "UDP floods". Here, the non-responsive traffic is defined as the type of traffic that does not respond to the congestion of the network by reducing its sending rate.

Also consider the example of the flash crowds. A flash crowd is a large volume of short-lived web traffic (short-term TCP traffic) trying to access a certain web site. It may cause the congestion along the path to the web site and also cause the web server to be overloaded and refuse further connection requests from other users. Those flash crowds consist of legitimate TCP flows, while the aggregate behavior of flash crowds is similar to that of DoS attacks. So the existence of short-term flows impacts the performance of network.

Because of the important role of short-term TCP flows on the netowrk performance, STFs have gained growing attention of researchers. Previous research has shown that: 1) the aggregated behavior of STF at the router appears to be non-responsive [5, 6]; 2) the performance of queue management schemes is different under a mixed workload with STFs when compared to workloads consisting of only long-term flows [7, 8].

The non-responsive traffic, including both network attacks and the flash crowds, causes various problems in the network. Those problems have been pointed out by authors in [9] and are summarized as follows:

- *Unfairness*

  The responsive TCP traffic suffers from less available bandwidth by competing with the non-responsive traffic, since the responsive flow is designed to reduce its sending rate at the time of congestion. With the increasing sending rate of the non-responsive traffic, the responsive traffic will stop transmission.

- *Congestion Collapse*

  The congestion collapse can occur when there are too many retransmitted packets from responsive flows in the network to overload the network, or when a large volume of undelivered packets arises from those non-responsive flows. Then the network resources will be wasted to try to deliver those packets which will be dropped before they reach the destination.

- *Exhausting Network Resources*

  As being brought out in the previous item, not only the network bandwidth but also the computation power and the memory of routers will be exhausted, while routers keep trying to deliver the large volume of non-responsive traffic.

B. Current Status and Trend of the Heterogeneous Network

Considering characteristics of the network, the heterogeneity of current network comes from asymmetric link bandwidths, non-congestion link errors, and larger bandwidth delay products [10].

The capacity of the backbone increases steadily due to new physical layer technologies, such as the DWDM technologies of high capacity optical fibers. This contribute to the increasing bandwidth delay product (BWDP) over the Internet.

The increased BWDP impacts largely on currently TCP-based traffic. The TCP protocol self-clocks using the estimated round trip time (RTT). The longer the RTT, the slower a TCP application sends packets and the lower the throughput will be. The increased BWDP also impacts the performance of queue management schemes. Since the buffer size of a queue management scheme in current routers is determined based on the BWDP of the link, the buffer size will increase with BWDPs. Consequently the queuing delay may also increase due to larger buffers. It is possible that future routers may be configured with smaller amounts of buffer than the traditional rule of one BWDP, since each interface card of a router may not be able to hold that much buffer memory at higher link capacities.

It is our interest to investigate the impact of larger BWDPs on the performance of TCP-base applications and queue management schemes. And if the routers are configured with smaller buffer sizes in the future, does this impact the decisions on buffer management algorithms? Do these trends impact the throughput of responsive and non-responsive flows differently? And does the early dropping of packets in RED allow non-responsive flows to force the responsive flows to realize smaller throughputs?

With the evolution of network technologies, the network itself becomes more and more heterogeneous in order to provide "increased productivity, performance, or throughput" [11]. And the evolution of network technologies gives new applications/services plentiful resources. Those new applications/services further stimulate the widespread use of the Internet. Internet users are able to gain the access to the network almost anywhere with wired or wireless connections. It is very convenient for legitimate users to connect to the network and use those new applications/services. At the same time, however, the convenience of access also makes it very easy for malicious users to launch network attacks to compromise hosts or to deny the service of end hosts or network to other users.

As we can see, the network traffic is becoming more and more heterogeneous due to the growth of new applications which employ protocols other than TCP. The proportion of traffic mix tends to change with the increase of UDP-based applications. The impact of non-responsive traffic on modern network is severe enough to warrant close attention. So it is necessary to investigate the impact of trends of the heterogeneous network traffic and network on both responsive traffic and queue management schemes.

C. Scope of the Presented Work

The trends in both network and Internet traffic attract researchers to investigate the potential impacts and possible solutions. But, as pointed out in [11], current short-term or point solutions to individual problems in the internet "have been the result of a tremendous amount of engineering intuition and heuristics, common sense, and trial and error, and have sought robustness to new uncertainties with added complexity, leading in turn to new fragilities." And the solutions are "lack of a coherent and unified theory of complex networks."

In order to systematically understand the behavior of heterogeneous traffic and further provide a possible solution based on the theoretical analysis, in this presented work, we emphasize on mathematically modeling the heterogeneous traffic behavior at a router. Based on the models, we develop techniques to estimate the proportion of the incoming non-responsive traffic. Then we validate our techniques with implementations and tests in both the NS-2 simulator and the Linux testbed.

Since our estimation technique is performed on an aggregate level, i.e. our technique estimates the incoming non-responsive traffic as one entity as long as the aggregate behavior of the traffic appears to be non-responsive at the router, it is desirable to investigate some possible approaches also based on an aggregate level to regulate non-responsive traffic and protect responsive traffic. So in this work, we evaluate the impact of heterogeneous network and network traffic on queue management schemes to probe the possibility of employing such approaches.

D. Motivation for the Presented Work

1. Modeling the Dynamics of Heterogeneous Traffic and Estimating the Proportion of Non-responsive Traffic

Considering the adoption of protocols other than TCP and the increasing proportion of UDP-based traffic in the Internet traffic, one can expect that the proportion of non-responsive traffic increases. The impact of such non-responsive traffic on the modern network has been pointed out earlier in this chapter. If the network could monitor and regulate the utilization of non-responsive traffic to a fraction of the link capacity, the impact of such attacks could be mitigated. If the link utilization by the non-responsive traffic can be estimated, some possible directions of applications can be adopted to regulate non-responsive traffic at the time of congestion:

- *Dealing within the router*

  The estimating router can raise an alarm to the network administrator for a manual interference. It can also de-route packets of non-responsive flows to alternative paths or tune traffic control schemes to drop packets from non-responsive flows to alleviate the congestion on the major path.

- *Dealing outside of the router*

  The estimating router can push back the congestion notification to upstream routers in order to trace back to attacking sources or drop packets of those sources at the nearest edge before those packets get into the network to cause congestion. By attaching information bits to packets to notify responsive sources, the router can inform responsive sources to react correspondingly to different levels of congestion.

Let us consider a possible tool which might be able to retrieve the utilization information. If all the non-responsive traffic used UDP for transport, a simple counter of UDP bytes at a link will provide an estimate of the proportion of non-responsive traffic. However, non-responsive traffic may use other protocols or use variants of TCP; some UDP applications, such as Real Audio/Video, actually respond to the congestion of network by adapting the sending rate. A simple counter of UDP bytes at a link/router, therefore, will not suffice for estimating the "apparent" proportion of non-responsive traffic. A protocol byte counter has other drawbacks: Consider a scenario where the arriving traffic consists of a large number of "small" bandwidth TCP flows, usually referred to as "web-mice", which send out a small amount of bytes intermittently[5]. In this scenario, a counter for UDP bytes will yield a count of zero while packet drops do not reduce congestion significantly at the router. Although each flow employs TCP, the aggregate behavior of "web-mice" at the router cannot be differentiated from that of non-responsive traffic. A simple protocol counter is also easily defeated by fake protocol ids in packet headers by malicious users for avoiding this detection mechanism.

It is necessary to find mechanisms to estimate the amount of arriving traffic at a router that is not responding to congestion. Such an estimation can lead to a better control of heterogeneous network traffic through appropriate adaptation of traffic control algorithms at a router. For example, an estimate of the proportion of non-responsive traffic (PONRT) at a router can aid the choice of appropriate parameters for Active Queue Management (AQM). A need for such a tuning is indicated recently in [12]. With the knowledge of the PONRT at a router, appropriate traffic management and congestion control algorithms can be employed in different operating domains of network traffic [9, 8, 13].

In the presented work, the model in [14] is extended to account for the effects of heterogeneity in traffic, by including the effects of non-responsive traffic into the model. Based on the extended model, a method is developed for estimating the PONRT at a router. The presented method employs a normalized gradient method for estimation and utilizes queue length history and packet drop history at the router, which are easily measurable at a router. The effectiveness of the proposed method is corroborated through NS-2 [15] based simulations and a Linux-based prototype.

2. Linux Implementation of the Estimation Algorithm

Following up our NS-2 implementation and simulations, a Linux implementation can give us a much practical perspective of the effectiveness of our estimation scheme in a realistic environment. It can also help us to improve the scheme for practical uses.

Linux is an open source operating system. Since kernel version 1.0 developed in 1994, it has gained more support from both individual developers and big companies. Its source code is developed under the GNU Public License and free for everyone, so it becomes a low-cost alternative operating system that everyone can adapt it to fit one's specific need.

A Linux prototype will enable us to evaluate the practical difficulties in implementing the proposed algorithm.

3. Performance Evaluation of Queue Management Schemes for Aggregate Control of Heterogeneous Traffic

With our estimation technique, the proportion of incoming non-responsive traffic can be estimated and available for other applications. So the next step that interests us is to investigate a possible approach to regulate the non-responsive traffic in order to protect the responsive traffic at the router at the time of a network attack. The attack can be defined by a threshold. If our estimation output is over the threshold, we say that the link is under attack due to the high volume of non-responsive traffic.

Since our estimation is performed on an aggregate level, the possible approach also needs to be effective on an aggregate level so that it can eliminate the overhead of examining packet headers or keeping per-flow state. The initiative of the possible approach is motivated by our early experiment results. Those results show that, by dynamically tuning RED parameters, under *long-term* traffic mix scenarios, the approach effectively regulates the bandwidth utilization of long-term non-responsive traffic and protects the responsive traffic.

To take one step further, we would like to investigate the feasibility of the possible approach under more heterogeneous traffic mix, by including both short-term flows and long-term flows, and various network configurations. The performance of both responsive traffic and queue management schemes with different parameters is studied in this work.

We study the performance of three different queue management schemes, drop-tail (DT), RED and RED with ECN enabled (RED-ECN) under different workloads, link capacities and buffer sizes. The workloads have different fractions of Long Term Non Responsive Flows (LTNRFs), Long Term Responsive Flows (LTRFs) and Short Term Flows (STFs). We consider different performance metrics, such as realized throughput for responsive flows, delay and link utilization.

The study in the presented work tries to address the resulting issues of these two trends of increasing link capacities and increasing non-responsive loads. We are motivated by the following questions: (a) what impact do higher non-responsive loads have on different queue management schemes?, (b) is one queue management scheme better at protecting responsive traffic over the others?, (c) are there differences in the performance of queue management schemes at different buffer sizes?, and (d) can we observe any discernible trends in the performance of queue management schemes with the expected trends in workloads and link capacities?

By answering previous questions, the feasibility of our possible approach to regulate non-responsive traffic and protect responsive traffic on an aggregate level can be thoroughly studied.

E. Present Status of Related Work

In this section, present status of related work is summarized. Related literature is categorized into three subjects: 1) modeling techniques; 2) estimation techniques; and 3) performance evalution of queue management schemes under various network and traffic configurations.

1. Related Work on Modeling Techniques

Mathematical models for the dynamics of TCP flows and associated control schemes have been proposed [14, 16, 17]. These studies have led to better analysis of TCP behavior and proposals for improved traffic controllers [18, 19].

The TCP/RED system is modeled with a set of differential equations in [14]. This model precisely describes the transient behavior of a TCP flow and a RED queue at the equilibrium state. A theoretical analysis of the TCP/RED system is presented and is further validated through simulations.

In [20], a fluid-based model is utilized to describe both responsive TCP flows and non-responsive flows. The non-responsive load is treated as an average load, which effectively reduces the link capacity allocated to responsive flows. Furthermore, the model of short-lived TCP flows is described as a shot noise process and long-lived

UDP flows as Markov ON-OFF process or $M/G/\infty$ model. The impact of non-responsive flows on the performance of AQM schemes is shown through linear analysis and simulation.

The basic fluid-based model of [14] is extended to a larger IP network topology in [21]. By considering only possible congested links, the computation complexity of the model can be reduced. Average queue lengths, loss probabilities, and average end-to-end delays can be calculated at a low computational cost using the reduced model. Variants of TCP and RED are also considered in the model. Authors show the scalability of the reduced model for large IP networks.

2. Related Work on Estimation Techniques

In [9], several per-flow-based approaches are proposed to detect high bandwidth flows and the limitations of those methods are discussed. A simple estimation mechanism to estimate arrival rates by using RED dropping history is proposed in [9, 22]. It utilizes RED packet drop history and maintains the fraction of packet drops from the flow with highest packet drops. To produce a reasonable estimation, it needs to accumulate the number of packet drops till the number meets the requirement calculated by preset parameters. The calculation is based on a statistical equation showing the probability of a flow receiving more than its arrival bandwidth share. The simulation results show that, with certain parameters, it gives a good estimate of the bandwidth of the high-bandwidth flow.

The estimation mechanism is extended in [23] to develop an AQM scheme – "RED-PD". RED-PD keeps tracking a certain number of high bandwidth flows and preferentially drops packets from them. In [24], authors further propose a clustered mechanism to detect aggregates of high bandwidth flows with RED dropping history instead of per flow detection. The aggregate-based congestion control (ACC) clusters a random sample of traffic based on one field in the packet header (usually the destination address). The ACC agent can also push back the aggregate information to the upstream so that the upstream routers can control the aggregate to alleviate the downstream congestion.

Authors of [25] propose a tomography-based congestion control (TCC) scheme. The network tomography is an edge-to-edge mechanism to infer per-link internal characteristics of a domain. The congestion detection includes delay and loss measurements. Both measurements use a stripe based probing mechanism by sending special unicast packets into the network. Loss probing happens after the scheme identifies higher delay paths. After identifying the congested link, the congestion information is sent back to the ingress router where flows causing congestion enter into the network. Ingress router identifies unresponsive flows by comparing ingress and egress rates of each monitored flow.

Partial state schemes are proposed to estimate and control high bandwidth flows by maintaining limited state information. The limited state information is selected by either sampling or caching shemes. SRED [26] utilizes a zombie list to record information of flows that have most recent packets enqueued. The information of an entry in the list is probabilistically replaced by a new information of the incoming flow if the list is full and the information of the incoming flow does not match the information of that entry. SACRED [27] uses sampling and caching method to keep a list with limited number of entries. Flows, information of which is marked in the list, receive higher drop rates when the dropping threshold is exceeded. LRU-RED [28] cooperates a LRU (Least Recently Used) cache with RED to maitain the states of high bandwidth and long-term flows. By identifying those flows with the LRU cache, LRU-RED is able to provide higher drop rate to penalize them in order to protect low bandwidth and short-term flows. LRU-FQ [29] is a Linux-based partial state router prototype. By using the similar LRU cache in LRU-RED, it is able to identify high bandwidth and long-term flows. The bandwidth distribution between cached flows and normal flows can be tuned with the weight of Fair Queuing (FQ) scheme. LRU-FQ provides shorter delays for short-term flows.

3. Related Work on Performance Evaluation

In [8], authors analyze TCP goodput, loss rate, and average queuing delay and deviation by changing number of LTRFs and STFs under a fixed RED/droptail configuration. In [30], authors analyze TCP average throughput, number of bytes sent, and UDP loss% by changing RED parameters. However, those changes do not clearly reveal the relation between RED configurations or buffer sizes and TCP throughput. And in both papers, a fixed 10% UDP load is used.

The impact of STFs on RED queue dynamics is illustrated in [7]. The importance to consider STFs is stressed while conducting any network modeling or simulation.

RED performance under different configurations is investigated with web traffic [12]. The response time is the performance metric. Authors compare cumulative distribution functions of response times between RED and droptail and conclude that the performance difference is fairly small between them and that tuning RED configurations gains little in performance. Authors further extend their previous work on RED to several AQM schemes in [31]. The response time of web traffic, along with other performance metrics, is compared between AQM schemes with and without ECN. Authors point out that AQM schemes with ECN enabled can improve the response time under highly loaded links and that droptail performs better under moderately loaded links.

The difficulties and stability issue in configuring RED for a better performance of the TCP/RED system are explained in [32]. A queue law and a feedback control law of RED control system are derived at equilibrium. The queue law describes the steady-state average queue size as a function of packet drop probability: $q = G(p)$. The feedback control law describes the packet drop probability as the function of the steady-state average queue size: $p = H(q)$. RED control system may become stable if there exists an equilibrium point (p_s, q_s), q_s of which is in the linear region of RED drop function (between min_{th} and max_{th}), such that $G(p_s) = H(q_s)$. The equilibrium point is the ideal operation point of RED.

F. Contributions of the Presented Work

Motivated by trends in both network and Internet traffic and the present status of related work, the major contributions of the presented work include:

- *Modeling the heterogeneous traffic behavior*

  The presented work is the first to extend the fluid-based TCP/RED model in [14] to a traffic mix model by accounting for the long-term non-responsive flow. The traffic mix model then is modified to describe the aggregate behavior of long-term responsive and non-responsive flows over a single bottleneck link based on assumptions for the purpose of simplifying the model. By extending the single-hop traffic mix model to a more general multi-hop model, the multi-hop model is able to clearly illustrate the aggregate traffic behavior under a general network topology with multiple congested links. The simulation results shows the correctness of the developed models when those assumptions are relaxed.

- *Developing and implementing the estimation technique based on models*

  Based the aggregate traffic mix models, corresponding estimation schemes are developed. It is the first attempt of utilizing the fluid-based model for practical use. In the estimation schemes, a parameter identification method is adopted from the adaptive control theory. Our schemes successfully combine techniques from different academic fields to achieve the estimation goal with a very low computation cost. Besides the NS-2 implementation, the estimtion technique is also implemented in the Linux kernel and tested in a Linux testbed. NS-2 simulation and Linux test results show the effectiveness of the estimation schemes.

- *Investigating the possible applications of the estimation technique*

  By utilizing the estimation information, the presented work investigates the possible application to control non-responsive traffic and protect responsive traffic on an aggregate level. It is the first attempt to control non-responsive traffic on an aggregate level without housekeeping any per-flow information.

G. Organization of Dissertation

The rest of the dissertation is organized as follows. In Chapter II, the development of extended traffic model is presented. A basic estimation algorithm is developed based on the extended model. Implementation details of and modifications to the basic algorithm are presented, along with the corresponding NS-2 based simulation results and analyses. Then the basic algorithm is extended to multi-hop topology. NS-2 simulations are also provided to verify the correctness of the multi-hop algorithm. Limitations and potential applications of the presented techniques are discussed.

In Chapter III, the Linux implementation of the one-hop estimation algorithm is presented and discussed in detail. The configuration and setup of the testbed is demonstrated. Estimation results obtained from Linux testbed are presented and analyzed.

In Chapter IV, the performance of RED, RED with ECN enabled and droptail routers is investigated under various network configurations. Performance metrics are collected, compared and analyzed. Configuration guideline and possible usage of REDs and droptail are presented.

Chapter V summarizes our current work and discusses some future work directions.

CHAPTER II

MODELING AND ESTIMATION OF HETEROGENEOUS TRAFFIC AT A ROUTER

In this chapter[1], a scheme for estimating the proportion of the incoming traffic, that is not responsive to congestion at a router, is presented. The idea of the proposed scheme is that if the observed queue length and packet drop probability do not match the predictions from a model of responsive (TCP) traffic, then the error must come from non-responsive traffic; it can then be used for estimating the proportion of non-responsive traffic. The proposed scheme is based on the queue length history, packet drop history, expected TCP and queue dynamics. The effectiveness of the proposed scheme over a wide range of traffic scenarios is corroborated using ns-2 based simulations. Potential applications of the proposed algorithms in traffic engineering and control are discussed.

A. A Model of the Aggregate Dynamics of Heterogeneous Traffic at a Router

The focus of this section is on the development of a dynamical model of heterogeneous traffic on a congested link in the network that is particularly well suited for estimating PONRT at a router. Traffic is assumed to consist of only two types of flows - TCP flows and Constant Bit Rate (CBR) flows. Without any loss of generality, it is assumed that TCP and CBR flows represent responsive and non-responsive flows respectively in the traffic. Following the fluid based models of TCP flows in [14], the extended model developed in this section is described in terms of three states - window size for responsive flows, sending rate for non-responsive flows and queue

[1] ©2004 IEEE. Reprinted, with permission, from "A Method for estimating the proportion of non-responsive traffic at a router" by Z. Zhao, S. Darbha, and N. Reddy, *IEEE Transactions on Networking*, August 2004.

length at the router. The following are the underlying assumptions in developing this model:

- The effect of all non-responsive flows can be modeled by a number of equivalent "average flows". Similarly, all responsive flows can be modeled by the same window adaptation behavior and observe the same round trip time (RTT). Such assumptions are crude first approximations of the real-world traffic; nevertheless, they capture the average or macroscopic dynamics of the heterogeneous traffic at the router, especially when the number of flows is large. This assumption seems reasonable for applications where one is interested in the evolution of the queue length at time scales slower than the longest possible RTT of a responsive flow. The estimate of PONRT is a representative of the true time-averaged PONRT at this time scale. Such approximations are also used in modeling ground traffic flow, see [33].

