A General Framework for Flow Control in Wireless Networks

by

Minghua Chen

B.Eng. (Tsinghua University) 1999

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2006

A General Framework for Flow Control in Wireless Networks

Abstract

A General Framework for Flow Control in Wireless Networks

by

Minghua Chen
Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Avideh Zakhor, Chair

Flow control, including congestion control for data transmission, and rate control for multimedia streaming, is an important issue in information transmission for both wired and wireless networks. Proper flow control allows users to fairly and fully utilize available bandwidth without the possibility of congestion collapse. Failure to apply proper flow control may result in serious performance degradation in a network.

Although the problem of flow control has been successfully addressed in wired networks, it is still open in wireless networks. Current widely accepted solutions, such as TCP, assume that congestion is the only cause of packet loss, and as such, are not applicable to wireless networks in which the bulk of packet loss is due to errors at the physical layer. We show that this often results in bandwidth underutilization in wireless networks. This problem is becoming increasingly more serious as wireless data and multimedia services are being rapidly deployed commercially on carries throughout the world with data rates of up to one Mbps.

In this thesis, we first formulate flow control in wireless networks as a convex optimization problem. We then propose a new class of solutions that properly adjust the number of connections of a user, to fully utilize wireless bandwidth and minimize end-to-end packet loss. Our solution differs from all existing schemes introduced in the past decade in the following ways:

- First, it is theoretically guaranteed to be optimal, stable and scalable. Practically, in a network with *arbitrary* topology, *arbitrary* number of users, and *arbitrary* initial source rates, our proposed schemes guarantee all users' source rates to globally exponentially converge to an equilibrium. This convergence guarantees no congestion collapse in the network. At the equilibrium, all bottlenecks are fully utilized and users are fair to each other. Furthermore, proposed schemes are fair to TCP/TFRC protocols, and are therefore amenable to incremental deployment in the current Internet where TCP is dominant.

- Second, our proposed schemes are end-to-end and require modifications to neither infrastructure nor transport protocol stack, making it easy to deploy in practice.

Based on this approach, we have designed practical schemes for data transmission over wireless networks, and characterized their performance using simulations and actual experiments over the Verizon Wireless 1xRTT and EVDO CDMA data networks.

This work implicitly provides a general framework for flow control. In this framework, both users' rates and the number of connections they open are properly controlled to pursue equilibrium in the network. We show it is sufficient to control users' rates and their number of connections *independently* in two separate timescales, in order to guarantee convergence to the desired equilibrium. This two timescale approach allows modification of the control law in one timescale without affecting the one in the other timescale, or the system's convergence. This framework is general in the sense that its usage is not limited to the problem we study in this thesis, which serves as an ideal platform to demonstrate the power of this approach.

Professor Avideh Zakhor
Dissertation Committee Chair

To my dear mother and father.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Flow control, including congestion control for data transmission, and rate control for multimedia streaming, is an important issue in information transmission for both wired and wireless networks. Since the network is shared among all users and the connections are stateless, no dedicated bandwidth is allocated to any end-to-end connection. Therefore, senders and receivers should have a mechanism to determine the sending rate at which packets are injected into the network.

Flow control is the mechanism to distributively determine sending rates of users to achieve the following goals: (a) full utilization of bottleneck links by ensuring sending rates are not too low; (b) preventing congestion collapse by ensuring sending rates are not too aggressive. For example there was an actual network collapse of the Internet in Oct. 1986 at University of California at Berkeley resulting in serious performance degradation [1, Section 1]. Further, it would be ideal for sending rates to converge to an equilibrium, such that there is no fluctuation in the received throughput, resulting in constant quality in multimedia streaming applications. Finally, it should ensure fairness between users sharing common links. The mechanism is required to be distributive in the sense that the control must be done based only on local information. This is because networks are growing into arbitrarily large size, and any centralized solution does not necessarily scale.

Table 1.1: Flow control solutions for data transmission and multimedia streaming in wired and wireless networks.

	Wired Network	Wireless Networks
DATA	TCP	?
MULTIMEDIA	TFRC	?