- Queue length and window size of responsive flows change slowly within a single RTT of a responsive flow. Packet drop rate at the router changes slowly.

The impact of non-responsive traffic is modeled through its effect on the queue length and hence, on RTTs and packet drop probabilities, which, in turn, impact the window size of TCP flows; this interaction is captured by the presented model.

Dynamic models of homogeneous traffic with TCP flows have been proposed and studied in [14, 16, 17]. This model extends [14] by:

- introducing an evolution equation for the representative (or aggregate) sending rate of non-responsive flows as seen by the router, and by

- accounting for non-responsive flows in the evolution of queue in the buffer.

This model is described by the following set of differential equations:

$$\dot{X}_u = 0, \tag{2.1}$$

$$\dot{W}_s(t) = \frac{1}{R(q(t))} - \frac{W_s(t)W_s\big(t - R(q(t))\big)}{2R(q(t))} \cdot p\big(t - R(q(t))\big), \tag{2.2}$$

$$\dot{q}(t) = \frac{N_s W_s(t)}{R(q(t))} + N_u X_u - C. \tag{2.3}$$

In eqn. 2.1, X_u is the sending rate of a representative non-responsive flow. In eqn. 2.2, $W_s(t)$ is the window size of a TCP flow; N_s is the number of incoming TCP flows; N_u is the number of incoming non-responsive flows; $R(t)$ is the Round Trip Delay, which is given by $\frac{q(t)}{C} + T_p$, where T_p is a fixed propagation delay; $p(t)$ is the packet drop probability. In eqn. 2.3, $q(t)$ is the queue length and C is the outgoing link capacity. Note that the impact of reverse path congestion is minimized by the implementation of TCP cumulative ACKs, and hence not considered in this model. Eqn. 2.1 describes the aggregate behavior of a CBR flow as a representative non-responsive flow. Eqn. 2.2 describes the behavior of a representative TCP flow in congestion control phase and is the same as in [14]. It indicates that the evolution of window size is related to the round trip delay, drop probability and to its history. Eqn. 2.3 represents the dynamics of queue length.

We then define $W_u(t) := X_u R(q(t))$. As shown in Figure 1, the deviation of measured RTTs from the equilibrium state of the system is sufficiently small. For the simplicity of the model, on an average, $W_u(t)$ can be approximated not to change with time, i.e.,

$$\dot{W}_u(t) = 0.$$

By replacing $X_u R(q(t))$ with W_u in eqn. 2.3, one gets:

$$\dot{q}(t) = \frac{N_s W_s(t) + N_u W_u}{R(q(t))} - C. \tag{2.4}$$

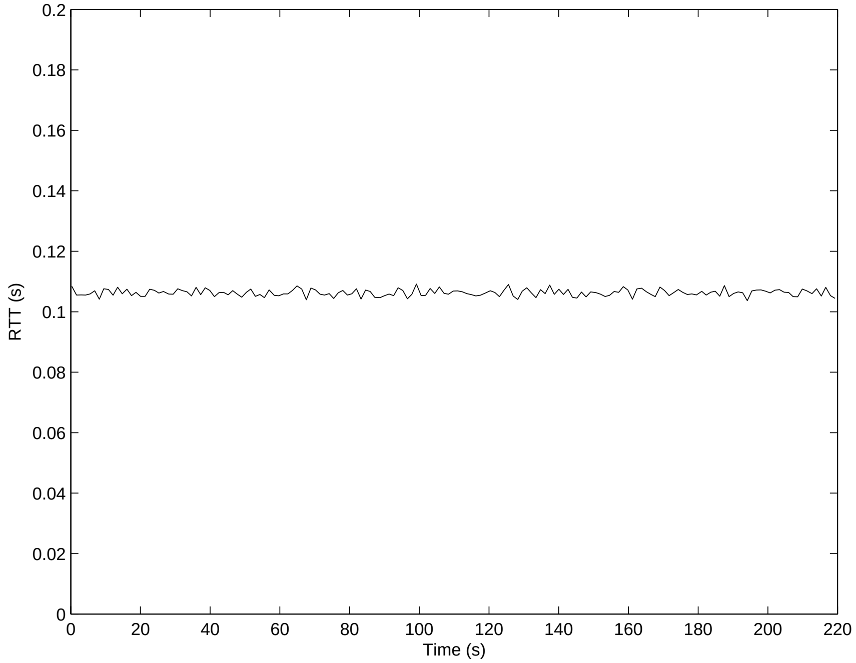


Fig. 1. Measured RTT under the Equilibrium State

Based on the model, an algorithm to determine the PONRT at a router is proposed in the next section. The focus is on estimating of PONRT on the aggregate as opposed to identifying individual flows that are non-responsive [28, 27, 9].

As with any model, the reasonableness of the proposed model depends on how well it predicts the proportion of non-responsive traffic. Simulation studies reported in the subsequent sections corroborate the suitability of the proposed model.

A model accounting for constant sending rate flows(CBR fows), instead of the approximation of using constant W_u flows, is presented in Appendix A. The main idea of the estimation algorithm based on that model remains the same as the one presented in section B. The only difference is that there are three unknown quantities $(N_s, z(t), X_uN_u)$ to be estimated and they are related by one equation. In contrast, here are two unknowns$(N_s, z(t))$ in the model presented in this chapter.

B. Basic Estimation Algorithm

In this section, an algorithm for estimating the PONRT will be developed based on the extended model presented in Section A. At first, a basic estimation algorithm will be presented in this section. This algorithm will then be modified to account for scenarios where no packets are dropped or where large number of incoming packets are dropped in Section C. The basic algorithm, which considers a single bottleneck link, will be extended to the multi-hop topology in Section D. The simulations corresponding to the algorithms developed in this section are presented in Section C.

For the purpose of developing an estimation algorithm, the dynamics of the traffic mix will be expressed in terms of the total responsive load $z(t)$ and the total non-responsive load D. The terms $z(t)$ and D are given by the following relationships:

$$z(t) := N_s W_s(t), \tag{2.5}$$

$$D := N_u W_u \tag{2.6}$$

The terms $z(t)$ and D are scaled loads and are respectively representative of the number of responsive and non-responsive packets seen by the router in a RTT. The quantity of D will be estimated in the algorithm.

In terms of $z(t)$ and D, the dynamics is given by:

$$\dot{z}(t) = \frac{N_s}{R(t)} - \frac{z(t)z(t-R(t))}{2N_sR(t)}p(t-R(t)), \tag{2.7}$$

$$\dot{q}(t) = \frac{z(t)+D}{R(t)} - C, \tag{2.8}$$

$$R(t) = \frac{q(t)}{C} + T_p, \tag{2.9}$$

where $\dot{z}(t) = N_s\dot{W}_s(t)$. The underlying assumption in the estimation algorithm is that the number of TCP flows and non-responsive flows does not change or changes very slowly.

The packet drop probability, p and the queue length, q, are sampled at each *Sampling Interval* to estimate the desired fraction of non-responsive load, ψ:

$$\psi = 1 - \frac{z(t)}{D + z(t)}. \tag{2.10}$$

The term $D + z(t)$ represents the total number of incoming packets and can be counted at the ingress link of a router. If $z(t)$ can be estimated, ψ can be calculated with eqn. 2.10.

By taking the second derivative of q using eqn. 2.8, one gets

$$\ddot{q}(t) = \frac{\dot{z}(t)}{R(t)} - \frac{z(t) + D}{R^2(t)}\dot{R}(t). \tag{2.11}$$

Combining eqn. 2.11 with eqn. 2.7 yields

$$\begin{aligned} R(t)\ddot{q}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) = & \frac{N_s}{R(t)} \\ & - \frac{z(t)z(t - R(t))}{2N_s R(t)} p(t - R(t)). \end{aligned} \tag{2.12}$$

It is possible to show that the only physically realistic equilibrium of the dynamics for a fixed packet drop rate is stable, using standard linearized analysis of nonlinear differential equations [34, 35, 36]. The stability of equilibrium indicates that a "small signal" approximation of the above differential equation describes the evolution of solutions of the nonlinear differential equation reasonably accurately when the deviation from the equilbrium is sufficiently small.

The estimation algorithm is based on the small-signal behavior of the dynamics, i.e., $q(t) \approx q_0; R(t) \approx R_0 = \frac{q_0}{C} + T_p; z(t) \approx z_0 \approx z(t - R_0)$, where variables with subscript 0 correspond to their respective equilibrium values. The problem at hand is as follows: Given that the equilibrium is known partially in terms of q_0, can we describe the equilbrium completely (i.e., determine z_0, D) from the measurements of the packet drop rate, p and the queue length, q. The determination of equilbrium provides an estimate of PONRT using eqn. 2.10.

At this point, we may question why it cannot be assumed that the equilibrium value of packet drop rate, p_0, is known. The term p_0 is assumed small and in practice, the signal-to-noise ratio (SNR) of p_0 is small. We can develop a scheme for parameter identification based on the Jacobi linearization of eqn. 2.12; however, this would involve the computation of the deviation of the packet drop rate, p, from its equilibrium value, p_0. This was the scheme we first tried, but with little success owing to the small SNR of p_0. We can then think of treating p_0 as an unknown constant; however, this leads to an overparametrization with an equilbrium constraint $2N_s^2 = z_0^2 p_0$. In order to circumvent such difficulties, we obtain the following parametrization of the dynamics based on the practical small-signal approximations stated above:

$$\begin{aligned} R_0\ddot{q}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) & \\ &= \begin{bmatrix} \frac{1}{R_0} & -\frac{p\left(t-R_0\right)}{2R_0} \end{bmatrix} \begin{bmatrix} N_s \\ \frac{z_0^2}{N_s} \end{bmatrix} \end{aligned} \tag{2.13}$$

The resulting model is still nonlinear; however, it has the advantage that the unknown parameters are linearly parametrized in terms of the output, which is the left hand side of the equation.

We can define the following from eqn. 2.13:

$$\begin{aligned} \chi(t) := R(t)\ddot{q}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) & \\ &= \mathbf{W}^T(t) \begin{bmatrix} \beta_0^* \\ \beta_1^* \end{bmatrix} = \mathbf{W}^T(t)\boldsymbol{\beta^*}, \end{aligned} \tag{2.14}$$

$$\text{where } \mathbf{W}^T = \begin{bmatrix} \frac{1}{R_0} & -\frac{p\left(t-R_0\right)}{2R_0} \end{bmatrix}, \text{ and}$$

$$\boldsymbol{\beta^*} = \begin{bmatrix} \beta_0^* \\ \beta_1^* \end{bmatrix} = \begin{bmatrix} N_s \\ \frac{z_0^2}{N_s} \end{bmatrix}.$$

The term $\chi(t)$ may be thought of as an output which is linearly parametrized in terms of the unknown vector of parameters, $\boldsymbol{\beta^*}$. The term $\boldsymbol{\beta}(t)$ represents the estimate of the unknown vector of parameters at time t and $\chi_e(t)$ represents the predicted output with the current estimate of parameters. Then,

$$\begin{aligned} \chi_e(t) &= \begin{bmatrix} \frac{1}{R_0} & -\frac{p\left(t-R_0\right)}{2R_0} \end{bmatrix} \begin{bmatrix} N_s \\ \frac{z_0^2}{N_s} \end{bmatrix} \\ &= \mathbf{W}^T(t) \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \mathbf{W}^T(t)\boldsymbol{\beta} \end{aligned} \tag{2.15}$$

To develop a parameter adaptation algorithm, we require the knowledge/measurement of the output, $\chi(t)$, and the regressor, $\boldsymbol{W}(t)$. Since the parameter identification is expected to evolve at a time scale slower than an RTT, sampling of $q(t)$ must be made at least once in a RTT. If T is the time between two successive samplings of $q(t)$ (also called a *sampling interval*), it is required to be *smaller than* one RTT. While the regressor can be computed directly from the measurements of the packet drop rate and the queue length, the determination of $\chi(t)$ requires the measurements of $\dot{q}$ and $\ddot{q}$. The signal $q(t)$ is numerically differentiated in order to obtain $\dot{q}$ and $\ddot{q}$; specifically, they are computed as: $\dot{q}(t) = (q(t) - q(t-1))/T$, and $\ddot{q}(t) = (\dot{q}(t) - \dot{q}(t-1))/T$.

The error $e(t)$ between $\chi(t)$ and $\chi_e(t)$ is used to update $\boldsymbol{\beta}$ recursively. The error $e(t)$ is given by:

$$e(t) = \chi(t) - \chi_e(t)$$

While there are several recursive algorithms available for updating the parameters, Kaczmarz's projection algorithm [37, 38, 39] is employed due to its low computational complexity and quick convergence properties. Applying *normalized* Kaczmarz's projection algorithm to update $\boldsymbol{\beta}(t)$ yields:

$$\boldsymbol{\beta}(t+1) = \boldsymbol{\beta}(t) + e(t)\frac{\gamma_2 \mathbf{W}(t)}{\gamma_1 + \mathbf{W}^T(t)\mathbf{W}(t)}, \qquad (2.16)$$
$$\text{where } \gamma_1 \geq 0 \text{ and } 0 < \gamma_2 < 2$$

γ_1 and γ_2 are user-defined tuning gains[2].

Once β_0 and β_1 are determined from eqn. 2.16, we can estimate the number of responsive flows, N_s and the scaled load of responsive flows, z_0, as:

$$N_s = \beta_0$$
$$z_0 = \sqrt{\beta_0 \beta_1}$$

From the last equation for z_0 and from eqn. 2.10, we can estimate PONRT. Note that, in eqn. 2.10, $z(t) + D$ is the total number of packets coming at time t and can be counted at the ingress interface of the router in practice. It must be emphasized that the proposed algorithm measures this fraction relative to the arrival rate at the switch, and not relative to the capacity C of the outgoing link.

Note that the algorithm only depends on drop probability and queue length. Since the required number of samples of history information of $q(t)$ and $p(t)$ is relatively small, the use of memory resource is limited and the computation complexity is $O(1)$.

[2] $\gamma_1 = 5, \gamma_2 = 0.45$ for the simulations in Section C and $\gamma_2 = 0.25$ in Section D

C. Implementation and Modifications of Basic Algorithm

The basic estimation algorithm developed in the earlier section was implemented in the RED module of the Network Simulator (ns-2). The main issues in the implementation are the choice of the *Sampling Interval T* and the numerical differentiation/filtering of the signals used in estimation.

It is a common practice in control applications [40, 41] to numerically difference discrete inputs to obtain a differentiated value. The implementation of numerical differentiation of queue length($\dot{q}(t)$ and $\ddot{q}(t)$) was provided in Section B. In some cases, one further filters the difference to attenuate the high frequency content in the numerically differentiated value; this is referred to as a "dirty" derivative of the signal. The rationale behind filtering is to attenuate the high frequency noise content in the "dirty" derivatives as well as in the signal. The corner frequencies of the filters may be chosen so as to filter frequency components faster than one RTT. In the implementation, the drop probability and queue length are filtered according to the relation: *wp* $=$ α $\times$ *wp*$_{old}$ *+ (1 -* α*)* $\times$ *ap of current sampling interval* and *wq* $=$ α $\times$ *wq*$_{old}$ *+ (1 -* α*)* $\times$ *aq of current sampling interval*, where α is the forgetting factor. The average value is calculated by averaging all inputs over one *sampling interval.* A value of $\alpha = 0.4$ is chosen for filtering the queue length and a value of $\alpha = 0.6$ is chosen for filtering the drop probability. This choice of parameters results in forgetting the history information of p and q within a few *sampling intervals*(approximately one RTT).

A bottleneck link topology shown in Figure 2 was employed for simulations. A RED drop function with (min$_{th}$, max$_{th}$, p$_{max}$) = (15, 45, 0.1) is chosen for managing the queue. The bottleneck router has 60 buffers and a link capacity of 28Mb and a propagation delay of 50ms. CBR flows with a transmission rate of 1Mbps were employed to simulate non-responsive flows, and FTP flows were used to simulate

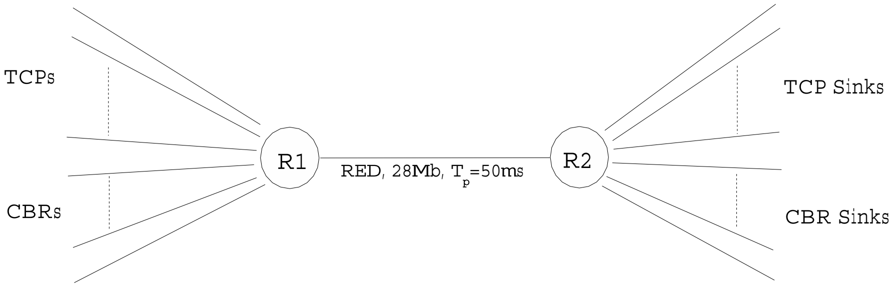


Fig. 2. Simulation Topology

long term responsive TCP flows. In the simulation, each packet has a size of 1000 bytes and the sampling interval, as discussed in Section B, is 33ms (corresponding to 1/3rd of RTT of TCP traffic).

Each simulation was run for 320s. The estimation algorithm was started after the first 100s of each simulation when the system stabilized. The estimation algorithm updated the unknown parameters every sampling interval. Several of these estimates were aggregated to produce a smoothed estimate over a larger time interval called an *estimation interval*. In our NS-2 implementation, the estimation interval was chosen to equal 20 sampling intervals and is approximately 700 ms with the choice of parameters made previously in this section.

In each estimation interval, the true value of PONRT is computed by counting the packets of non-responsive flows and dividing it by the total number of arrived packets. The estimate produced by the algorithm is then compared to their respective true values.

Mean Square Error(MSE) and Relative Error(RE) are chosen as a metric for the accuracy of estimation. Mean square error is computed as $\sum_{i=1}^{n}(estimate_i - actual_i)^2/n$ and relative error is computed as $\sum_{i=1}^{n}(estimate_i - actual_i)/\sum_{i=1}^{n} actual_i$, where n is the total number of estimations per simulation.

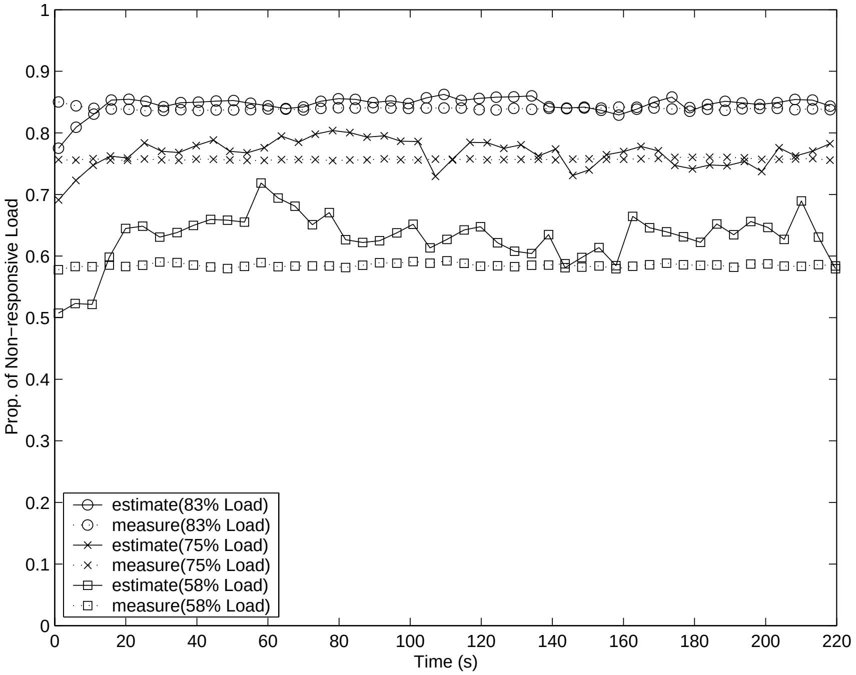


Fig. 3. High Non-responsive Load

1. High Non-responsive Load

Three simulations were set up to examine the effectiveness of the proposed algorithm under different non-responsive load conditions. There were 35 responsive flows in each simulation; three simulations correspond to 16, 22 and 25 CBR flows respectively. Each CBR flow sends packets at the rate of 1Mbps. The PONRT corresponding to 25 CBR flows is larger than PONRT corresponding to 16 or 22 flows. Shown in Figure 3 are the estimated and measured values of the PONRT obtained with the basic algorithm.

The PONRT is computed relative to the arrival rate at the switch and hence, it varies over time as the responsive traffic arrival rate changes over time.

The estimates of PONRT in Figure 3 fluctuate around their true values in a very small band. This indicates that the accuracy of estimating PONRT with the basic algorithm is high when the non-responsive load is high.

2. Persistency of Excitation and Modification of Basic Algorithm

Persistency of Excitation (PE) is an important issue in the convergence of parameters to their true values. For the parameter estimates to converge to their true values through the known regressor $\boldsymbol{W}(t)$, it is required that $\boldsymbol{W}(t)$ be persistently exciting. The rate of convergence, in general, depends on the strength of the reference signal, as can be inferred from the proofs of convergence [42].

Since the "small signal" behavior of the nonlinear dynamics of heterogeneous traffic is approximated with a static linear parametrization in terms of unknown parameters, the allowable strength of the reference signals will necessarily be limited by the region of validity of this approximation. Such an analysis is out of the scope of our research work; as such, this contribution is focused on and can only be viewed in the engineering design context.

Since there are only two unknown parameters with the parametrization chosen in the algorithm, it follows that there must necessarily be a non-zero frequency component in the regressor for the parameter estimates to converge to their true values. When the load of non-responsive arriving traffic is high, continued packet drops and fluctuations in queue length provide the necessary persistence of excitation. It is for this reason, the basic algorithm performs well under these conditions, as can be noticed from Figure 3.

In a real scenario, packet drops and variations in queue length occur persistently except when the buffer is empty. The first modification to the basic algorithm is specifically meant to address this shortcoming of the basic algorithm when there are no packet drops or queue variations.

When the buffer is empty, zero packet drop rate corresponds to the additive increase of the window size of TCP flows. For this reason, by applying $p(t - R_0) = 0$

to eqn. 2.7 and eqn. 2.12, we get:

$$\dot{z}(t) = \frac{N_s}{R(t)} = R(t)\ddot{q}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) \tag{2.17}$$

Note that $\dot{z}(t)$ is reflective of the load change and can be calculated with known parameters/measurements ($R(t)$, $q(t)$ and C) according to eqn. 2.17. If one knows $z(t_0)$, which is the last estimate of $z(t)$ prior to having no packet drops, then $z(t_x)$ ($x = 1, 2, \cdots, n-1$) can be computed recursively in the following way when $p(t - R_0) = 0$:

$$\begin{aligned} z(t_1) &= z(t_0) + \dot{z}(t_1)(t_1 - t_0), \\ z(t_2) &= z(t_1) + \dot{z}(t_2)(t_2 - t_1), \\ &\cdots \\ z(t_{n-1}) &= z(t_{n-2}) + \dot{z}(t_{n-1})(t_{n-1} - t_{n-2}). \end{aligned}$$

This algorithm is employed when no packets are dropped. As soon as packets are dropped, the basic algorithm developed in the earlier subsection is used. In Figure 4, the MSEs of basic and modified algorithms under different non-responsive loads are compared. As can be seen from this figure, the modified algorithm is more accurate than the basic algorithm in terms of MSE when non-responsive load is below 60%. Therefore, this modified estimation algorithm will be used in the rest of the simulations.

3. Effectiveness of the Estimation Algorithm with Time Varying Non-responsive Loads

To examine the effectiveness of the proposed algorithm under time varying non-responsive loads, 35 responsive flows, and 26 non-responsive flows were considered;

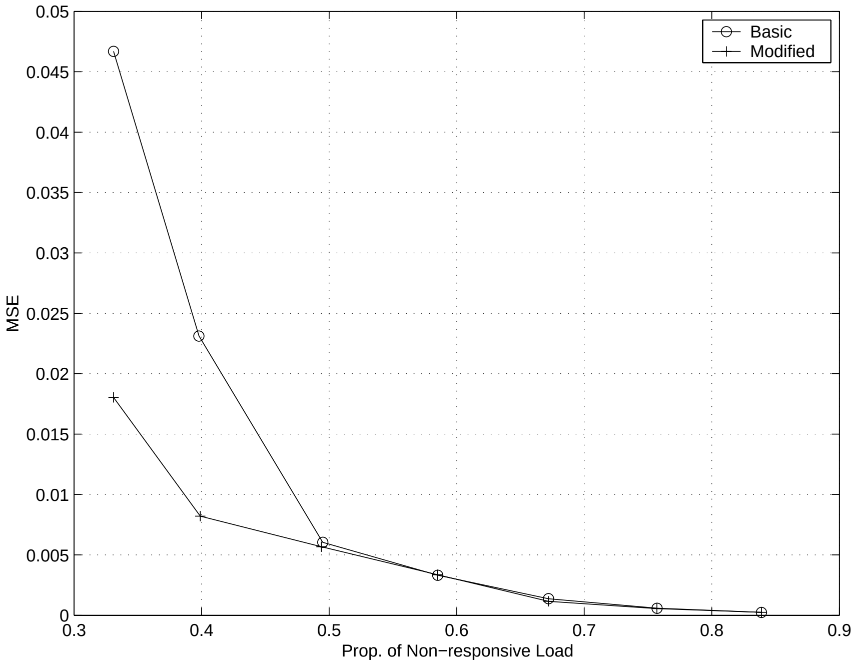


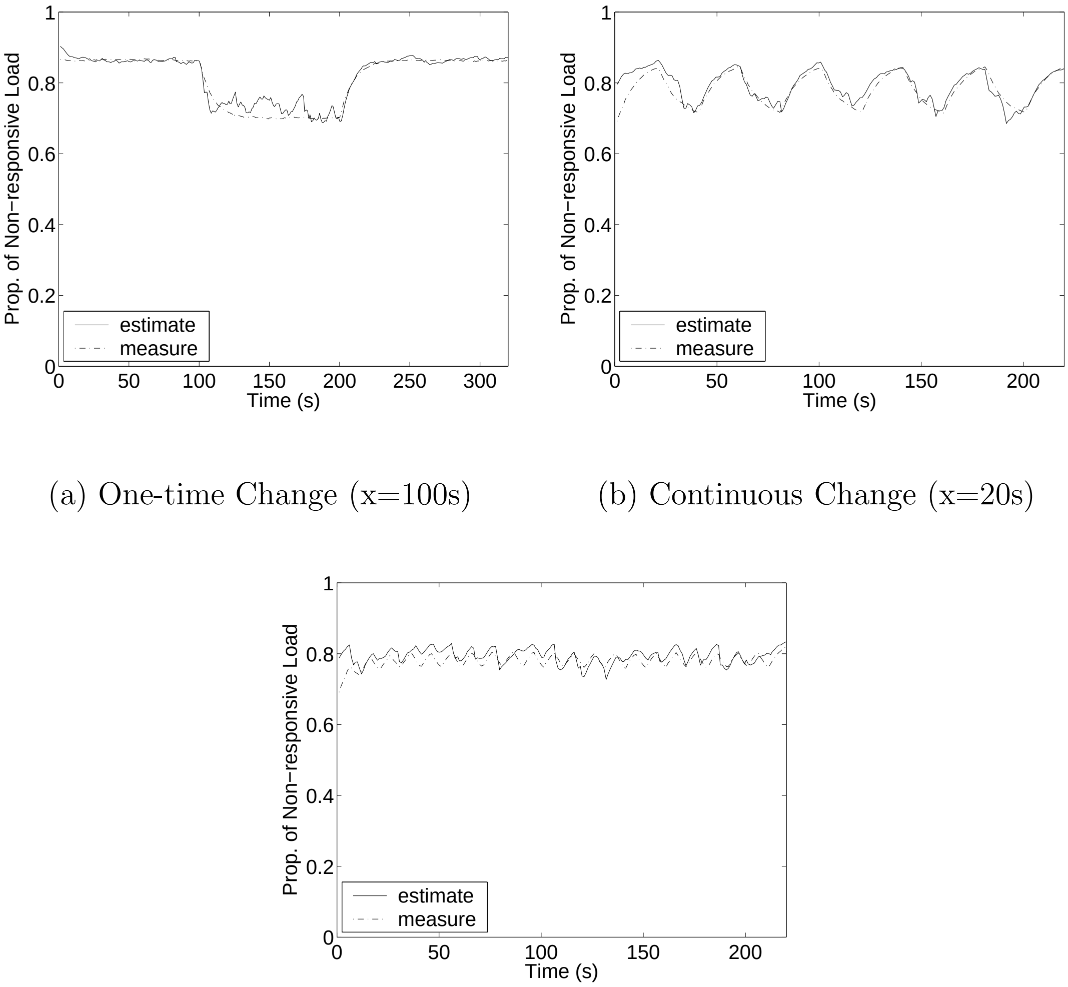
Fig. 4. Comparison of Basic and Modified Algorithm Accounting for p=0 Periods

of the non-responsive flows, 6 were ON/OFF type flows. The ON/OFF flows were ON for "x" number of seconds and OFF for the next "x" number of seconds. As a result, the non-responsive load has the shape of a square wave with a period of "2x" seconds. Different sets of simulations were performed corresponding to three different values of x: x = 100 seconds, 20 seconds, and 5 seconds. The results from these simulations are shown in Figure 5.

From Figure 5, one can observe that the algorithm can estimate the PONRT fairly well even when the non-responsive load is varying with time. It is possible to estimate faster varying non-responsive loads by choosing an estimation interval smaller than 700ms, which is used in the above simulations.

4. Mixed Traffic

To test the effectiveness of the proposed algorithm under a more realistic traffic scenario, mixed traffic consisting of short-term TCP flows, long-term TCP flows and



(a) One-time Change (x=100s)

(b) Continuous Change (x=20s)

(c) Continuous Change (x=5s)

Fig. 5. Dynamic Response to Change of Traffic

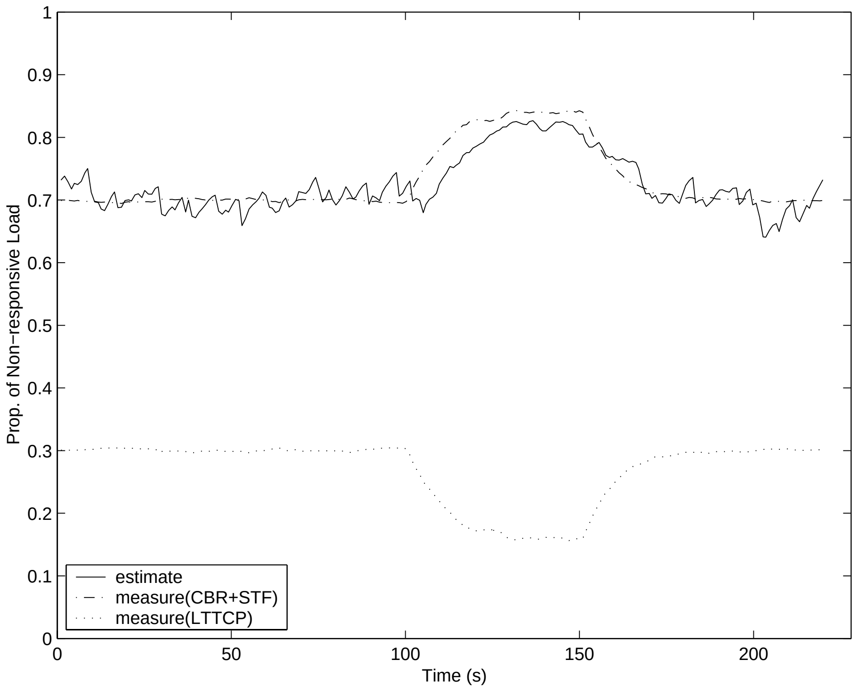


Fig. 6. Traffic Mix – Short-term TCP, Long-term TCP and Non-responsive Load

a number of non-responsive flows was simulated. In the simulations, only long-term TCP flows and non-responsive flows populate the traffic initially. At 100s, 300 short-term TCP flows were introduced into the traffic. Each short-term flow sends 20 packets randomly five times in a 50-second time period.

From Figure 6, it is clear that the estimation algorithm treats short-term TCP flows as a part of the non-responsive load. These flows do not persist in the network long enough to experience significant number of packet drops and the congestion response of a short-term TCP flow only results in an insignificant difference in the amount of traffic at the router. Moreover, the response of a single short-term TCP flow may be replaced by the arrival of another flow. As a result, these flows on an aggregate appear to be non-responsive. Similar observations about short-term flows have been made in a number of recent studies [5, 6].

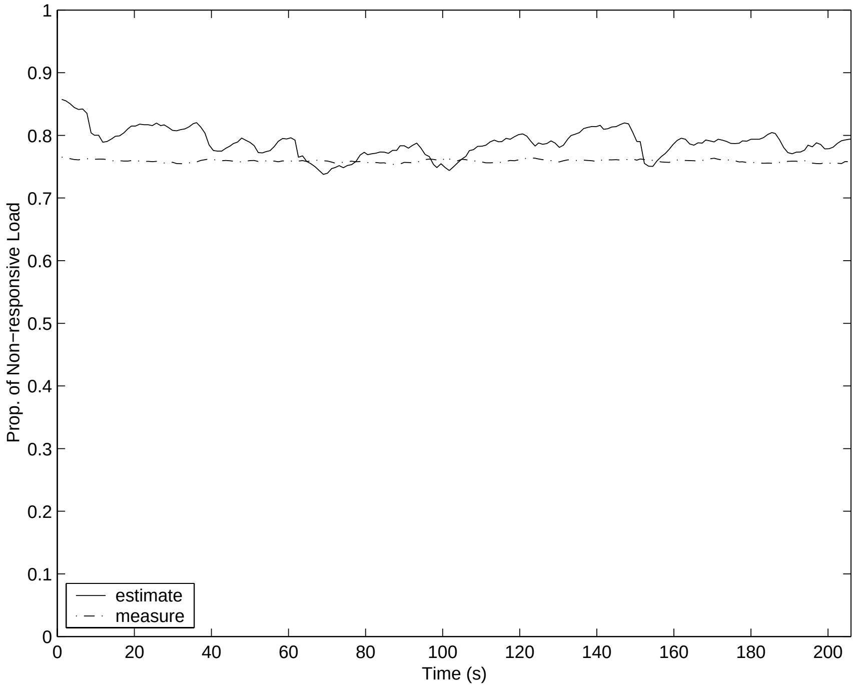
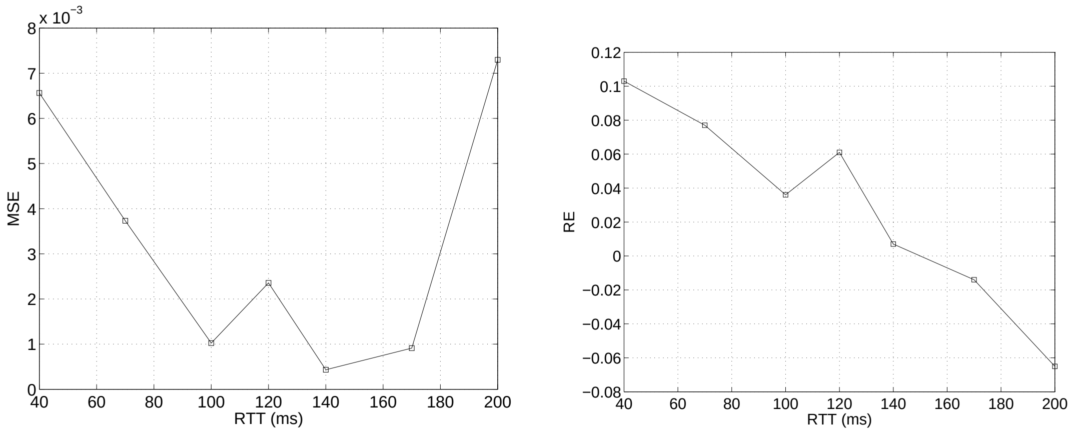


Fig. 7. TCPs with Different RTTs

5. Impact of RTTs

Another set of simulations was conducted to study the impact of RTTs. In these simulations, 35 TCP flows with different RTTs ranging from 24.4ms to 175.5ms were considered. The parameter R0 in the algorithm was set to be the average value of the range (90ms). The result of simulations is shown in Figure 7.

From Figure 7, the estimation algorithm is still effective, although the algorithm over-estimates the non-responsive traffic by a small amount. This small discrepancy can be attributed to the assumption that all TCP flows have the same RTT in the traffic model. Nevertheless, the results here show that as long as we employ a reasonable average RTT in the estimation algorithm, the different RTTs of different flows do not impact the accuracy of estimation significantly. Recent studies based on wavelets provide a convenient way to estimate the range of RTTs of flows passing through a router [43] and further help us to set a reasonable average value.



(a) MSE vs. RTT of Flows (R0=120ms)

(b) RE vs. RTT of Flows (R0=120ms)

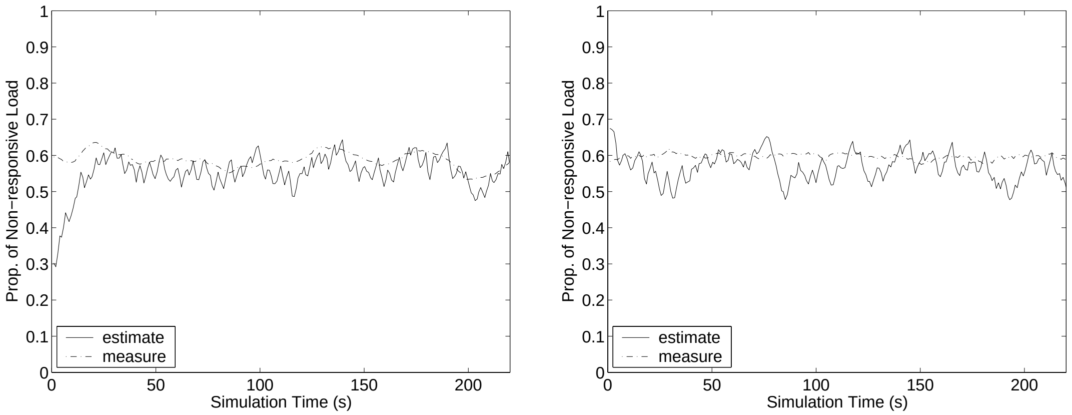
Fig. 8. Accuracy vs. RTT of Flows

In order to further study the impact of RTT on the estimation, a set of simulations, where R0 in the algorithm was fixed to be 120ms, were conducted. The RTTs of all TCP flows in one simulation were the same, but different between simulations, varying from 40ms to 200ms. MSEs and REs from the set of simulations were collected and presented in Figure 8.

From Figure 8, we can notice that when RTT of the flows is below R0, the algorithm overestimates the non-responsive traffic (since it underestimates the responsive TCP traffic). When RTT of the flows is higher than R0, the algorithm underestimates the non-responsive traffic. However, it is observed the relative errors are within 10% even over such a wide range of RTTs.

6. Impact of Variable Bit Rate(VBR) Traffic

The estimation algorithm is based on the model, in which non-responsive flows were represented by CBR flows. Since not all non-responsive flows are CBR flows, the effectiveness of the estimation algorithm in realistic scenarios will depend on its



(a) Slowly Changing VBR Flows - Period in [1 20]s (MSE=0.003534)

(b) Fast Changing VBR Flows - Period in [0.1 1]s (MSE=0.002445)

Fig. 9. Estimation under VBR and TCP Traffic

ability to estimate the PONRT in the presence of other non-responsive flows.

To address this issue, a set of simulations was set up with 16 VBR flows and 35 TCP flows. Each VBR flow is a non-responsive flow and changes its sending rate randomly selected in the range of 0.5Mbps and 1.5Mbps. A time interval between 2 different sending rates is also randomly selected within a given interval range. Each VBR flow changed its sending rate at the end of each interval till the simulation finished.

Figure 9 shows the simulation the proportion of VBR traffic with 2 different interval ranges, [1 20]s and [0.1 1]s. The larger the interval is, the less frequently the non-responsive flow changes its sending rate. So a VBR flow with intervals in [0.1 1]s changes its sending rate faster than one with intervals in [1 20]s. X axis shows the simulation period of 220s. It is noticed that the estimate of PONRT is still accurate, although the non-responsive load changes randomly. The MSEs (0.003534 and 0.002445) of this set of simulations compares well with MSE(0.003353) of the simulation with 16 CBR flows.

7. Responsive Protocols Other Than TCP

New applications, such as multimedia applications, require smooth bandwidth adaption in order to deliver quality service over internet. As a result, variants of TCP congestion control have been proposed and studied. They tend to provide much smoother sending rate than TCP does [4, 3] and still be fair to TCP over a longer time scale. In [4], the authors propose IIAD and SQRT binomial algorithms and claim that they are TCP-friendly using AQM schemes, such as RED.

Since TCP flows are one type of responsive flows, the effectiveness of the proposed estimation algorithm can be checked against other types of responsive flows. If, indeed, the other variants of TCP employing IIAD and SQRT binomial algorithms were responsive to congestion at the time scale of estimation, then the estimate of PONRT using the proposed algorithm should be accurate, provided the algorithm is effective.

To test this hypothesis, a simulation was set with up 22 CBR flows representing non-responsive traffic and 35 responsive flows represented by a mixture of TCP and IIAD/SQRT flows. The same topology and RED configurations, as in previous simulations, was used. A comparison of MSEs with different proportions of IIAD and SQRT binomial flows is shown in Table I. If there is 0% of IIAD/SQRT, it means that all the 35 responsive flows are TCP flows. If there is 100% of IIAD/SQRT, it means that no TCP flow is among the 35 responsive flows. From the simulation results, it can be observed that the difference among MSEs is very small. This result corroborates the effectiveness of the proposed scheme with other responsive flows.

Table I. Comparison of MSEs under Different Mixture

Prop. of binomial flows	**TCP est. model**	
	IIAD	**SQRT**
0%	0.000550	0.000550
50%	0.000584	0.001206
100%	0.002199	0.000451

8. Modification of Basic Algorithm for High Packet Drop Rate

The model, on which the estimation algorithm is based, assumes that the packet drop rate is small enough to affect the queue dynamics (see eqn. 2.8). However, packet drop rates can be significant when the queue lengths are close to the buffer capacity or to the maximum threshold of a RED router. In order to account for such high drop rates, the estimation algorithm is improved by considering packet drops in the queue dynamics of the model.