Transport Control Protocol (TCP), a widely accepted congestion control protocol, has been extremely successful on the wired Internet since its first implementation by Jacobson in 1988 [1]. TCP Reno, the most popular TCP version today, increases its window size by one in its congestion avoidance stage if no packet is lost in the previous round trip time, and halves the window size otherwise. Similarly, TCP-Friendly Rate Control (TFRC) is a rate control protocol for multimedia streaming, proposed by Floyd et. al. [2]. There are basically three advantages to rate control using TFRC: first, it does not cause network instability, thus avoiding congestion collapse. Second, it is fair to TCP flows, which is the dominant source of traffic on the Internet. Third, the TFRC's rate fluctuation is lower than TCP, making it more appropriate for multimedia streaming applications which require constant quality. The *key* assumption behind TCP and TFRC is that packet loss is a sign of congestion. TCP and TFRC have been shown to work well in wired networks. As a result, every computer and handheld device today runs TCP, and every router signals congestion by dropping packets.

In wireless networks however, packet loss can also be caused by physical channel errors, thus violating this assumption. Neither TFRC nor TCP can distinguish between packet loss due to buffer overflow and that due to physical channel errors, resulting in underutilization of the wireless bandwidth. Particularly, our experiments over Verizon Wireless 1xRTT wireless data network have shown that TFRC achieves only 56% of the available wireless bandwidth [3]; Balakrishnan et. al. have shown that TCP Reno achieves only 22% utilization in wireless LAN environments [4]. Hence streaming rate control and congestion control over wireless networks are still open issues, as indicated in Table 1.1 .

The need to solve the problem is becoming urgent as wireless data and streaming services are becoming increasingly more popular:

- Wireless bandwidth is increasing. For example, Verizon wireless EVDO service provides up to 2 Mbps bandwidth for a single user. Such high bandwidth can support high bit-rate video streaming applications with quality close to DVD using advanced video coding techniques such as H.264 [5]. Such high bandwidth also implies users can download music files on the order of seconds, or engage in instantaneous online gaming.

- Handheld wireless devices are becoming powerful. For example, Lenovo cell phone model ET560 has an Intel X-Scale 400Mhz processor and 64MB memory. These powerful handheld devices can support on-the-fly processing of received video and music. Hence users can interact with multimedia information at the same time as they are being received.
- Users can use these advanced applications for up to hours at a time, thanks to the improved battery performance.

1.2 Previous Work on TCP/TFRC over Wireless

There have been a number of efforts to improve the performance of TCP or TFRC over wireless [4, 6–26]. These approaches either hide end-hosts from packet loss caused by wireless channel error, or provide end-hosts the ability to distinguish between packet loss caused by congestion, and that caused by wireless channel error. To gain a better understanding of the spectrum of approaches to rate control over wireless, we briefly review TCP and TFRC solutions over wireless; we will provide a fundamental overview of all these solutions in Section 3.2.

Snoop, a well-known solution, is a TCP-AWARE local retransmission link layer approach [4]. A Snoop module resides on router or base station on the last hop, which is assumed to be wireless, and records a copy of every forwarded packet. Assuming snoop module can access TCP acknowledgement packets (ACK) from the TCP receiver, it looks into the ACK packets and carries out local retransmissions when a packet is identified to be corrupted by wireless channel errors. While doing the local retransmission, the ACK packet is suppressed and not forwarded to the TCP sender. Other similar approaches based on local link layer retransmission include [11, 14–18]. These schemes can potentially be extended to TFRC in order to improve performance, by using more complicated treatment of ACK packets from TFRC receiver.

Explicit Loss Notification (ELN) can also be applied to notify TCP/TFRC sender when packet loss is caused by wireless channel errors rather than congestion [6, 19]. In this case, TFRC can take into account only the packet loss caused by congestion when adjusting the streaming rate.

End-to-end statistics can be used to detect congestion when a packet is lost [7–10, 12, 13, 20–24, 26]. For example, by examining trends in the one-way delay variation, Parsa and Garcia-Luna-Aceves [23] interpret loss as a sign of congestion if one-way delays are increasing, and a sign of wireless channel error otherwise. One-way delay can be associated with congestion in the sense that it monotonically increases if congestion occurs as a result of increased queueing delay, and remains constant otherwise. Similarly, Barman and Matta have proposed a loss differentiation scheme based on the assumption that the variance of round trip time is high when congestion occurs, and is low otherwise [7].

Cen et. al. present an end-to-end based approach to facilitate streaming over wireless [21]. They combine packet inter-arrival interval and relative one way delay to differentiate between packet loss caused by congestion and that due to wireless channel errors. There are two key observations behind their approach; first, relative one way delay increases monotonically if there is congestion; second, inter-arrival interval is expected to increase if there is packet loss caused by wireless channel errors. Therefore, these two statistics can differentiate between congestion and wireless errors. Nevertheless, the high wireless error misclassification rate may result in underutilizing the wireless bandwidth, as shown in [21]. Yang et. al. [26] also propose a similar approach to improve video streaming performance in presence of wireless error, under the assumption that wireless link is the bottleneck.