Specifically, the queue dynamics can be modeled as:

$$\dot{q}(t) = \frac{z(t) + D - l(t)}{R(t)} - C, \tag{2.18}$$

where $l(t)$ is the number of packet drops at time t and is known at the router.

Following the same procedure presented in section B, but replacing eqn. 2.8 with eqn. 2.18, one gets:

$$\begin{aligned} R(t)\ddot{q}(t) + \dot{l}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) &= \frac{N_s}{R(t)} \\ &- \frac{z(t)z(t-R(t))}{2N_sR(t)}p(t-R(t)) \end{aligned} \tag{2.19}$$

The left hand side of eqn. 2.19 is either known or can be easily calculated by employing a numerical differentiation scheme: $\dot{l}(t) \approx (l(t) - l(t-1))/T$. The calculation of $\dot{q}(t)$ and $\ddot{q}(t)$ was given in section B. The left hand side of this equation can be

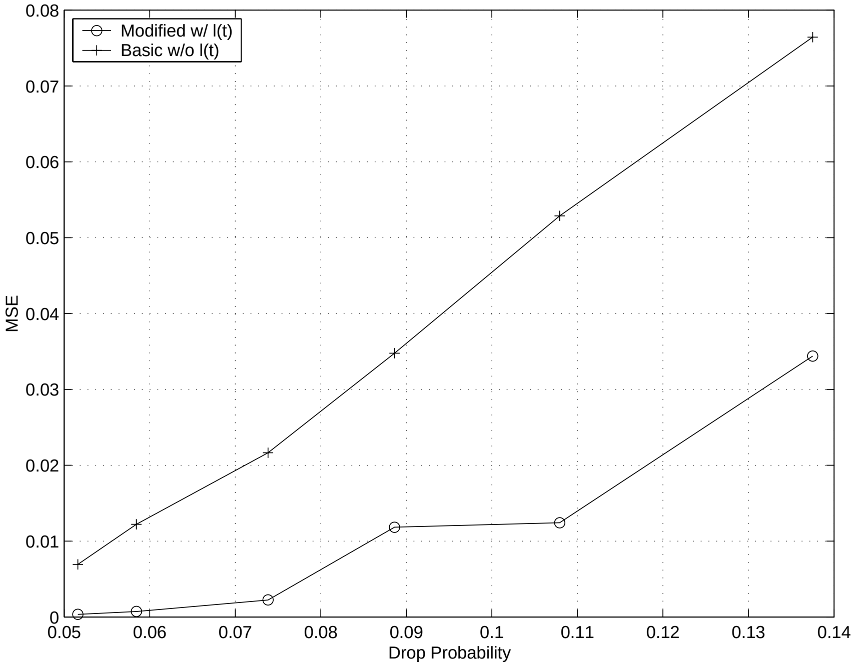


Fig. 10. Comparison of Basic and Modified Algorithm w/ l(t)

thought of as the modified $\chi_{mod}(t)$, while the right hand side is the same as that for the basic algorithm developed in an earlier subsection. Eqn. 2.19 provides a linear parametrization of the modified output, $\chi_{mod}(t)$, with respect to the set of unknown parameters, β^*; following the same procedure as in B, the normalized Kaczmarz's projection algorithm is employed to update the unknown parameter vector recursively.

To corroborate the effectiveness of the modified algorithm when the drop probability is high, a simulation with 35 TCP flows and 22 CBR flows was set up. In order to increase drop probability $p(t)$, the minimum threshold of RED was increased. In Figure 10, MSEs of algorithm with and without $l(t)$ are compared. It is noticed that, with the increase of drop probability(large number of packet drops), the algorithm *that accounts for* $l(t)$ yields more accurate estimates of PONRT.

D. Estimation Algorithm for Multi-hop Topology

The traffic model developed in Section 2 for one-hop network topology is extended to a multi-hop network topology in this section. Correspondingly, the estimation algorithm is modified. Under a multi-hop topology, the drop probability observed by each TCP flow reflects all the packet drops along the path from its source to its destination. Let P_T^i denote the total drop probability of TCP flow i along its path. In particular, P_T^i = *total number of packet drops/total number of packets sent by TCP flow i*.

The total drop probability P_T^i of TCP flow i can be further decomposed as $p_e(t) + p^i_{\sum}(t)$. The term $p_e(t)$ represents the drop probability seen by the flow at the router employing the proposed estimation algorithm and $p^i_{\sum}(t)$ is the sum of all drop probability encountered by TCP flow i along its path, excluding the drop probability $p_e(t)$. This decomposition is based on the fact that RED routers operate in the linear region of the drop function under recommended configuration so that the drop rate is small enough. Then the approximation rule of $(1 - p_a)(1 - p_b) \approx 1 - p_a - p_b$ can be applied to decompose P_T. It is noted that $p_e(t)$ is known by the algorithm, while the measurement of $p^i_{\sum}(t)$ is not available and must be determined or taken into account by the estimation algorithm.

Under a multi-hop topology, the window dynamics of TCP flow i is as follows:

$$\begin{aligned}\dot{W}_s^i(t) &= \frac{1}{R(q(t))} - \frac{W_s^i(t)W_s^i\Big(t - R(q(t))\Big)}{2R(q(t))} \\ &\quad \cdot\Big(p_e\Big(t - R(q(t))\Big) + p^i_{\sum}\Big(t - R(q(t))\Big)\Big) \qquad (2.20)\end{aligned}$$

The aggregated TCP traffic dynamics $\dot{z}(t)$ at the ingress interface of the estimation router is defined as $\dot{z}(t) = \sum_{i=1}^{N_s} \dot{W}_s^i(t)$, where N_s is the total number of TCP flows. Applying eqn. 2.20 to the definition of $\dot{z}(t)$, we have:

$$\begin{aligned}\dot{z}(t) = {} & \frac{N_s}{R(t)} - \frac{z(t)z(t-R(t))}{2N_s^2 R(t)} \\ & \cdot \Big(N_s p_e(t-R(t)) + S_P(t-R(t))\Big) \end{aligned} \tag{2.21}$$

where $S_P(t-R(t)) = \sum_{i=1}^{N_s} p^i_{\sum}(t-R(t))$.

Following the same procedure presented in section B and accounting packet drops $l(t)$ in section 8, we get the following equation using eqn. 2.21:

$$\begin{aligned} R(t)\ddot{q}(t) + & \dot{l}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) \\ & = \frac{N_s}{R(t)} - \frac{z(t)z(t-R(t))}{2N_s^2 R(t)} \\ & \cdot \Big(N_s p_e(t-R(t)) + S_P(t-R(t))\Big) \end{aligned} \tag{2.22}$$

An observation from ns-2 simulations is that $S_P(t-R(t))$ changes very slowly or is constant within each sampling interval T. So are $z(t)$ and N_s. We can then parametrize the output linearly in terms of the unknown parameters, analogous to eqn. 2.22:

$$\begin{aligned} R(t)\ddot{q}(t) & + \dot{l}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) \\ & = \begin{bmatrix} \frac{1}{R(t)} & -\frac{p(t-R(t))}{2R(t)} & -\frac{1}{2R(t)} \end{bmatrix} \cdot \begin{bmatrix} N_s \\ \frac{z(t)z(t-R(t))}{N_s} \\ \frac{z(t)z(t-R(t))S_P(t-R(t))}{N_s^2} \end{bmatrix} \\ & \approx \begin{bmatrix} \frac{1}{R_0} & -\frac{p(t-R_0)}{2R_0} & -\frac{1}{2R_0} \end{bmatrix} \cdot \begin{bmatrix} N_s \\ \frac{z_0^2}{N_s} \\ \frac{z_0^2 S_P(t-R(t))}{N_s^2} \end{bmatrix} \\ & = \mathbf{W}^T(t)\boldsymbol{\beta} \end{aligned} \tag{2.23}$$

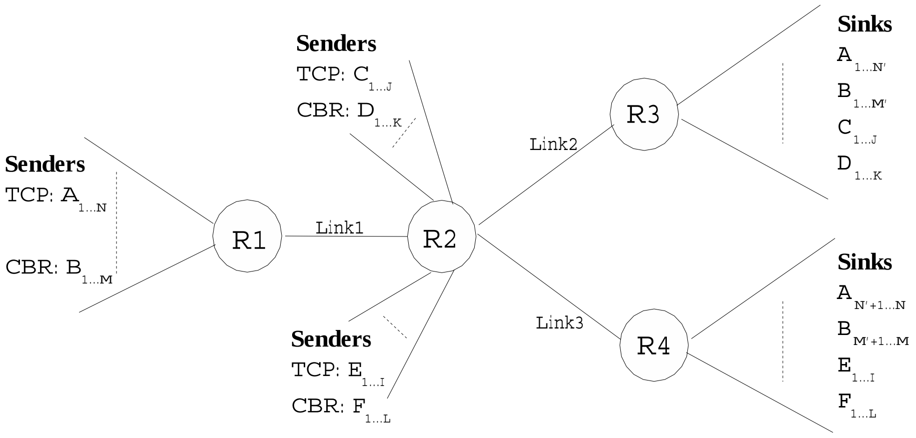


Fig. 11. Multi-hop Simulation Topology with Cross Traffic

With normalized Kaczmarz's projection algorithm, the unknown vector $\boldsymbol{\beta}$ can be estimated by utilizing the error $e(t)$ between measured and estimated values. The term $e(t)$ has the same import as in in section B.

Figure 11 shows the simulation topology with multiple hops and congestion links. The capacity of each link is 28Mb. Router R1 employs the extended estimation algorithm on Link 1. Router R2 employs RED queue management scheme on its outgoing Links 2 and 3. Let *TCP flows* be assigned to the flow set of $\boldsymbol{A, C}$ and $\boldsymbol{E}$ and *CBR flows* to the flow set of $\boldsymbol{B, D}$ and $\boldsymbol{F}$. The terms N, M, J, K, I and L represent the number of flows in their corresponding flow set. The proposed multi-hop estimation algorithm is corroborated using two sets of simulations.

First, the number of CBR flows M of flow set $\boldsymbol{B}$ was changed to be 25, 22 and 16. The number of TCP flows was set to 35 for flow set $\boldsymbol{A}(N = 35)$, 17 for flow set $\boldsymbol{C}(J = 17)$ and 18 for flow set $\boldsymbol{E}(I = 18)$. The number of CBR flows was set to 11 for flow set $\boldsymbol{D}(K = 11)$ and 6 for flow set $\boldsymbol{F}(L = 6)$. Half of the flows in $\boldsymbol{A}(N' = N/2)$ and in $\boldsymbol{B}(M' = M/2)$ went through router R3. Rest of the flows in $\boldsymbol{A}$ and $\boldsymbol{B}$ went through router R4. Drop probabilities of each link are shown in Table II. The simulation results are shown in Figure 12.

Table II. Drop Probabilities under Different Non-responsive Loads

# of CBRs (M)	**Link1**	**Link2**	**Link3**
25	3%	2.7%	1.6%
22	1.1%	2.8%	1.7%
16	0.3%	1.8%	1.4%

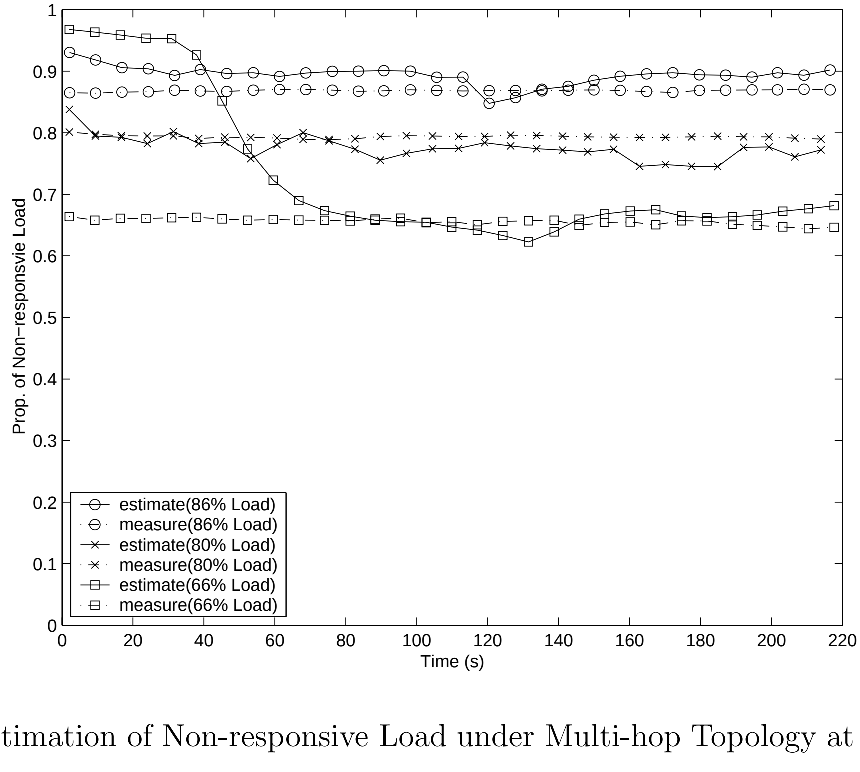


Fig. 12. Estimation of Non-responsive Load under Multi-hop Topology at Router R1

It is observed that even though the drop probability at the router under consideration is equal to or smaller than drop probabilities over other links, the extended estimation algorithm still can produce accurate estimates. In case of 16 CBRs, since most of drops happen at other links, the drop probability at the router under consideration was very low($p_e = 0.3\%$). The level of excitation is low, so the convergence is slow (Notice that estimates converge to measures after 70s.) Here, the modification to the basic algorithm for low packet drop rate is employed to ensure accuracy of estimation.

Second, instead of evenly distributing M to either branch of sinks, 6 flows were assigned to the top branch and 16 to the bottom branch. So in the case, $M = 22$ and $M' = 6$. The number of flows for other flow sets was the same as it was in the previous simulation. The simulation result is shown in Figure 13. It is observed from this simulation that the extended estimation algorithm can produce accurate estimates, independent of the distribution of CBR flows.

E. Discussion

In this chapter, an estimation algorithm to estimate the fraction of incoming traffic that is non-responsive to congestion signals (packet drops) is presented. This method relies on the evolution of queue length and packet drop rate.

When the arrival rate at the router is low, the queue lengths tend to remain low and so is the dropping probability. In this case, the level of excitation used in this method is low. As a result, the presented method tends to be more accurate in higher load situations. It is at these times of higher loads that traffic engineering decisions or potential attack detections need to be made. Hence, the proposed method seems well suited for such situations.

Besides the persistency of excitation, the estimation error may also come from

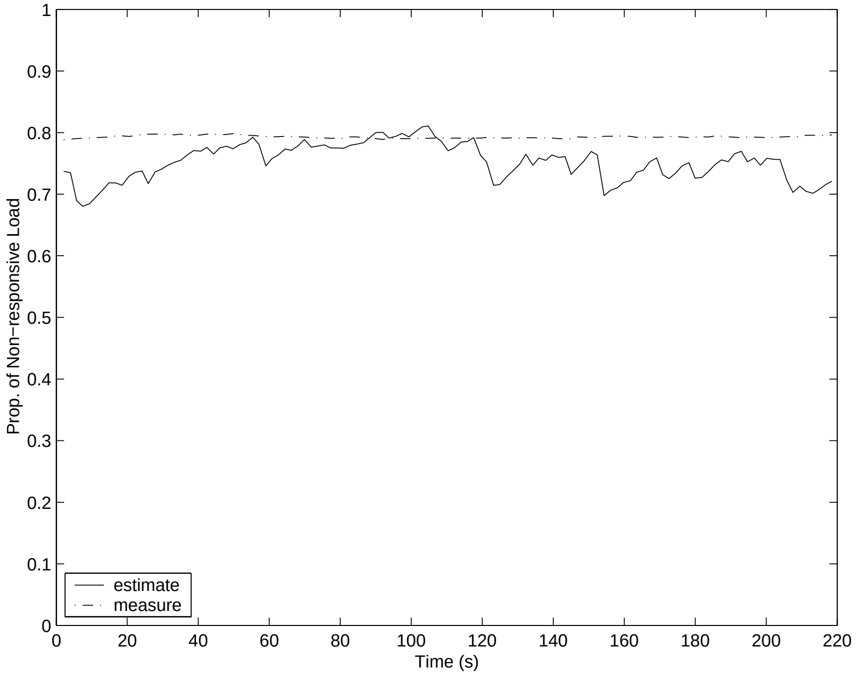


Fig. 13. Estimation under Multi-hop Topology with Unevenly Distributed Flows at Router R1

the following:

- *The accuracy of the model*

  The TCP model describes the general behavior in the congestion avoidence phase. It does not cover the fast recovery and the fast retransmission in the implementation of TCP-Reno. Neither does it completely describe the behavior of TCP-Sack in its congestion avoidence phase. Furthermore, Linux enhances the performance of TCP-Sack for various traffic scenarios. Some of those enhancements have already deviated from TCP-Sack related RFCs. So the general TCP model may lead to a certain level of error.

- *The assumptions*

  To simplify the development of the estimation scheme and make it easier to understand, some assumptions are made. Some of those assumptions are hard

> to conform in the real network. For instance, TCP window behavior is assumed to be identical so that the window behaviors of all TCP flows can be aggregated by using a simple equation. It is a strict assumption. The similarity of window behavior (similar window sizes) can be approximated when the non-responsive load is high, i.e. the window size of each TCP flow is small enough that the difference between window sizes of different TCP flows is negligible. A similar assumption is the same RTT for all TCP flows. Although, through simulations, it is shown that a reasonably accurate estimation can still be obtained by using an average RTT, the range of RTTs still needs apriori knowledge.

The focus of the presented method is on estimating the aggregate amount of non-responsive traffic at the router; this is in contrast to the earlier work suggesting the implementation of checks to see if individual flows are responding to congestion [9].

The presented work is motivated by traffic engineering concerns. It is expected that the estimation of non-responsive traffic would lead to the following applications: (a) providing a means to controlling non-responsive traffic; it is possible that an attack detection mechanism could be developed based on a robust estimation of PONRT. One may probably declare that an "attack" is in progress if the PONRT exceeds certain acceptable threshold. (b) providing a mechanism for tuning traffic control algorithms at the time of congestion. In [8], it is shown that in the presence of high non-responsive loads, drop tail buffer management may be better than RED style active queue management. The presented method could possibly be used to make such decisions at the times of congestion. To illustrate this, a simulation was set up with 22 CBR flows and 35 TCP flows competing over one 28Mb bottleneck link. In first two rows of Table III, the ability of Droptail and RED(15/45/0.1) routers to handle non-responsive flows and protect responsive flows is compared. One can

Table III. Performance of Droptail and RED

Queue Mngt.	% of TCP	% of CBR	Drop Rate
Droptail	71.5%	28.4%	34.2%
RED (15/45/0.1)	24.3%	75.6%	5.64%
RED (min_{th}=max_{th}=*buf. size*)	61.3%	38.6%	29.3%

observe that Droptail outperforms RED in protecting TCP flows from non-responsive CBR flows, since TCP still consumes approximately 70% of link bandwidth using Droptail, compared to 25% of bandwidth using RED. But, as shown in the third row of Table III, if one sets min_{th} = max_{th} = *buffer size of RED*, RED performs better to protect TCP flows than RED with (15/45/0.1) parameters. So it may be possible to tune the configurations of RED to suit different workloads. (c) providing a better control for enforcing service differentiation. Earlier work [44] on analyzing assured forwarding in differentiated services has shown that non-responsive traffic may disrupt service for responsive sources even when traffic is marked differently at the edge. With the presented estimation method, it is possible to adapt the traffic control parameters to provide better service.

Initial results based on the simulation are promising. In the next two chapters, the implementation of the estimation method in the Linux kernel will be presented and the performance of different queue management schemes for aggregrate control of traffic will be investigated as a part of possible applications of the estimation method.

CHAPTER III

LINUX IMPLEMENTATION OF ESTIMATION ALGORITHM

In this chaper, the Linux implementation of the estimation algorithm is presented. Linux is an open source operating system with traffic control functionalities. By implementing our scheme in Linux kernel, we are able to test and improve the scheme in a more realistic environment. Implementation details and encountered issues are discussed. Test results are presented.

A. Implementation in Linux Kernel

Since kernel version 2.2, Linux provides a wide variety of traffic control functions [45]. Those functions are categorized into: 1) filters; 2) classes; 3) queuing disciplines; 4) policing mechanisms. They support the architectures of both intserv and diffserv depending on dynamically loadable configurations. Since the traffic control happens after the network layer has decided to pass on a packet to other hosts, control functions are loaded at the output queue of a network interface of a Linux router. Besides forwarding or dropping a packets, richer functionalities can be easily added to the Linux traffic control, such as flow classification, rate limitation, and etc. Figure 14 shows a simple forwarding path in the Linux kernel and the location where the traffic control happens. Figure 15 shows a typical Linux traffic control structure with a root queuing discipline and classes. A simplified procedure of the Linux traffic control process is described as follows: Filters classify packets into different classes. Each class enqueues packets into its own queuing discipline. Before packets are sent to the output device, they are scheduled by the root queuing discipline.

Linux traffic control functionalities are divided into two components: kernel component and user space component(“tc” utility in the “iproute” package). The

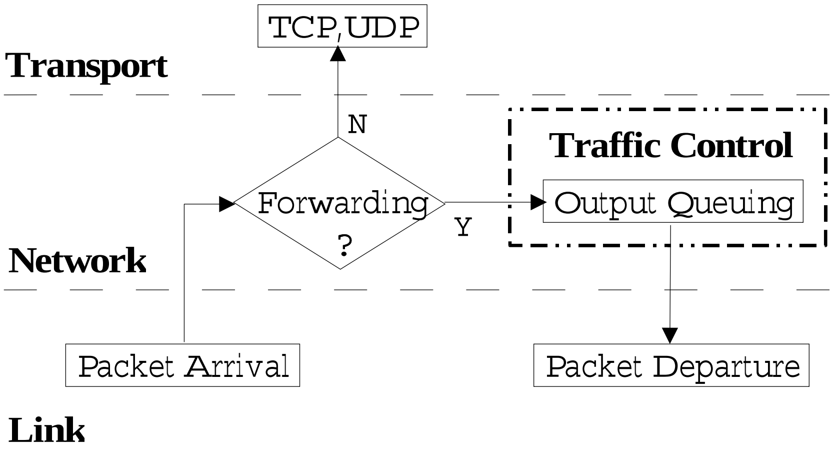


Fig. 14. Packet Forwarding Path in Linux Kernel

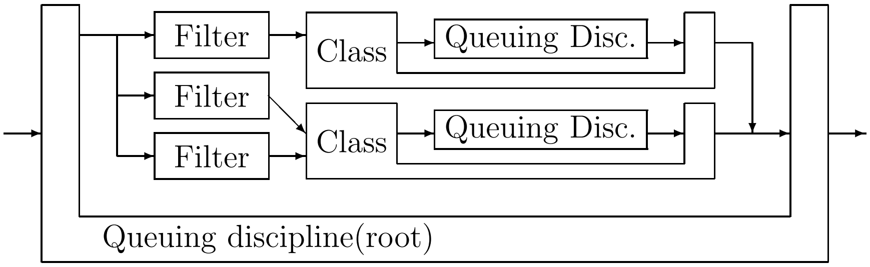


Fig. 15. Linux Traffic Control Structure

kernel component implements the traffic control functions to regulate the outbound packets, while the user space component interacts with the kernel component in order to configure or get status from it. Via the user space component, the traffic control structure, and hence the functionalities, can be dynamically modified and configured without affecting other services of the system.

Because of the merits of Linux and Linux traffic control functionalities, Linux becomes an ideal platform to implement our estimation algorithm so that our scheme can be tested and improved in a realistic environment. Conforming to the Linux traffic control architecture, our estimation algorithm is also implemented in both kernel and user space.

Linux traffic control functionalities include a variety of queuing disciplines , such as First In First Out (FIFOQ), Random Early Detection (RED), Generalized RED (GRED), Stochastic Fairness Queuing (SFQ), Class Based Queuing (CBQ), and HTB (Hierarchical Token Bucket, a new replacement of CBQ). FIFOQ is the default queuing disciplines, if no other discipline is loaded. Dynamically loading/unloading a discipline and setting/reading parameters are realized with the "tc" utility.

Our estimation algorithm, similar to the implementation in NS-2 simulator presented in Chapter II, is built upon Linux *RED* module. We call our queue management scheme "ESTMRED". In order to control the estimation module in the Linux kernel, tc utility is also modified. The detailed implementations are presented in: 1) the estimation scheme as part of Linux traffic control functions; 2) modified user space tc utility to control the estimation scheme.

1. Design of the Estimation Scheme

The estimation scheme implemented in the Linux kernel is the modified one-hop algorithm accounting for the common scenario when $p(t) = 0$ (refer to Chapter

II - Section B and C.2). The modified one-hop algorithm is the fundamental of the multi-hop algorithm. Once it is successfully implemented, corroborated and improved, the implementation of one-hop algorithm can be easily extended to the multi-hop algorithm, which can be further tested accordingly.

Our estimation scheme employs a recursive parameter adaptation algorithm based on our traffic mix model (refer to Chapter II- Section A). The recursive algorithm is implemented as follows: It collects the drop probability and the queue length and updates the estimation parameters $\beta[.]$ each time when "enqueue" function of $ESTMRED$ is called. Figure 16 shows the recursive algorithm.

Our estimation algorithm collects N_0 consecutive sets of sampling data within 1 RTT before producing one estimate output. Each sampling set includes $p(t-R)$, $\dot{q}(t)$, and $\ddot{q}(t)$. $p(t)$ and $q(t)$ are collected using Exponential Weighted Mean Average (EWMA). Weights of either EWMA scheme can be tuned independently to achieve better accuracy. After collecting N_0 samples, our scheme produces one output every $RTT/3$ (*one sampling interval*). Those outputs are processed with EWMA scheme to produce one weighed average estimation every 7 $RTTs$ (*one estimation interval*). If there are $PZTHD$ consecutive $p(t-R) = 0$ periods, our scheme will switch to calculate $z(t)$ using the modified algorithm derived in Chapter II - Section C.2. Once $p(t-R)$ is not 0, our scheme will switch back to the estimation algorithm. All parameters mentioned above, along with the tuning gains (γ_1 *and* γ_2) of the parameter identification algorithm, are required to be carefully tuned for an accurate estimate.

2. Implementation in Linux Kernel

Our new queuing discipline is added under "linux/net/sched" directory. Various modifications are distributed into other directories of the Linux source tree in order

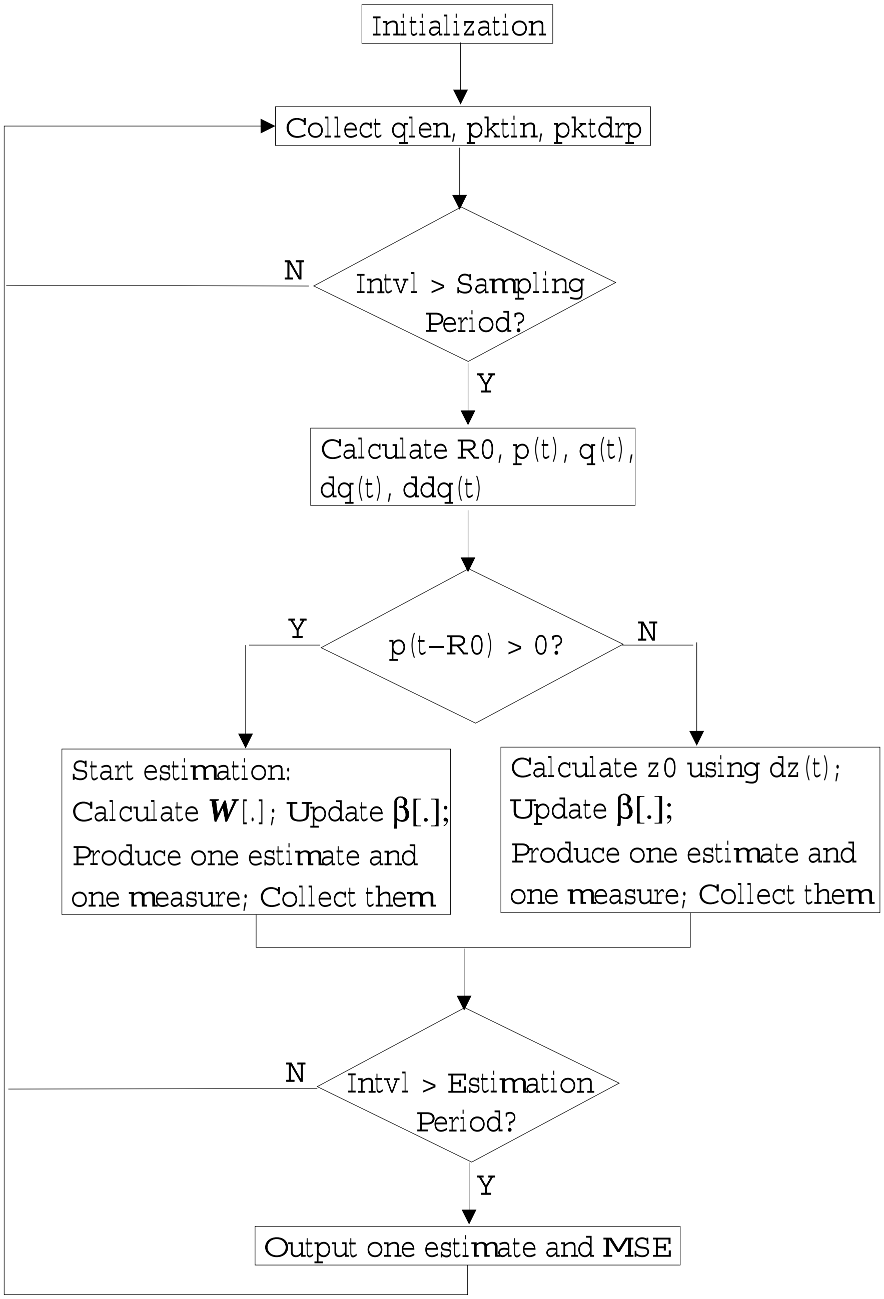

Initialization
Collect qlen, pktin, pktdrp
Intvl > Sampling Period?
N
Y
Calculate R0, p(t), q(t), dq(t), ddq(t)
p(t−R0) > 0?
Y
N
Start estimation:
Calculate W[.]; Update β[.];
Produce one estimate and one measure; Collect them
Calculate z0 using dz(t);
Update β[.];
Produce one estimate and one measure; Collect them
Intvl > Estimation Period?
N
Y
Output one estimate and MSE

Fig. 16. Estimation Algorithm Implemented in Linux Kernel

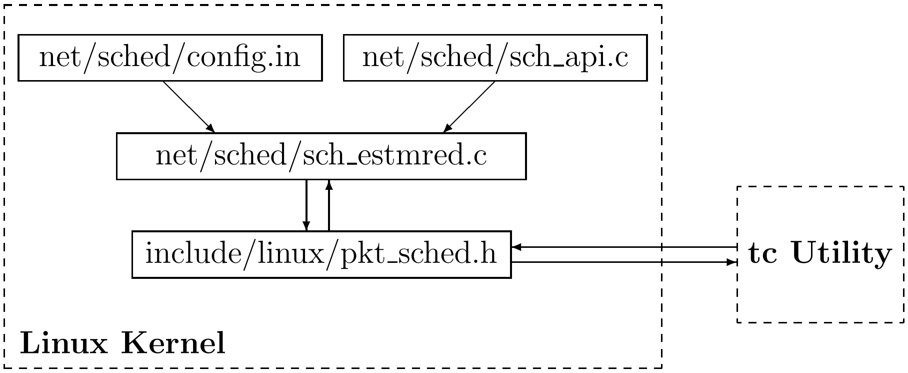


Fig. 17. Implementation Related Files in Kernel

for the kernel to recognize the new queuing discipline and interact with it. In Figure 17, it shows the relationship between some related files and the new discipline.

Here are some major issues of the implementation:

- *The unit of packets*

  Recall that in NS-2 implementation described in Chapter II - Section C, the unit of packet is used by the estimation, while RED module in the Linux kernel works under the "byte mode", i.e. the size of each packet and the queue length are counted in bytes.

  In the process of implementation, it is observed that:

  1) with different sampling intervals (because RTT changes) and different workloads, ingress bytes, dropped bytes, and queue lengths change unpredictably. Sometimes they can become very large;

  2) during the estimation process, there are many multiplication and division operations. These operations can cause the overflow/underflow of the integer storage unit unexpectedly. It is harder to contain those overflow/underflow errors when operands are some unpredictably large or small numbers. After trying different implementations, it is noticed that by converting large inputs in the unit of bytes into relatively small numbers in the unit of packets, the

entire intermediate calculation is independent of inputs and easier to debug.

Hence, the current implementation of our estimation scheme has both units coexisting. The procedure for processing data is as follows: Our scheme first collects all data, such as total ingress bytes, total bytes dropped, and queue lengths, in the unit of bytes. Then it converts queue lengths into the number of packets by dividing them by the average packet size [1]. It guarantees that there is no overflow/underflow error during intermediate calculations. Before it produces estimation results, our scheme converts the estimation data into units of bytes so that the estimation data can be combined with total ingress bytes to give the proportion of ingress non-responsive traffic in eqn. 2.10.

- *Floating point numbers or arithmetic in kernel*

  One of the limitations of programming in the Linux kernel is that there should have no floating point arithmetic in the kernel, since floating point calculations in the kernel do NOT preserve any state of the floating point unit (FPU)[2]. Since FPU is shared with user space processes, a kernel floating point operation without preserving the FPU state will cause unexpected floating point error in user space. So floating point is not recommended in the kernel programming.

  A common practice of dealing with the situation is to scale up all floating point numbers into “signed long” integers. The scaling number is selected according to the number of significant digits being reserved. It is usually a number of the power of 2 so that up or down scaling can be efficiently accomplished by bit shifting.

[1] The average packet size is imported from *tc* when the scheme is loaded.

[2] Extra code needs to be added to manually perform the context switch of FPU, if floating point arithmetic is inevitable. Because those extra code for context switching are heavy weight, the entire kernel operation is slower than it is without them.

Our estimation involves a lot of floating point numbers and calculations. And as mentioned in the previous issue, inputs are unpredictable. So are intermediate calculation outputs. It is very easy to overflow a “signed long” variable after being scaled up, if either inputs or the intermediate outputs are too large. In our implementation, a dynamic scaling scheme is employed as follows: A preset scale is defined (in our implementation, it is 14, i.e. every floating point number multiplies 2^{14} to scale up to a ”signed long” integer); Then the number is tested for overflow if it is scaled up with the preset value; If the number is small enough without overflow, the preset scale will be used; If the upscaled value of the number overflows the storage unit, it will be scaled up by using a dynamically assigned smaller scale. After desired calculation (usually division), the output of the calculation will be scaled up again to the preset scale. So all scaled outputs are always in the same magnitude, when no overflow happens. Note that the arithmetical accuracy is lower with the dynamic assigned scale than the preset scale, since the reserved significant digits is less. The second advantage to the preset scale does NOT improve the accuracy, but the potential overflow with the preset scale is effectively prevented.

- *Mathematical functions in kernel*

  There is no math function library supported within the Linux kernel and the user space C “math.h” library cannot be called from the kernel. So in case of using a math function, a dedicated integer version of the math function should be coded by utilizing only integer operations supported by the kernel.

  In our implementation, a square root function is needed to calculate the estimated z_0. One can use Taylor series to do the operation. Here, we employ a faster and resource-friendly square root algorithm with fast convergence and reasonable accuracy. Our algorithm is adapted from [46] and improved to bound the iteration times (usually lower than 10 times) and achieve better accuracy. It includes only integer operations supported by the kernel.

- *Time related issues*

  For Intel compatible platforms, using a kernel global variable *Jiffies* is the most efficient way to measure the elapsed time. *Jiffies* is increased by 1 every 10 *msec*. So it limits the finest time granularity of our implementation.

  Recall that the sampling interval is $RTT/3$, which is not necessary to be a multiple of 10 *msec*. So the finish time of a sampling period is triggered at the time point when the difference, called "rT", between it and the last sampling finish time is larger than $RTT/3$ and a multiple of 10 *msec*(In our implementation, rT is usually 40 *msec*). Then rT is used in the estimation algorithm to calculate average values and differentials, instead of $RTT/3$.

  Because *Jiffies* is counted every 10 *msec*, while our estimation algorithm uses the standard unit of seconds to calculate the first and second order of the differentiation of queue lengths, units of milli-seconds and seconds coexist in our implementation. They are converted to each other when needed.

tc utility is able to read the status and statistic data from current loaded traffic control modules. To utilize it, our estimation scheme writes the current estimation result into the shared data structure interface so that *tc* can read it at any time during the estimation period by issuing the status/statistics showing command. It also requires to modify *tc* utility to realize the function. Detailed *tc* implementation will be presented in the next subsection. Our estimation scheme also logs some debug information and all estimation results in the system log file. One can examine the running status and retrieve all estimation results during the entire estimation period by looking into the log file.

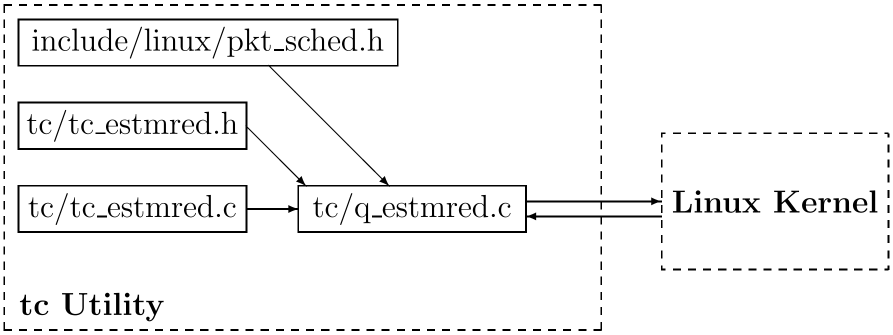


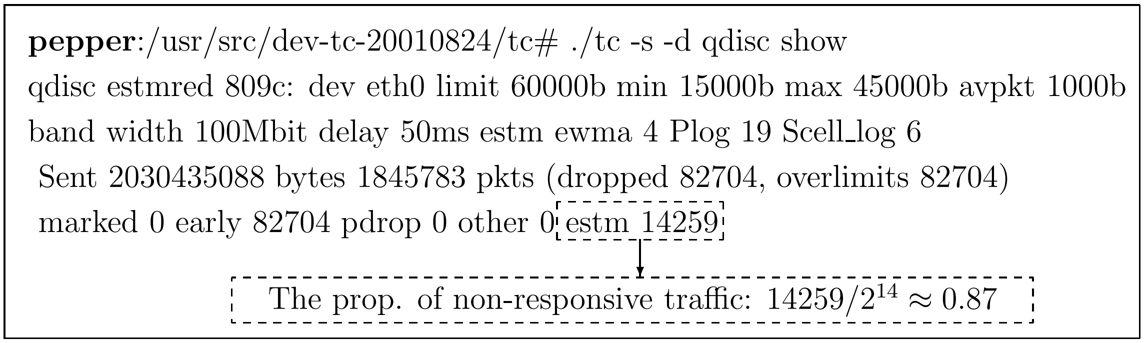
Fig. 18. Implementation Related Files in tc Utility

Our scheme is implemented such that if estimation is not required, it can be turned off at the loading time and our scheme works just like a normal RED scheme. Our scheme also automatically turns off the estimation function when there are no drops at the router.

3. Implementation in *tc* Utility

The *tc* utility is part of *iproute2* package. It is a user space utility to control traffic control functions of the Linux kernel. It can load/unload, set parameters, and show status/statistic information of kernel queuing disciplines. It utilizes a kernel supported mechanism – "rtnetlink" to interact with the kernel to realize all its functionalities. It shares data structures with the Linux kernel so that it can read/write them to exchange data with traffic control modules in the kernel. Figure 18 shows the relationship among new files related to our kernel module in tc utility. q_estmred.c is the main interface to the kernel *ESTMRED* module.

The *tc* utility includes some unit conversion function calls (in tc_util.h and tc_util.c). It is very important for us to understand the units of final outputs of those functions and to make sure that the data are converted correctly at the input of our estimation scheme. For example, at tc command line input, the "bandwidth" can be described in the unit of bits/sec, Kbits or Kbytes/sec, Mbits or Mbytes/sec,



```
pepper:/usr/src/dev-tc-20010824/tc# ./tc -s -d qdisc show
qdisc estmred 809c: dev eth0 limit 60000b min 15000b max 45000b avpkt 1000b
band width 100Mbit delay 50ms estm ewma 4 Plog 19 Scell_log 6
 Sent 2030435088 bytes 1845783 pkts (dropped 82704, overlimits 82704)
 marked 0 early 82704 pdrop 0 other 0 estm 14259
```

The prop. of non-responsive traffic: $14259/2^{14} \approx 0.87$

Fig. 19. Status/Statistical Output of the *tc* Utility

and bytes/sec. One can also omit the unit, if it is in the unit of bits/sec. The output is always converted into bytes and the conversion uses the standard 1024 bytes(=1K bytes, not 1000) for Kilo or Mega units.

Then during the running period of our scheme, the intermediate estimation result and current status/statistical data can be retrieved using “tc -s -d qdisc show” command. Figure 19 shows an example of the command output. The estimation value “estm” is scaled up using the preset scale. By dividing “estm” with 2^{14}, one gets the estimated proportion of incoming non-responsive traffic, as illustrated in the figure.

B. Experimental Testbed

In this section, the setup of Linux systems and the configuration of the experimental testbed are presented. Some important configuration issues are discussed.