Other schemes such as [8–10, 12, 13, 20] that use end-to-end statistics to detect congestion, can also be combined with TFRC for rate control. The congestion detection scheme can be used to determine whether or not an observed packet loss is caused by congestion; TFRC can then take into account only those packet loss caused by congestion when adjusting streaming rate.

Tang et. al. proposed the idea of using small dummy packets to actively probe whether the network is congested in case of packet loss, so as to differentiate between packet loss due to congestion and that due to channel error [25]. Yang et. al. [27] propose a cross-layer scheme that uses link layer information to determine whether a packet loss is caused by channel error or congestion, assuming that only the last link is wireless. In this approach, when a packet is lost, TFRC goes beyond layering abstraction and enquiries the link layer about the recent signal strength. The packet loss is recognized to be a result of wireless channel error if recent signal strength is low, and due to congestion otherwise.

The disadvantage of end-to-end statistics based approaches is that congestion detection schemes based on statistics are not sufficiently accurate, and they either require cross layer information or modifications to the transport protocol stack.

Another alternative is to use non-loss based rate control schemes. For instance, TCP Vegas [28], in its congestion avoidance stage, uses queueing delay as a measure of congestion, and hence could be designed not to be sensitive to any kind of packet loss, including that due to wireless channel error. It is also possible to enable the routers with ECN marking capability to do rate control using ECN as the measure of congestion [29]. As packet loss no longer corresponds to congestion, ECN based rate control does not adjust sending rate upon observing a packet loss.

In summary, although flow control in wired networks has been successfully addressed, extending the known solutions, i.e. TCP and TFRC, to wireless scenarios is not trivial. All existing solutions either require support from network infrastructure such as routers and base stations, or require modifications to the transport layer protocol stack, in the operating systems of every computer and handheld device. Infrastructure providers such as Cisco, and operating system providers such as Microsoft are reluctant to carry out such massive modifications to their products.

Therefore, an open question of practical interest is the following:
Is it possible to solve the problem of flow control in wireless networks, without changing today's network infrastructure, operating systems, protocol stack, or violating the end-to-end principle?

1.3 Previous Work on Flow Control

In the past eight years, there has been a great deal of theoretical research on understanding and designing distributed end-to-end network flow control algorithms. A widely recognized setting has been introduced by Kelly et. al. in the seminal work [30], and is based on a fluid-flow approximation of packets propagating over links; it associates a utility function to each flow, a cost function to each resource, and maximizes the aggregate *net* system utility function. Under this framework, flow control schemes can be viewed as algorithms to compute the optimal solution to this maximization problem. Kelly and his colleagues have proposed two complementary flow control algorithms, the *primal* and the *dual* [30].

In primal algorithms, the users adapt their sending rates dynamically based on the costs that incur along the path through the network, while the routers determine their prices directly from the arrival rates at the links according to a static law. Hence the primal algorithms are end-to-end, with only simple feedback prices from the network. One example of pricing strategy, which is used in TCP Reno, exploits the packet loss rate. The algorithms analyzed by Kunniyur and Srikant [31], Alpcan and Basar [32], Vinnicombe [33] belong to this class. The stability of the various primal algorithms is investigated in [30, 33–35].

In dual algorithms, on the other hand, the routers adapt the prices dynamically based on the link rates, and the users select a static law to determine the source rates directly from the prices along the path and the source parameters. These schemes rely on the network resources to implement the congestion control. The algorithms proposed by Low and Lapsley [36], Paganini et. al. [37], Yaiche et. al. [38] belong to this class. The stability properties of various dual algorithms are investigated in [30, 37].

There is also another class of algorithm named primal-dual, where both the prices and the source rates are updated dynamically by routers and users, respectively. The algorithms proposed by Low and Lapsley [36], Kunniyur and Srikant [39], Paganini et. al [37] belong to this class. The stability of various primal-dual algorithms is investigated in [36, 37, 39].