Three Linux systems are set up in the testbed. Each has the kernel version of either 2.4.18 or 2.4.22 due to hardware support issues. The sender has two 100Mbit network interface cards(NICs). It sends out both TCP (long-term and short-term) and UDP based traffic to saturate the testbed network. TCP-based traffic is isolated from UDP-based traffic by being dispatched through different NICs. The router has three 100Mbit NICs, two for incoming traffic and one for outgoing traffic. The router

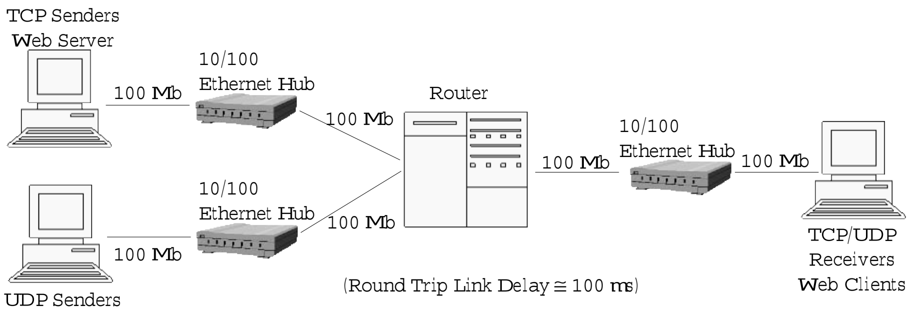


Fig. 20. Testbed Topology

Table IV. Hardware Configurations of Linux Systems

Type	CPU	RAM	Swap Space	Kernel Ver.	# of NICs
TCP Senders	AMD K6 500MHz	256MB	256MB	2.4.18	2
UDP Senders					
Web Server					
Router	Pentium 4 2.4GHz	512MB	512MB	2.4.22	3
TCP Receivers	AMD K6 500MHz	256MB	128MB	2.4.18	1
UDP Receivers					
Web Clients					

box is powerful enough to ensure no kernel drops due to routing computation and forwarding overhead. It guarantees that packets are dropped only because the output queue length is over the limit. The receiver has one 100Mbit NIC to receive both types of traffic through the router. Three pairs of NICs are connected independently with a designated 10/100Mbit auto-sensing ethernet hub to isolate traffic from one another. Figure 20 illustrates the testbed topology. Table IV lists the hardware configuration of each Linux system.

Total round trip link delay is around 100ms. It is an average round trip delay in the Internet [47]. Link delays are introduced by the Nist Net emulator [48].

Nist Net is developed at NIST Internet Technology Group. It is a general purpose emulator for Linux platform to simulate variety of dynamics in IP network. It can be easily loaded as a module when needed and comes with a user interface to make configuration easier. In our testbed, it is configured such that forward and return paths evenly split the total link delay, i.e. 50ms for the forward path and 50ms for the return path.

Each TCP/UDP sender and receiver can collect statistical information, such as total bytes and packets sent/received and data rate in bits/sec. By utilizing these data, we can verify if our testbed is set up correctly. For example, since we have one 100Mbit outbound link from the router, the sending rate of one UDP sender in the full speed should be close to 100 $Mbits/sec$. If the statistical information shown by either the UDP sender or the UDP receiver were not close to 100 $Mbits/sec$, there would be some configuration problems in the testbed. The UDP packet size is set to 972 bytes. The TCP segment size is usually 1448 bytes for our testbed network.

The generation of short-term TCP traffic(web mice) is triggered by *WebStone2.5* benchmark [49]. It is designed to evaluate the performance of web servers with or without CGI and API tests. In our testbed, WebStone is used to emulate hundreds of browsers requesting files in various sizes from the web server. The requesting frequency of each file size by web clients is controlled with a “filelist” file. Figure 21 shows the requesting frequencies of five different file sizes used in our testbed. The requesting frequency is calculated as: $the\ weight\ of\ each\ file\ size/total\ weights$. So take every 1000 page requests for example. It can be calculated using Figure 21 that, among every 1000 page requests, 500 requests are for the file with the size of 5 kbytes and 350 requests are for the file with the size of 500 bytes.

The web server used in our testbed is the well-known open source *Apache* server [50]. The *Apache* server is configured to its maximal capacity, i.e. allowing the

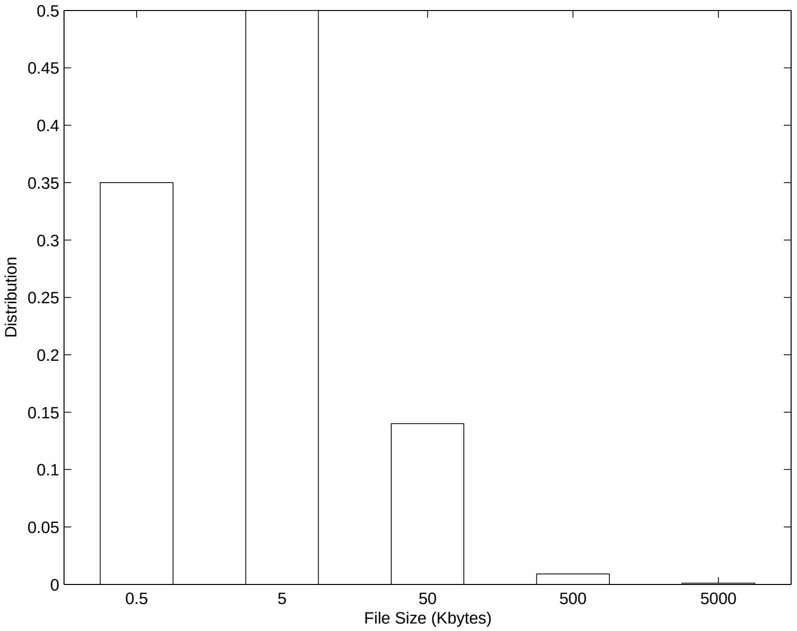


Fig. 21. Requesting Frequencies of File Sizes

maximal number of alive requests, clients, and running servers, so that a higher server throughput is possible to be reached if the number of clients/requests keeps increasing. So with the configurations of machine hardwares in our testbed, the upper bound of the web server throughput is more likely to be bounded by the computation power of the machine than by the upper limits in the *Apache* configuration.

C. Implementation Results

In this section, testing results will be presented and analyzed. Each test runs for more than 300 seconds. The estimation starts at the 60^{th} second of the entire testing span when the traffic becomes stable, since some simplifications of our algorithm are made under the condition that the system is in the equilibrium state.

Queue lengths and drop probabilities collected from RED queue management scheme are filtered by using the EWMA. The forgetting factor α for queue lengths is 0.23; The forgetting factor α for drop probabilities is 0.8. After each *estimation*

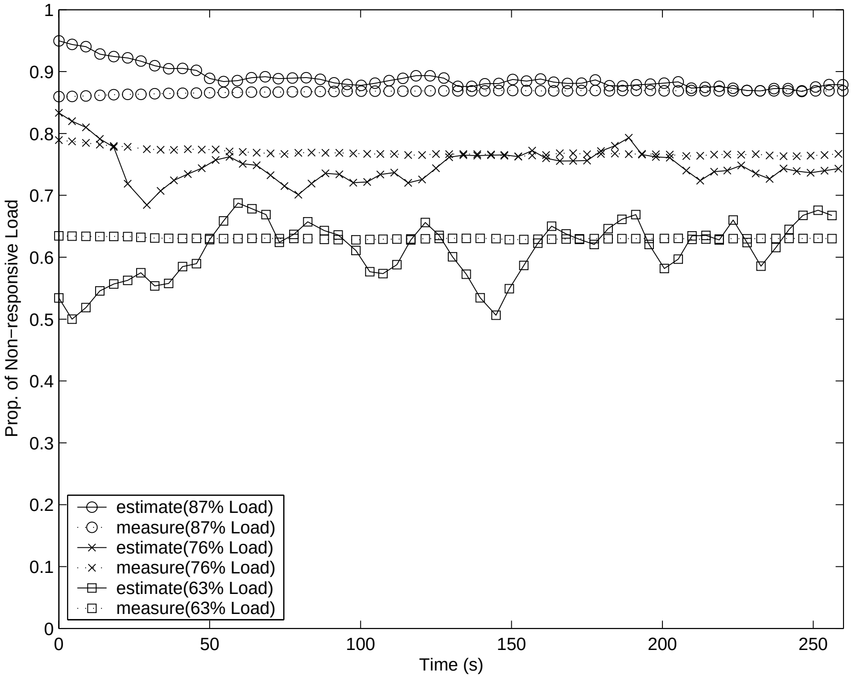


Fig. 22. High Non-responsive Loads under Linux Testbed

interval, estimation results then are compared with measurement values and the accuracy of the estimation is described using MSE defined in Chapter II - Section C. Measurement values are calculated the same way as estimation results with EWMA. The forgetting factor α for both estimate and measure is 0.975 in order to effectively damp down the fluctuation.

Since UDP-based flows and web mice are considered to be non-responsive in our tests, each measurement of non-responsive load is obtained by counting bytes from those two types of flows.

1. High Non-responsive Loads

In this experiment, non-responsive loads were 63%, 76%, and 87%. 35 TCP flows were set up to generate responsive load. Figure 22 shows the test results. Curves in dashed lines are measure values. Curves in solid lines are estimate values.

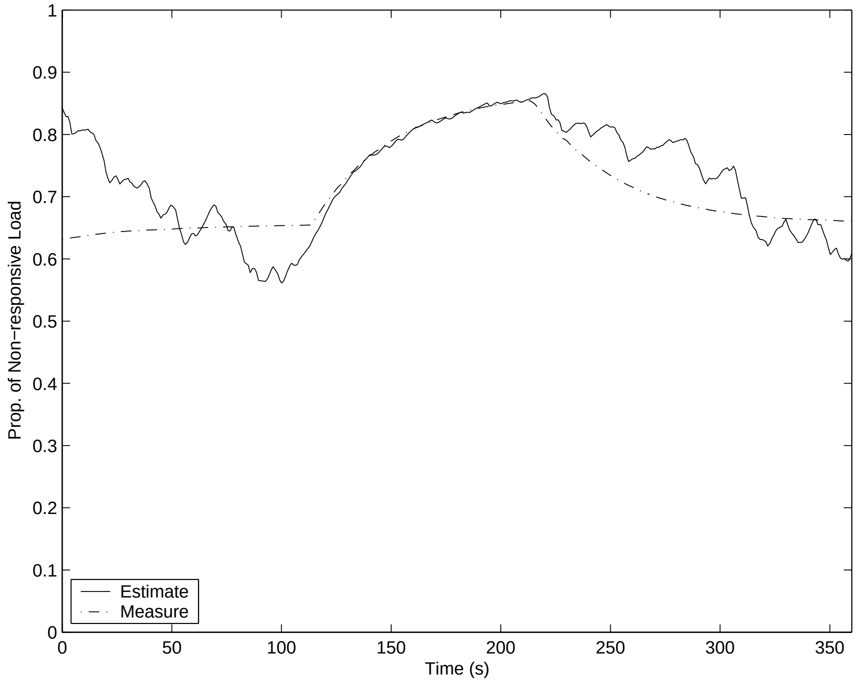


Fig. 23. Dynamic Response under Linux Testbed

It is observed that the higher the non-responsive load, the more accurate the estimation values. These results conform to those from NS-2 simulations.

2. Dynamic Response to Change of Traffic

In this experiment, initial non-responsive load were around 63%. TCP-based flows were still 35. At 150s, another 24% non-responsive load started and added up to the current non-responsive load. After 100s, the dynamic non-responsive load decreased to a stop. Figure 23 shows the results.

It is observed that our estimation follows the dynamic change of the traffic reasonably well. So our scheme is able to track changes in the non-responsive traffic and identify high volume attacks at large time scales. Since the estimation/measurement output is calculated through EWMA and the forgeting factor is 0.975, it takes about

200 EWMA calculations[3] to "forget" the history information. Recall that *the estimation interval* is 700 ms. So it takes over 100 s for the estimate/measure to converge to the new value. That is the reason why our estimation has long up and down slopes and the "flat top" (the convergence to the new proportion of non-responsive traffic) is not noticeable.

3. Heterogeneous Traffic

Since the aggregate behavior of short-term TCP traffic appears to be non-responsive at a router, this test is designed to verify if our implemenation identifies short-term TCP traffic as part of non-responsive traffic. There were 35 long-term TCP flows and the initial non-responsive load was around 67%. At 120s, 200 web clients started to request files from the web server. The crowd of web mice existed for about 120s and then stopped. Figure 24 shows the testing result.

It is noticed from Figure 24 that our estimation scheme correctly identifies the short-term TCP traffic as part of the non-responsive traffic. The same conclusion has also been drawn from the NS-2 simulation.

D. Discussion

There are two major concerns to the estimation scheme. One is the tuning process. It is a heuristic process and needs to be performed whenever the network configuration, such as RTT or the link capacity, changes. The other is the "carry-over" problem. That is if the gains (the values of tuning parameters), which yield accurate estimates in the simulator, can be carried over to the testbed and further to the real world product so that the testbed or the product only needs local fine-tunes to obtain reasonably accurate estimates. If it can be done, the estimation scheme will be

[3] $0.975^{200} < 1\%$

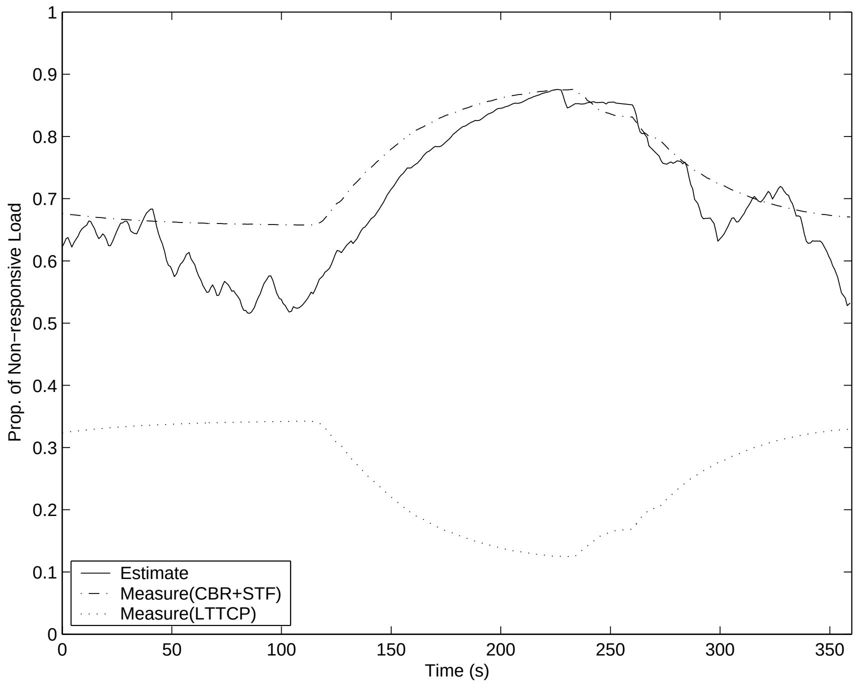


Fig. 24. Traffic Mix under Linux Testbed

reasonably easy to deploy due to the less effort to tune the scheme on the real product. But, apparently, tuning on the simulator for different configurations is inevitable.

In order to investigate the second concern, the gains in the Linux implementation are carried back to the NS-2 implementation. Then a NS-2 simulation is run to obtain MSEs, which show the accuracy of the estimate. The network and the traffic in the NS-2 simulation is configured similar to those in the Linux testbed. The NS-2 simulation is set up as follows. There are 35 TCP-Sack flows (since Linux employs TCP-Sack) and 80 CBR flows (each CBR sends $1Mbps$). The bottleneck link capacity is $100Mb$. The round trip propagation delay is around $100ms$. The simulation runs over $300s$.

The MSE obtained from the NS-2 simulation using the gains carried over from the Linux testbed is 0.001586. So the error is less than 4%. It is a reasonably small and shows the feasibility of “carry-over” as long as the network and the traffic configurations are similar.

CHAPTER IV

PERFORMANCE EVALUATION OF QUEUE MANAGEMENT SCHEMES FOR AGGREGATE CONTROL OF HETEROGENEOUS TRAFFIC

Motivated by the results in Table III, we investigate the feasibility to develop an approach to regulate non-responsive traffic and protect responsive traffic at the time of congestion. Observed from the table, a droptail router strongly biases against *long-term* non-responsive traffic on an aggregate level, i.e. the throughput of non-responsive traffic is much lower with a droptail router than with a normally configured RED router. This aggregate characteristic of a droptail router is desirable for our approach, since it can eliminate the overhead to keep state of all flows.So it is possible, by dynamically tuning RED parameters based on our estimation output, that RED behaves much more like a droptail router when the proportion of non-responsive load is over the preset threshold so that it can bias against long-term non-responsive traffic better than normally configured REDs. The feasibility of this possible approach needs to be further studied under scenarios of heterogeneous traffic mix and various network configurations.

In this chapter, we study the impact of bandwidth-delay products and short term flows on queue management schemes. Our focus is on understanding the aggregate performance of various classes of traffic under different queue management schemes. Our work is motivated by the expected trends of increasing link capacities, increasing amounts of non-responsive traffic, and a possible approach to regulate non-responsive traffic on an aggregate level. The impact of non-responsive flows and different link bandwidth-delay products on the performance of RED, RED with ECN enabled (RED-ECN) and DropTail (DT) routers is investigated. Our study considers the aggregate bandwidth of different classes of traffic and the delays observed at the router. The workloads consist of Long Term Non-Responsive Flows (LTNRFs), Long Term Responsive Flows (LTRFs), and Short Term Flows (STFs).

A. Performance Evaluation

In this section, the impact of bandwidth-delay product and non-responsive flows on the performance of DT, RED and RED-ECN is investigated under various configurations of workloads, buffer sizes and link capacities.

Different traffic workloads are considered in our evaluation. The workloads differ in the amount of the long-term non-responsive load and the amount of load from short-term flows. The total non-responsive load is fixed to be 60%. We consider workloads with both long-term only flows and a mixture of long-term and short-term flows, so we can study the impact of short-term flows on both long-term flows and buffer management schemes. We use web mice as representative short-term flows. These flows arrive at a random rate and typically only send a few packets. We use CBR flows as representative long-term non-responsive flows. We consider high loads of LTNRFs, from 30% to 60% of the total traffic load, to study their impact on the buffer management schemes.

We study the impact of link capacities by employing three different link capacities of 5Mb, 35Mb and 100Mb. This is expected to allow us to study the performance trends over a range (×20 *times* difference) of link capacities. The number of flows (STFs, LTRFs, and LTNRFs) is scaled correspondingly (based on the capacities) to result in similar workload mix from different classes of traffic (long-term responsive and non-responsive traffic, and short-term traffic). For each link capacity, we considered three different buffer sizes of 1/3, 1 and 3 times the bandwidth-delay product. These buffer sizes are chosen to study the impact of under provisioning and over provisioning as well as the normal rule of provisioning the buffer sizes.

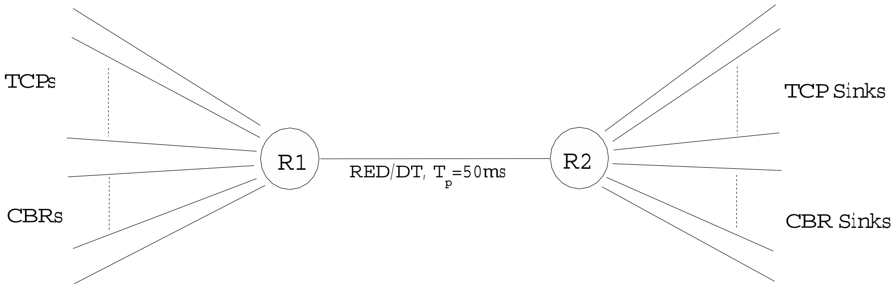


Fig. 25. Simulation Topology

Table V. 1 BWDP of Each Link Capacity

Link BW(Mb)	5	35	100
1 BWDP(pkts)	75	500	1500

The realized throughput of responsive flows (long-term TCP throughput), along with average queueing delay, link utilization and standard deviation of queueing delay, are considered as performance metrics.

1. Simulation Setup

All simulations are conducted in ns-2 simulator [15]. Simulation topology is shown in Figure 25. The link between R1 and R2 is the bottleneck link, deployed with DT, RED or RED-ECN. Link capacities employed at the bottleneck link are 5Mb, 35Mb and 100Mb. Rest of the links have capacities high enough to avoid packet drops. T_p is the one-way propagation delay of the bottleneck link. The one-way propagation delay of each ingress link of R1 and each outgoing link of R2 is $5ms$. So the total round trip propagation delay is $120ms$.

Link bandwidth-delay product (BWDP) is used as a criterion to choose the

Table VI. Buffer Sizes of Different Link Capacities and BWDPs

Multiple of BWDP	**Link Capacity (Mb)**		
	5	**35**	**100**
1/3	25	200	500
1	75	500	1500
3	225	1500	4500

buffer size of a buffer management scheme at the bottleneck link. Three buffer sizes are chosen here: 1/3, 1 time, and 3 times of 1 BWDP. Based on the link capacity and the round trip propagation delay, in Table V, 1 BWDP is calculated and rounded up in units of packets (1 packet=1000 bytes). Different buffer sizes are shown in Table VI.

RED or RED-ECN is deployed at the bottleneck link as a typical AQM scheme. It is configured by following the recommendations of [51]: $Max_{th} = 3 * Min_{th}; Max_p = 0.1; w_p = 0.002$. Its performance then is compared to that of a droptail router of the same buffer size based on different performance metrics.

Table VII. Characteristics of Different Workloads for 35Mb Link

STF Load	**35Mb Link**		
	# of LTRFs	**# of STFs**	**# of LTNRFs**[1]
0%	55	0	22
5%	55	250	22
30%	55	1300	14

[1] Each LTNRF sends at 1 Mbps

TCP flows include both long-term responsive and short-term flows. Long-term

TCP connections(FTP) represent LTRFs. Short TCP connections, sending 10 packets every 10s on average, represent typical STFs (web mice). 0%, 5%, and 30% STF loads are generated to change the proportion of traffic mix. CBR flows represent typical LTNRFs. Each CBR flow sends at 1Mbps under 35Mb and 100Mb links and 0.5Mbps under 5Mb link for easily adjusting long-term non-responsive load across different link capacities.

In current Internet traffic, total non-responsive load of LTNRFs and STFs is about 40-50%. We intend to investigate scenarios when the non-responsive load is high as explained in Section 3, so total non-responsive load, including loads of both LTNRFs and STFs, is set to 60%. 60%, 55% and 30% LTNRF loads are generated corresponding to respective STF loads. A fixed number of LTRFs are employed to complete the traffic mix. The packet size is chosen to be 1000 bytes for all flows. Table VII shows the actual number of LTRFs, LTNRFs and STFs deployed in each simulation with different STF loads under 35Mb Link. For 5Mb and 100Mb links, the number of flows are scaled down or up according to the link capacity.

2. Changing Buffer Sizes

By changing buffer sizes, realized long-term TCP throughput, link utilization and standard deviation of queueing delays are collected for analysis. 0%, 5% and 30% STF loads were generated. The performance impact of 30% STF load with different buffer sizes is presented and analyzed in the following simulations.

Figure 26 shows the correlation between average queueing delays and different multiples of BWDP(buffer sizes) under 30% STF load and 35Mb link. It is noticed that the average queueing delay increases linearly with the buffer size. Because of this linear relationship, in order to clearly illustrate the difference in performance among different buffer management schemes with more information in each plot,

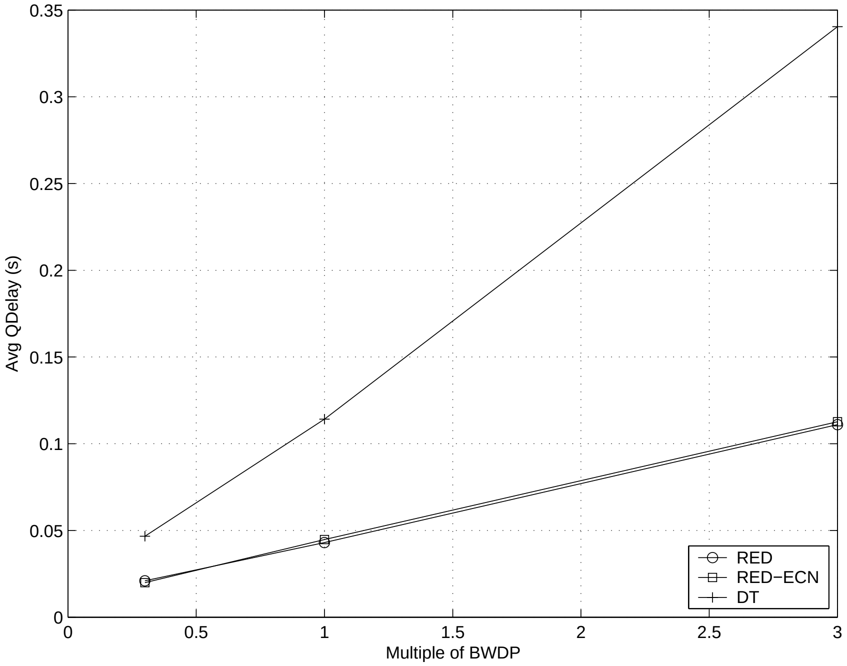


Fig. 26. Average Queueing Delays with Different Multiples of BWDP (30% STF Load and 35Mb Link)

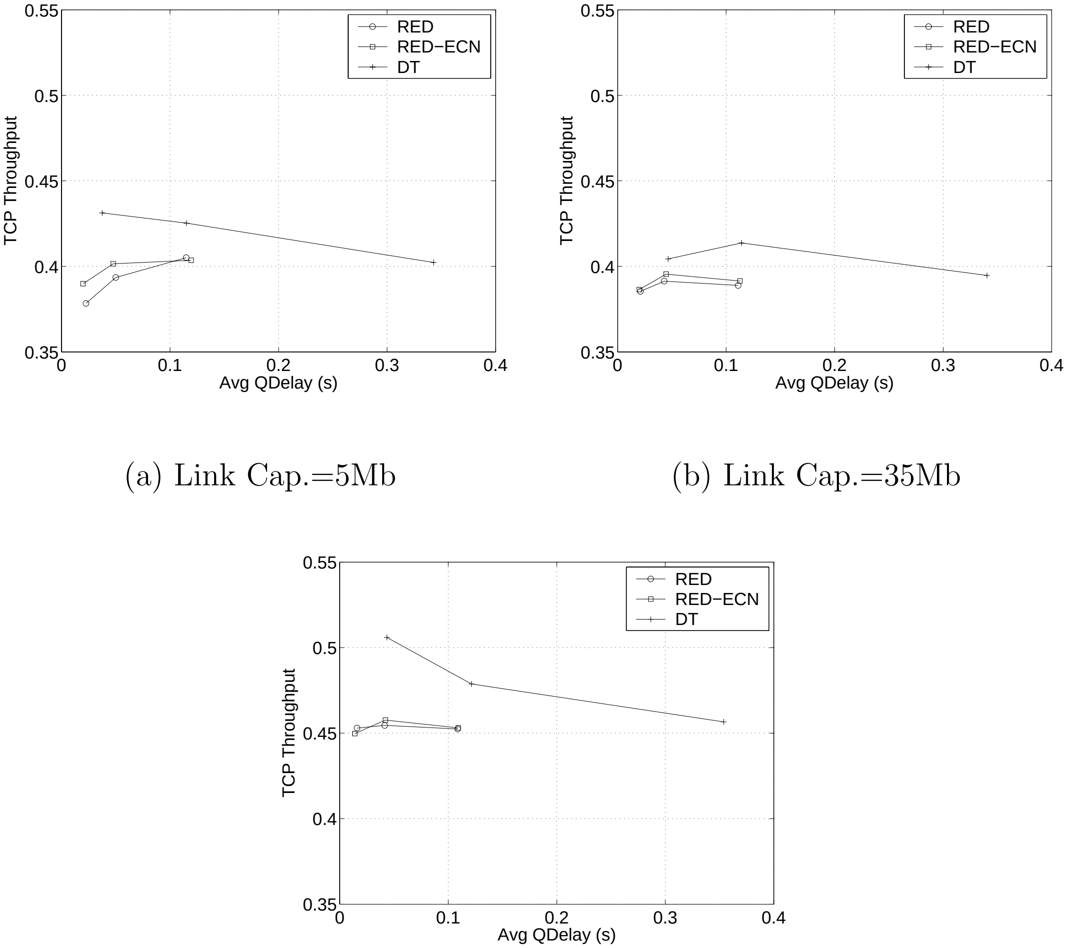
Table VIII. Link Utilization in Different Configurations under 30% STF Load

Multiple of BWDP	**5Mb Link**			**35Mb Link**			**100Mb Link**		
	RED	**RED-ECN**	**DT**	**RED**	**RED-ECN**	**DT**	**RED**	**RED-ECN**	**DT**
1/3	.943	.947	.974	.961	.955	.968	.967	.959	.971
1	.963	.965	.975	.967	.967	.971	.971	.971	.972
3	.973	.973	.976	.969	.970	.972	.972	.972	.973

average queueing delays(instead of buffer sizes) are used.

In Figure 27, realized long-term TCP throughput and average queueing delays are shown in different configurations under 30% STF load. It is noticed that TCP throughput of a droptail router is always higher than that of either RED or RED-ECN. But with the increase of BWDP, the difference in throughput is getting smaller. Average queueing delay of the droptail router under higher BWDPs is more than 3 times higher than that of RED under the same BWDP. It is undesirable to have high average queueing delays, especially for delay sensitive multimedia applications. So, under large BWDPs, RED or RED-ECN is a better choice considering the realized TCP throughput and lower average queueing delay. It is also noticed that using RED-ECN has a marginal gain of TCP throughput over RED, while RED-ECN requires ecn-compatible TCP sources.

Link utilization under 30% STF load is listed in different configurations in Table VIII. Under low BWDP cases, link utilization of the droptail router is higher than that of either RED or RED-ECN. It is intuitive, since RED tends to drop packets earlier than DT. And the impact gets magnified when the buffer is smaller. So the utilization difference between DT and RED under 5Mb link is larger than those under 35Mb and 100Mb links. With the increase of BWDP, the utilization difference between DT and RED becomes smaller. REC-ECN gains marginally on link



(a) Link Cap.=5Mb

(b) Link Cap.=35Mb

(c) Link Cap.=100Mb

Fig. 27. TCP Throughput with Different Average Queueing Delay under 30% STF Load

utilization compared to RED.

Besides the average queueing delay, standard deviation of queueing delay is also interesting, when jitter may be an important factor for the performance of applications. In Figure 28, standard deviation under 30% STF load and 30% ON/OFF LTNRF load is shown with different configurations. Each ON/OFF long-term non-responsive flow was “ON” for 20s and “OFF” for the next 20s. By configured like this, the queue length fluctuated much higher than it did under constant non-responsive load. This can help us to clearly illustrate the differences in queue management among different buffer management schemes.

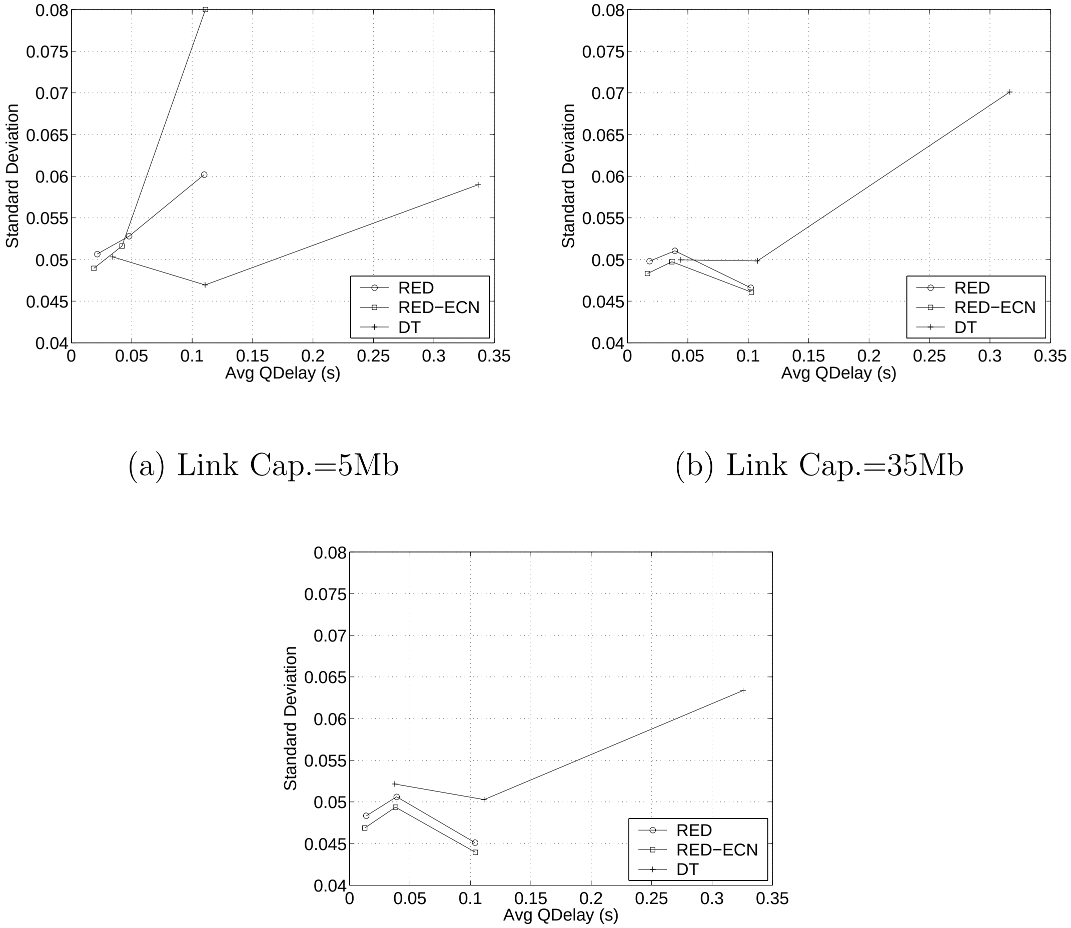
It is observed from Figure 28: 1) Under 5Mb link, DT has comparable standard deviations to RED and RED-ECN; 2) With the increase of the link capacity and buffer sizes, RED and RED-ECN have much smaller standard deviations (i.e. less queue length fluctuation) than DT. RED and RED-ECN seem more suitable for high link bandwidths and larger buffer sizes when delay jitter is an important consideration under dynamically changing workloads.

3. Changing Link Capacities

The realized long-term TCP throughputs with different link capacities have been shown in Figure 27 in Section 2. When compared across those three plots, it is also noticed that: 1) the increase of link capacities has minor impact on the differences of TCP throughputs among queue management schemes; 2) TCP throughputs are higher under 100Mb link than those under other link capacities.

In Figure 29, *relative* average queueing delays with different link capacities are compared under different configurations. *Relative* average queueing delay is defined as

$Avg\ QDelay/Round\ Trip\ Propagation\ Delay$. The round trip propagation delay is

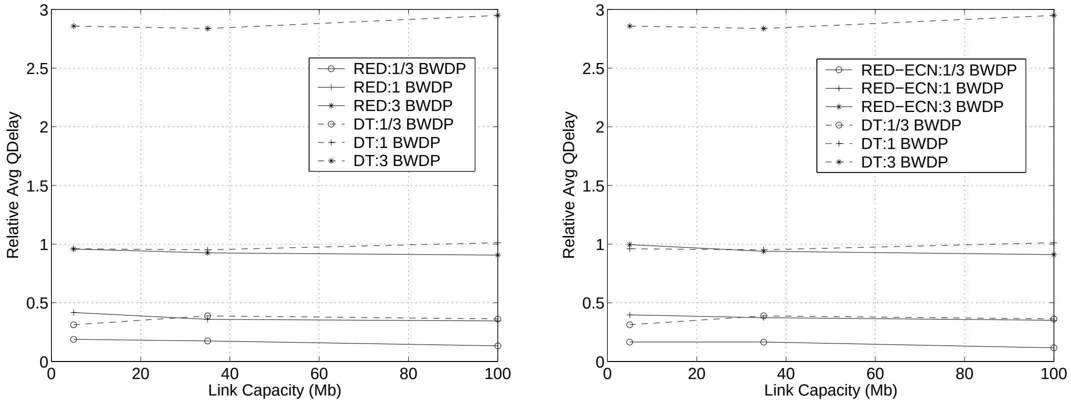


(a) Link Cap.=5Mb

(b) Link Cap.=35Mb

(c) Link Cap.=100Mb

Fig. 28. Standard Deviations of Queueing Delay with Different Average Queueing Delay under 30% STF Load and ON/OFF LTNRF Load



(a) ECN Disabled

(b) ECN Enabled

Fig. 29. Relative Average Queueing Delay with Different Link Capacities under 30% STF Load

$120ms$.

It is noticed that, for a droptail router under different BWDPs, the relative average queueing delay tends to be close to the buffer size. For example, DT under *3* BWDP has the average queueing delay around *3* times of 1 round trip propagation delay. Similar tendencies can be observed in other DT cases. The reason is that, in a droptail router, under heavy non-responsive loads, the queue tends to stay fully occupied most of the time. So queueing delay of the droptail router is close to $Buffer\ Size/Serving\ Rate$. RED, however, has noticeable lower average queueing delays. For example, under 3 BWDP, the relative average queueing delay of RED is around 1, while that of DT is around 3; under 1/3 BWDP, that of RED is around 0.1, while that of DT is around 0.3. The average queueing delay is about 3 times smaller with RED than with DT under the same multiple of BWDP. It is also noticed that RED-ECN has similar average queueing delays to RED. Changing link capacities has almost no impact on the trends of average queueing delays, by observing the almost flat curves.

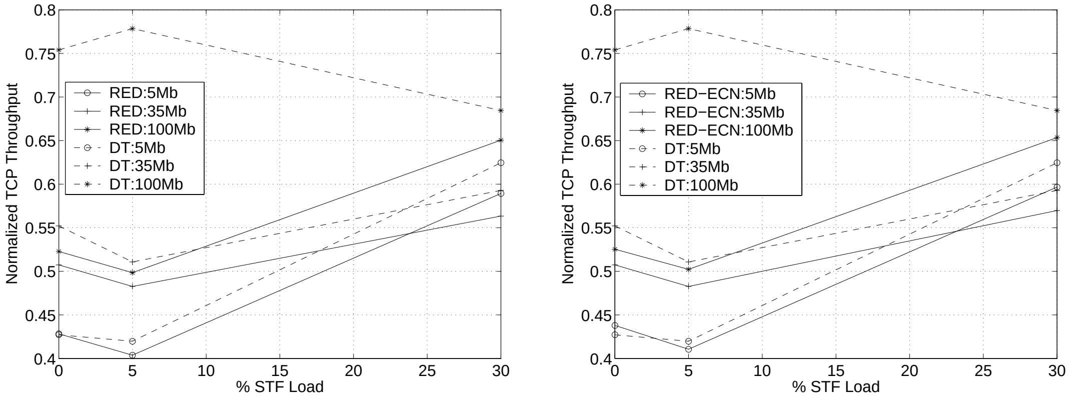
Table IX. Drop Rates or Marking Rates under 30% STF Load and 1 BWDP

QM	**Type of Flow**	**Link Capacity (Mb)**		
		5	**35**	**100**
RED	LTRF[1]	.03627	.03112	.02503
	LTNRF	.03681	.03891	.02814
RED-ECN	LTRF[2]	.00352/.04256	0/.04123	0/.03036
	LTNRF	.04688	.05352	.03406
DT	LTRF[1]	.01787	.01992	.01662
	LTNRF	.10229	.09954	.12189

[1] Format: Drop Rate

[2] Format: Drop Rate/Marking Rate

Table IX compares the drop rate or marking rate of long-term responsive and non-responsive flows from RED, RED-ECN and DT under 30% STF load and 1 BWDP with different link capacities. It is observed that drop rates of LTNRFs are much higher than those of the other flows in a droptail router, while the drop rates of RED router and the marking/drop rates of RED-ECN router show no significant difference across different types of flows. RED-ECN has higher *marking rates* of LTRFs than *drop rates* of LTRFs in RED, but with much smaller actual drop rates of LTRFs. The drop rates of LTNRFs in RED-ECN are higher than those in RED, so RED-ECN gains marginally long-term TCP throughput improvement over RED (see Figure 27).



(a) ECN Disabled

(b) ECN Enabled

Fig. 30. Normalized TCP Throughput with Different STF Loads under 1 BWDP

4. Changing STF Loads

In Figure 30, *normalized* long-term TCP throughputs with different STF loads are compared under 1 BWDP of each link capacity. The normalized TCP throughput is defined as $Total\ TCP\ Throughput/(Total\ TCP\ Throughput + Total\ UDP\ Throughput)$ so that TCP throughput is always compared to the amount of throughput from both TCP and UDP flows regardless of added STF loads. It is observed that with the increase of STF load, the throughput difference decreases or stays the same between DT and AQM schemes(RED or RED-ECN).

Because of the similar TCP throughputs between RED and RED-ECN in Figure 30, the throughputs of different classes of traffic in different configurations is listed in in Table X again. Data in the table were collected with different STF loads and the buffer size of 1 BWDP at 100Mb link. It is noticed that long-term TCP (LTRF) throughput under DT is noticeably higher than REDs when STF load is low (0% and 5%). It is also worth mentioning that short-term flows claimed the proportion

Table X. Comparison of Throughputs with Different STF Loads under 1 BWDP and 100Mb Link

STF Load	**RED**		**RED-ECN**		**DT**	
	LTRF	**LTNRF**	**LTRF**	**LTNRF**	**LTRF**	**LTNRF**
0%	.505	.461	.507	.458	.730	.238
5%	.457	.460	.460	.456	.729	.190
30%	.454	.244	.457	.242	.478	.220

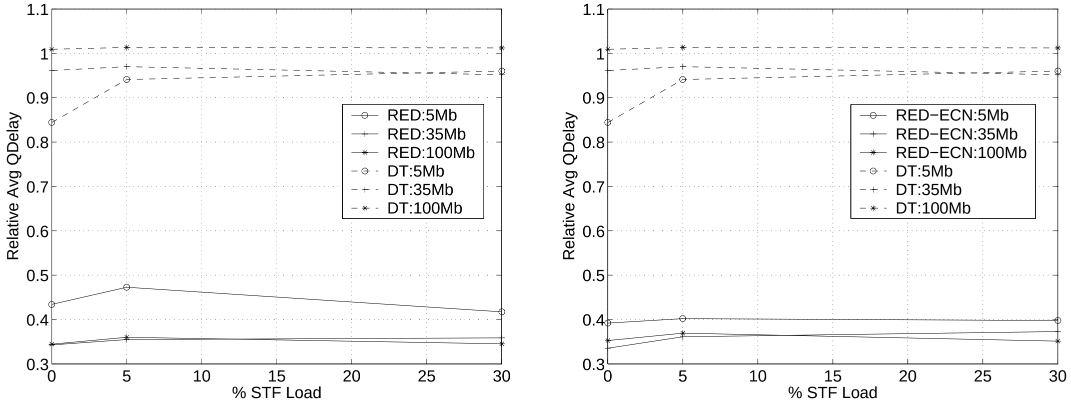
of the link capacity(5% or 30%) with almost no impact from long-term flows in this set of simulations, since the aggregated behavior of short-term flows at the router are more aggressive than long-term flows. RED-ECN has marginal improvement in TCP throughput compared to RED. Under higher STF load (30% in our experiment), TCP throughputs from RED and RED-ECN become very close to that of DT.