All these frameworks can be used to understand and design the congestion control algorithms and predict their performance. For example, Low has pointed out that different versions of TCP, as well as queue management algorithms such as DropTail and RED, can be analyzed under the same duality model with different utility and update functions [40]. Kunniyur and Srikant have investigated two algorithms that can be used to understand the behavior of TCP [31]. Kelly has shown TCP to be a primal like algorithm with packet loss rate as the associated price function [39]; in this work the stability for the system with and without delay, with and without disturbance, are reviewed and further investigated. Kelly's paper also discusses the selection of the TCP parameters, in order to achieve a scalable robust congestion control.

Specifically, networks consisting of TCP Reno and routers implementing DropTail or RED, have shown to achieve all flow control goals:

- Users adjust sending rate based on only the end-to-end packet loss observed, and routers drop packets only based on the difference between aggregate incoming rate and links' capacities. Hence, all the algorithms can be implemented in a distributed manner.

- It has been shown that sending rates controlled by TCP Reno converge to an optimal and unique equilibrium exponentially fast [41]. Optimality of the equilibrium lies in the fact it maximizes an aggregate net utility [40, 42].

- At the converged equilibrium, all bottleneck links are fully utilized, and users are fair to each other under certain fairness criteria [42].

Nevertheless, all these frameworks either work for wired networks only, or require additional functionalities from infrastructures, or require change to the existing transport layer protocol stack. These either limit their applicability to wireless networks, or would make them hard to be deployed in practice.

Therefore, an open question of theoretical interest is the following:
What is the framework for flow control problem in wired or wireless networks, without modifying today's network infrastructure, operating systems, or the protocol stacks?

1.4 Thesis Organization

In this thesis, we aim at providing answers to the two open questions mentioned in the previous sections. We assume a wireless link is associated with a fixed bandwidth and a fixed packet loss rate caused by the physical channel errors. We first show that in the presence of the packet loss caused by physical channel error, flow control in the wireless network can be formulated as the same concave optimization problem defined by Kelly in wired networks [39]. TCP and TFRC in the wireless networks pursue the optimal solution using inaccurate feedback[1]. All existing approaches to this problem correct the inaccurate feedback by casting modifications to existing protocols, such as TCP, or infrastructure elements such as routers, thereby making them hard to deploy in practice.

In this thesis, we formulate the problem as another concave optimization problem with a different utility function, followed by a new class of solutions. Our approach is end-

[1] In [43, 44], we have also shown the similar formulation using the framework in [30].

to-end, and achieves reasonable performance by adjusting the number of users' connections according to a properly selected control law. The control law is based on only one bit of information, which can be reliably measured at the application layer. We show that the resulting control system has a unique stable equilibrium that solves the concave optimization problem, implying scalability and optimality of the solution. We then apply our results to design a practical rate control scheme for data transmission over wireless networks, and characterize its performance using NS-2 simulations, and actual experiments over Verizon Wireless 1xRTT and EVDO CDMA data network. Analysis and simulation results indicate our scheme is applicable to both wired and wireless scenarios.

This thesis is organized as follows. In Chapter 2, we discuss the available design space, and study a simple case of streaming over one wireless link in order to gain intuition about the problem. Problem formulation and analysis are included in Chapter 3. A new approach addressing the problem is proposed in Chapter 3, together with analysis for its optimality and stability. Chapter 4 shows the design of practical schemes following the insights derived from theoretical analysis of Chapter 3. Trade-off analysis between bandwidth utilizations and responsiveness over a single user, single wireless link scenario are also included in Chapter 4. NS-2 simulations and actual experiments over 1xRTT and EVDO CDMA data networks are included in Chapter 5. Chapter 6 concludes the thesis with discussions and future work.

Chapter 2

Design Space and A Simple Case Study

In this chapter, we first argue that our design space is the application layer, since modifications to underlying layers are overly restrictive in our setting. We study a simple case of streaming over one wireless link in order to gain intuition on how modifications to application layer can improve wireless bottleneck utilization.

2.1 Design Space

In previous chapter, we stated our objective of not requiring modifications to network infrastructure such as routers, or transport protocol stack such as TCP. As seen from Figure 2.1, this implies keeping IP and underlying physical layers intact. Also no modification to transport protocol stack implies intact transport layer. Therefore, the only available design space is the application layer.

In the next section, we will intuitively show what can be done in application layer to address the problem of flow control in wireless networks.