In Figure 31,*relative* average queueing delays under 1 BWDP are compared with different STF loads, queue management schemes and link capacities. Under 1 BWDP, *relative* average queueing delays of DT are close to 1, as explained in section 3, and about 2-3 times larger than those of RED. It is observed that changing STF loads has almost no impact on average queueing delay and that RED and RED-ECN have similar average queueing delay under each STF load.

B. Discussion

In this section, we summarize and discuss simulation results presented in Section A within the scope of our investigation.

With the existence of STFs, the performance (realized long-term TCP throughput, average queueing delay, and standard deviation of queueing delay) of both RED



(a) ECN Disabled

(b) ECN Enabled

Fig. 31. Relative Average Queueing Delay with Different STF Loads under 1 BWDP

and RED-ECN with the recommended configuration is comparable to or prevails over that of DT, especially under higher BWDP cases. Change of link capacity or STF load has minor impact on trends in long-term TCP throughputs and relative average queueing delays. With the increase of STF load, TCP throughputs of AQM schemes become very close to that of a droptail router. As a result, our possible approach to regulate non-responsive flows on the aggregate level by utilizing the estimation output is effective only under *long-term* traffic mix scenarios.

Other observations and discussions are detailed as follows:

- The throughput of STFs is uncontrollable. This observation has been made earlier. Our results confirm that the throughput of STFs is uncontrollable even under higher non-responsive loads.

- RED-ECN has marginal long-term TCP throughput improvement compared to RED, but provides lower drop rate. It was shown earlier that, with ECN mechanism enabled, TCP throughput will be benefit significantly in a highly congested network [52]. The congestion is moderate (drop rates are between 1% and 10%) in our simulations. Secondly, although our simulations employ recommended TCP-Sack(with or without ECN) by following the recommendations in [53], the performance differences between RED and RED-ECN are still very small under the workloads in our simulations. However, it is possible that RED-ECN would perform better with different workload configurations. So advantages of ECN mechanism are less significant within the scope of our investigation.

- Between DT and AQM shemes(RED or RED-ECN), for lower BWDPs or smaller buffers ($\ll 1\ BWDP$), droptail routers provide better realized long-term TCP throughput, but at slightly higher average queueing delays; for high BWDPs or larger buffers ($\geq 1\ BWDP$), such as those long-distance high-capacity links, AQM schemes have better performance in terms of acceptable realized long-term TCP throughput and significantly lower average queueing delay and jitter.

CHAPTER V

CONCLUSION AND FUTURE WORK

In this chapter, contributions of the presented work are summarized. Future research directions extending our current work are discussed.

A. Summary of the Presented Work

In this dissertation, a mathematical model describing the aggregate dynamics of heterogeneous traffic at a router was presented. This model extends the fluid based model by accounting for non-responsive traffic. The effect of non-responsive traffic was described in the queue dynamics equation and indirectly reflected by the change of the drop probability of the queue management scheme.

Then the traffic mix model was used in deriving an algorithm for estimating the fraction of the incoming traffic that is non-responsive to congestion. The purpose of the presented algorithm is to estimate the proportion of non-responsive traffic on an aggregate level regardless of protocols the traffic uses. The presented algorithm utilizes the parameter identification technique in adaptive control theory and has merits of low memory requirment and computation complexity. The basic estimation algorithm was modified to estimate accurately under different traffic scenarios. It was further extended to an estimation algorithm for the multi-hop topology. The effectiveness of the proposed algorithms, over a wide range of traffic conditions, was corroborated using ns-2 based simulations. Possible future applications of the proposed algorithms were discussed.

To further study the feasibility of our estimation algorithm in a realistic network scenario, our modified basic estimation algorithm was implemented in the Linux kernel. It helped us to validate the estimation technique and improve it in a realistic environment. The detailed implementation process and encountered issues were presented and discussed. A one-hop testbed was set up and test results on the testbed were shown and discussed.

Motivated by our estimation technique, we investigated the possibility of using an approach on an aggregate level, by utilizing the estimation output, to regulate a high volume of the non-responsive traffic and protect responsive traffic as needed and the effectiveness of this possible approach. The impact of bandwidth-delay product and non-responsive flows on the performance of both droptail and AQM schemes was investigated. Simulations under combinations of different link capacities, buffer sizes and loads of short-term flows were conducted. Observations and discussions of simulation results were provided. From the evaluation results, we observe that, with the existence of short-term flows, AQM schemes(RED or RED-ECN) have comparable long-term TCP throughput to a droptail router and have the advantage of lower queueing delays and jitters, over high-capacity links or links with high bandwidth-delay products. Droptail routers provide better long-term TCP throughput with higher delays over links with smaller bandwidth-delay products.

Since there is no obvious advantage of a droptail router over a RED/RED-ECN router when short-term flows (web mice) exists with long-term flows, our approach of dynamically tuning RED parameters to regulate non-responsive traffic is effective only under long-term traffic mix scenarios.

B. Future Work

Motivated by our previous research work, some future work directions are interesting to be further investigated:

- *Auto-tuning the estimation algorihtm*

  Currently our estimation algorithm is manually tuned by finding the minimal

MSE among various settings of the algorithm and it may take some time to find the local optimal point. With changes of network configurations(e.g. link capacity, link propagation delay, and etc.), the algorithm needs to be re-tuned to obtain an accurate estimation output. An auto-tuning technique should be incorporated into the algorithm so that the estimation scheme can be deployed easily in different configurations.

- *Building prototypes of applications based on the estimation technique*

  The estimation output of our algorithm is very useful to determine if there is abnormal behavior in the network without the need to inspect individual flows. Our previous efforts to contain a high volume of the non-responive traffic were successful only under the long-term traffic mix. An approach utilizing our estimation results to regulate non-responsive traffic needs to be investigated. Besides the regulation application, other prototypes of applications using the estimation output could be designed and developed.

- *Implementing the multi-hop algorithm in Linux*

  In our research work, we implemented a one-hop algorithm in the Linux kernel to validate the algorithm under a single-hop testbed with only one congested link. The one-hop algorithm can be easily extended to the multi-hop one in Linux by following the similar procedure in the NS-2 implementation. The Linux implementation of the multi-hop algorithm could be tested in a multi-hop testbed.

REFERENCES

[1] R.H. Zakon, "Hobbes' Internet timeline," Jan. 2004, Available via http://www.zakon.org/robert/internet/timeline/.

[2] National Laboratory for Applied Network Research, "Network traffic packet header traces," 2000, Available via http://moat.nlar.net/Traces/.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. of ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 43–56.

[4] D. Bansal and H. Balakrishnan, "TCP-friendly congestion control for real-time streaming applications," in *Proc. of Infocom*, Anchorage, Alaska, Apr. 2001, pp. 631–640.

[5] End-to-end discussion group, "Web mice versus elephants," 2000-2001, Mailing list discussion, available via http://www.postel.org/mailman/listinfo/end2end-interest.

[6] N. Caldwell, S. Savage, and T. Anderson, "Modeling tcp latency," in *Proc. of IEEE Infocom*, Tel Aviv, Israel, Mar. 2000, pp. 1742–1751.

[7] S. Floyd and E. Kohler, "Internet research needs better models," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, Jan. 2003.

[8] C. Diot, G. Iannaccone, and M. May, "Aggregate traffic performance with active queue management and drop from tail," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 3, pp. 4–13, Jul. 2001.

[9] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *ACM/IEEE Trans. on Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.

[10] C. Barakat, E. Altman, and W. Dabbous, “On TCP performance in a heterogeneous network: A survey,” *IEEE Communication Magazine*, vol. 38, no. 1, pp. 40–46, Jan. 2000.

[11] W. Willinger and J. Doyle, “Robustness and the Internet: Design and evolution,” Mar. 2002, Available via http://netlab.caltech.edu/internet/.

[12] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, “Tuning RED for web traffic,” in *Proc. of ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 139–150.

[13] Packeteer Inc., “Packetshaper: Opening a world of intelligent bandwidth management services,” 1998, Technology white paper, http://www.packeteer.com/.

[14] V. Misra, W. B. Gong, and D. Towsley, “A fluid based analysis of a network of AQM routers supporting TCP flows with an application to RED,” in *Proc. of ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 151–160.

[15] University of California at Berkeley, “Network simulator (ns),” 1997, http://www-nrg.ee.lbl.gov/ns/.

[16] C. V. Hollot, V. Mishra, D. Towsley, and W. Gong, “A control theoretic analysis of RED,” in *Proc. of Infocom*, Anchorage, Alaska, Apr. 2001, pp. 1510–1519.

[17] E. Altman, K. Avrachenkov, and C. Barakat, “Stochastic Model of TCP/IP with Stationary Random Losses,” in *Proc. of ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 231–242.

[18] S. Kuniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," in *Proc. of ACM SIGCOMM*, San Diego, California, Aug. 2001, pp. 123–134.

[19] C. V. Hollot, V. Mishra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. of Infocom*, Anchorage, Alaska, Apr. 2001, pp. 1726–1734.

[20] C.V. Hollot, Y. Liu, V. Misra, and D. Towsley, "Unresponsive flows and aqm performance," in *Proc. of IEEE Infocom*, San Francisco, California, Apr. 2003, pp. 85–95.

[21] Y. Liu, F.L. Presti, V. Misra, D. Towsley, and Y. Gu, "Fluid models and solutions for large-scale ip networks," in *Proc. of ACM Sigmetrics*, San Diego, California, Jun. 2003, pp. 91–101.

[22] S. Floyd, K. Fall, and K. Tieu, "Estimating arrival rates from the red packet drop history," Apr. 1998, Available via http://www.icir.org/floyd/papers/red-dropping.ps.

[23] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," in *Proc. of 9th International Conference on Network Protocols (ICNP)*, Riverside, California, Nov. 2001, pp. 192–201.

[24] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Aggregate-based congestion control," Aug. 2001, Poster at SIGCOMM 2001 (Extended version available via http://www.icir.org/pushback/).

[25] A. Habib and B. Bhargava, "Network tomography-based unresponsive flow control," in *Proc. of 9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, San Juan, Puerto Rico, May 2003, pp. 258–264.

[26] T.J. Ott, T.V. Lakshman, and L.H. Wong, "SRED: stabilized RED," in *Proc. of IEEE Infocom*, New York, Mar. 1999, pp. 1346–1355.

[27] D. Tong and A. L. N. Reddy, "QoS enhancement with partial state," in *Proc. of IWQoS*, London, UK, Jun. 1999, pp. 87–96.

[28] Smitha and A. L. N. Reddy, "LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers," in *Proc. of Globecomm*, San Antonio, Texas, Nov. 2001, pp. 2311–2315.

[29] P. Achanta and A. L. N. Reddy, "Design and evaluation of a partial state router," in *Proc. of ICC*, Paris, France, Jun. 2004, Available via http://ee.tamu.edu/˜reddy/papers/lru-fq.pdf.

[30] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proc. of 7th. International Workshop on Quality of Service (IWQoS '99)*, London, UK, Jun. 1999, pp. 260–262.

[31] L. Le, J. Aikat, K. Jeffay, and F.D. Smith, "The effects of active queue management on web performance," in *Proc. of ACM Sigcomm*, Karlsruhe, Germany, Aug. 2003, pp. 265–276.

[32] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. of IEEE Infocom*, Tel Aviv, Israel, Mar. 2000, pp. 1435–1444.

[33] S. Darbha and K. R. Rajagopal, "Limit of a collection of dynamical systems: An application to modeling the flow of traffic," *Mathematical Modeling and Methods in Applied Sciences*, vol. 12, no. 10, pp. 1381–1399, Oct. 2002.

[34] M. Vidyasagar, *Nonlinear Systems Analysis*, Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2nd edition, 2002.

[35] A. M. Lyapunov and A. T. Fuller, *The General Problem of the Stability of Motion*, London, UK: Taylor and Francis, Inc., 1992.

[36] K. L. Cooke and W. Huang, "On the problem of linearization for state-dependent delay differential equations," *Proc. Amer. Math. Soc.*, vol. 124, no. 5, pp. 1417–1426, May 1996.

[37] K. J. Åström and B. Wittenmark, *Adaptive Control*, Reading, MA: Addison Wesley, 1995.

[38] L. Ljung and E. J. Ljung, *System Identification: Theory for the User*, Upper Saddle River, NJ: Pearson Education, 2nd edition, 1999.

[39] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control*, Upper Saddle River, NJ: Prentice Hall, 1984.

[40] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, Reading, MA: Addison-Wesley, 2nd edition, 1989, p.289.

[41] J. Green and K. Hedrick, "Nonlinear speed control of engines," in *Proc. of the American Control Conference*, San Diego, California, 1990.

[42] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*, Upper Saddle River, NJ: Prentice Hall, 1989.

[43] P. Huang, A. Feldmann, and W. Willinger, "A non-intrusive, wavelet-based approach to detecting network performance problems," in *Proc. of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, California, Nov. 2001, pp. 213–227.

[44] I. Yeom and A. L. N. Reddy, "Realizing throughput guarantees in a differentiated services network," in *Proc. of IEEE Conf. on Multimedia Computing and*

Systems, Florence, Italy, Jun. 1999, pp. 372–376.

[45] W. Almesberger, “Linux traffic control — implementation overview,” Tech. Rep. SSC/1998/037, EPFL, Nov. 1998, Available via ftp://lrcftp.epfl.ch/pub/people/almesber/pub/tcio-current.ps.gz.

[46] “Improving calculation of sqrt(x),” Sept. 2003, Available via http://www.mprv.biz/iwb/practice/examples/sqrt.html.

[47] H. Jiang and C. Dovrolis, “Passive estimation of tcp round-trip times,” *ACM SIGCOMM Computer Communications Review*, vol. 32, no. 3, Jul. 2002.

[48] National Institute of Standards and Technology, “Nist Net,” 2003, http://snad.ncsl.nist.gov/itg/nistnet/.

[49] Mindcraft, “WebStone,” 2002, http://www.mindcraft.com/webstone.

[50] The Apache Software Foundation, “Apache HTTP Server Project,” 2003, http://httpd.apache.org/.

[51] S. Floyd, “RED: Discussion of setting parameters,” Nov. 1997, Available via http://www.icir.org/floyd/REDparameters.txt.

[52] S. Floyd, “TCP and explicit congestion notification,” *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, Oct. 1994.

[53] S. Floyd, “Notes on Evaluating ECN,” Apr. 2002, Available via http://www.icir.org/floyd/ecn.html.

APPENDIX A

ESTIMATION ALGORITHM WITH CBR TRAFFIC

In eqn. 2.1-2.3, we define a model of heterogeneous traffic with the CBR traffic as a representative for non-responsive traffic. For the simplicity and easy understanding of our algorithm, we further approximate $X_u R(t)$ to be constant because of the sufficient small deviation in measured $R(t)$ (refer to Chapter II - Section A). In this section, we would like to derive our estimation algorithm without the approximation.

We first define $d_u = N_u X_u$. d_u is the aggregate sending rate of N_u CBR flows. The definition of $z(t)$ is the same as it is in Chapter II - Section B. So in terms of $z(t)$ and d_u, the traffic mix model is given by:

$$\dot{z}(t) = \frac{N_s}{R(t)} - \frac{z(t)z(t-R(t))}{2N_s R(t)} p(t-R(t)), \tag{A.1}$$

$$\dot{q}(t) = \frac{z(t)}{R(t)} + d_u - C, \tag{A.2}$$

$$R(t) = \frac{q(t)}{C} + T_p, \tag{A.3}$$

where $\dot{z}(t) = N_s \dot{W}_s(t)$.

By being taken the second derivative of $q(t)$, eqn. A.2 becomes:

$$\ddot{q}(t) = \frac{\dot{z}(t)}{R(t)} - \frac{z(t)}{R^2(t)} \dot{R}(t). \tag{A.4}$$

By combining eqn. A.4 with eqn. A.1 and then making some simple mathematical manipulations, one can get:

$$\begin{aligned} R(t)\ddot{q}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t) = {} & \frac{N_s}{R(t)} \\ & - \frac{z(t)z(t-R(t))}{2N_s R(t)} p(t-R(t)) + d_u \frac{\dot{q(t)}}{C}. \end{aligned} \tag{A.5}$$

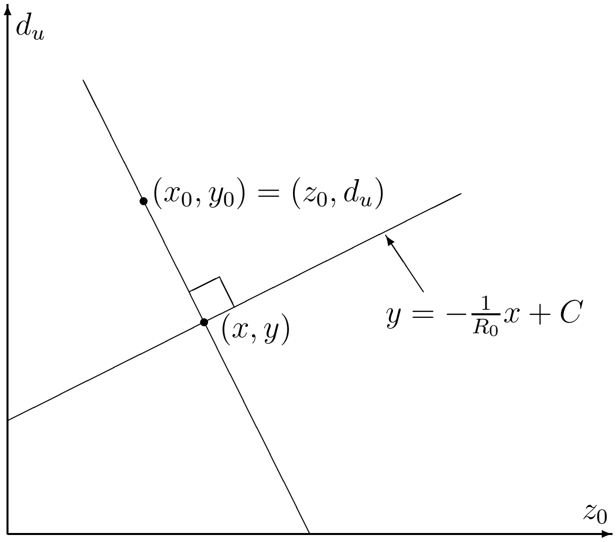


Fig. 32. Projection of Estimation Results

Taking the small signal approximation pointed out in section B yield:

$$R_0\ddot{q}(t) + (\frac{\dot{q}(t)}{C} + 1)\dot{q}(t)$$
$$= \begin{bmatrix} \frac{1}{R_0} & -\frac{p\big(t-R_0\big)}{2R_0} & \frac{\dot{q(t)}}{C} \end{bmatrix} \begin{bmatrix} N_s \\ \frac{z_0^2}{N_s} \\ d_u \end{bmatrix}. \tag{A.6}$$

So there are 3 unknown parameters to estimate, i.e. $N_s, z_0,$ *and* d_u, instead of 2 unknown parameters, i.e. N_s *and* z_0, in Chapter II - Section B. Following the same procedure presented in Chapter II - Section B, this estimation algorithm can recursively update 3 unknown parameters with the normalized Kaczmarz's projection algorithm.

In the equilibrium state, we have $\dot{q}(t) = 0$. So one gets from eqn. A.2:

$$C = \frac{z_0}{R_0} + d_u.$$

The equation above constrains our estimation results of both z_0 and d_u, when the link utilization is close to 100%. To bound our estimation results within the constraint, each pair of the estimation results needs to be projected to the constraint line to get the correct estimation value as shown in Figure 32. (x, y) is the pair of results after projection, while (x_0, y_0) is the direct output from our estimation algorithm.

Here we assign: $x_0 = z_0, y_0 = d_u, m = -\frac{1}{R_0}$. So we get 2 functions of vertical crossing lines:

$$\begin{aligned} \textit{Line } 1 \quad &: \quad y = mx + C, \\ \textit{Line } 2 \quad &: \quad y = -\frac{1}{m}x + y_0 + \frac{x_0}{m}. \end{aligned} \tag{A.7}$$

By solving eqn. A.7 with regarding to $x \; and \; y$, one gets:

$$\begin{aligned} x &= \frac{my_0 + x_0 - mC}{m^2 + 1}, \\ y &= mx + C. \end{aligned} \tag{A.8}$$

Then plug in $x_0, y_0, \; and \; m$ into eqn. A.8 to get projected estimation results$(x \; and \; y)$.

VITA

Zhili Zhao received his B.S. in electronic technology from East China Normal University, Shanghai, China, in July 1994 and M.S. in optoelectronics from Shanghai Institute of Technical Physics, Academia Sinica, Shanghai, in April 1997. He received his Ph.D. in computer engineering, Department of Electrical Engineering, Texas A & M University in May 2004. His current research interest is Internet traffic estimation and regulation.

Zhili Zhao's permanent address is Apt 15/502, No.3 ECNU Staff Village, 895 Jin Sha Jiang Road, Shanghai 200062, P.R.China.

The typist for this thesis was Zhili Zhao.