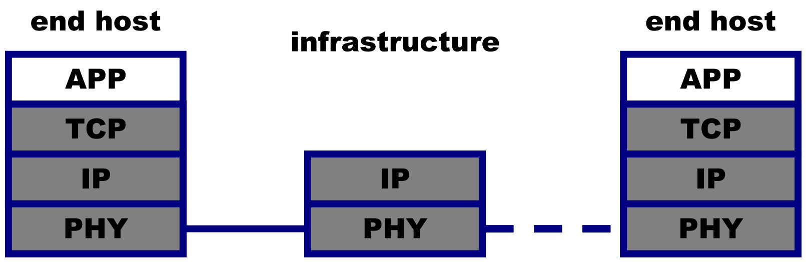


Figure 2.1: Layering abstraction of packets traveling across network. Under the requirement of no modification to infrastructure and transport layer protocol, the only available design space lies in application layer.

2.2 A Simple Case and Resulting Intuition

A simple scenario for data transmission and streaming over one wireless link is shown in Figure 2.2 where a server s in the wired network is sending data or streaming video to a receiver r in the wireless network. The wireless link is assumed to have available bandwidth B_w, and packet loss rate p_w, caused by wireless channel error. There could also be packet loss caused by congestion at node 2, denoted by p_c. The end-to-end packet loss rate observed by receiver is denoted by p, and the streaming rate is denoted by T. We refer to the wireless channel as underutilized if the streaming throughput is less than the maximum possible throughput over the wireless link, i.e. $T(1-p) < B_w(1-p_w)$.

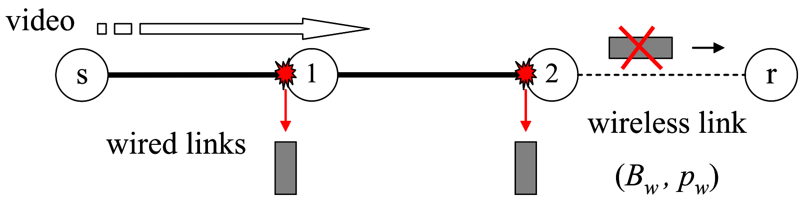


Figure 2.2: Typical scenario for streaming over wireless.

Given this scenario, we assume the following. First, there are no cross traffic at either node 1 or node 2. Second, in the long term, the wireless link is assumed to be the bottleneck. By this, we mean there is no congestion at node 1. Third, we assume there is no congestion and queuing delay at node 2 if and only if wireless bandwidth is underutilized, i.e. we achieve $p_c = 0$ and minimum round trip time, defined as RTT_{min}, if and only if

$T \leq B_w$. When $T > B_w$, we have $p_c \geq 0$ and $rtt \geq RTT_{min}$. Fourth, B_w and p_w are assumed to be constant, at least on the timescale analysis is carried on; packet loss caused by wireless channel error is assumed to be random and stationary. Fourth, for simplicity, the backward route is assumed to be error-free and congestion-free.

As pointed out in literature, the problem of TCP/TFRC over wireless is underutilization [3,4]. We now investigate the sufficient and necessary condition for underutilization for this very simple scenario to gain intuition on how to solve it.

2.2.1 A Sufficient and Necessary Condition for Underutilization

We use the following model for TCP/TFRC sending rate in the analysis [45]:

$$T = \frac{kS}{rtt\sqrt{p}}, \tag{2.1}$$

where T represents the sending rate, S is the packet size, rtt is the end-to-end round trip time, p is the end-to-end packet loss rate, and k is a constant factor. Although this model has been refined to improve accuracy [2,46], it is simple, easy to analyze, and more importantly, captures all the fundamental factors that affect the sending rate. Furthermore, the results we derive based on this simple model can be extended to other more sophisticated models, such as the one used in [2].

The overall packet loss rate is p, a combination of p_w and p_c, and can be written as:

$$p = p_w + (1 - p_w)p_c. \tag{2.2}$$

This shows that p_w is a lower bound for p, and that the bound is reached if and only if there is no congestion, i.e. $p_c = 0$. Combining this observation and Equation (2.1), an upper bound, T_b, on the streaming rate of one TFRC connection can be derived as follows:

$$T \leq \frac{kS}{RTT_{min}\sqrt{p_w}} \equiv T_b \tag{2.3}$$

If there is no congestion, i.e. $p_c = 0$, and hence no queuing delay caused by congestion, we get $rtt = RTT_{min}$, $p = p_w$, and T achieves the upper bound $T = T_b$ in Equation (2.3). In this case, the throughput is $T_b(1 - p_w)$, which is the upper bound of throughput given one TFRC connection for the scenario shown in Figure 2.2. Based on these, we can state the following theorem:

Theorem 2.2.1. Given the above scenario and assumptions, a sufficient and necessary condition for one TFRC connection to under-utilize wireless link is

$$T_b < B_w. \tag{2.4}$$

Proof. Since $T_b(1 - p_w)$ is the upper bound of one TFRC's throughput, clearly Equation (2.4) implies under-utilization of the wireless channel, and hence the "sufficient" part of the Theorem is obvious. To see the necessary part, note that if under-utilization happens, i.e. $T(1 - p) < B_w(1 - p_w)$, then no congestion happens, thus $rtt = RTT_{min}$, $p = p_w$ and $T = T_b$, resulting in $T_b(1 - p) < B_w(1 - p_w)$. □

Theorem 2.2.1 implies that if the available bandwidth is larger than the highest sending rate one TCP/TFRC can achieve, then underutilization happens. In essence, the approaches taken in [4, 6, 8–24, 26] ensure the condition in Equation (2.4) is not satisfied, through modifications to network infrastructure or protocols. For example in the TFRC-AWARE Snoop-like solution, p_w becomes effectively zero after local retransmissions, and thus Equation (2.4) can never be satisfied. By effectively setting $p_w = 0$, Snoop-like module translates the new problem, i.e. rate control for streaming over wireless, into an old one, i.e. rate control for streaming over wired networks, for which a known solution exists. Similar observations can be made for other solutions such as the end-to-end statistics based approaches [8–10, 12, 13, 20–24, 26]. Similarly, ELN and end-to-end statistics based approaches make TFRC not respond to packet loss caused by wireless channel errors, thus not taking p_w into account when adjusting streaming rate. This is effectively the same as setting $p_w = 0$, thus improving the performance of the TFRC connection.

Theorem 2.2.1 also indicates the two regions in which TCP operates, as shown in Figure 2.3. In the wired scenario, TCP keeps increasing its rate until the rate hits wired link capacity and packet loss due to router queue overflow is observed. Therefore, TCP's rate is limited by the link capacity, operating in capacity-limited region. In wireless scenario, however, TCP can not differentiate between packet loss caused by congestions and that due to physical layer errors. Consequently, TCP halves its rate when packet loss caused by wireless channel error is observed, which might happen far before TCP's rate reaches the

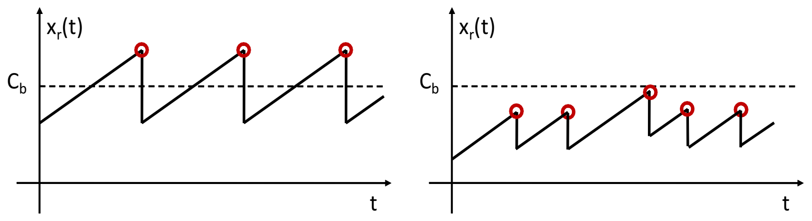


Figure 2.3: Sending rate of TCP: limited by link capacity in wired scenario (left); limited by wireless channel error in the wireless scenario (right).

link capacity. Therefore, in channel-error-limited region, TCP's rate is limited by wireless channel error, resulting in underutilization. On the other hand, TCP achieves full utilization if it operates in the capacity-limited region. Similar analysis and intuitions apply to TFRC as well.

2.2.2 A Strategy to Reach the Optimal Performance

It is not necessary to avoid the condition in Equation (2.4) in order to achieve full utilization for one *application*. This is because it is conceivable to use multiple simultaneous connections for a given streaming application. The total throughput of the application is expected to increase with the number of connections until it reaches the hard limit of $B_w(1 - p_w)$. Intuitively, although each single TCP/TFRC still operates in channel-error-limited region, the aggregate rates of multiple parallel TCP/TFRC can achieve the link capacity, and hence improve utilization.

Analysis on the Optimal Number of Connections

Given the scenario shown in Figure 2.2, and the associated assumptions, we now argue that multiple connections can be used to achieve optimal performance, i.e. throughput of $B_w(1 - p_w)$, and packet loss rate of p_w. To see this, let us consider a simple example in which

$$B_w(1 - p_w) = \frac{2.5kS}{RTT_{min}\sqrt{p_w}}(1 - p_w) = 2.5T_b(1 - p_w)$$

By opening one TCP/TFRC connection with packet size S, the application achieves a throughput of $\frac{kS}{RTT_{min}\sqrt{p_w}}(1 - p_w) = T_b(1 - p_w)$ and packet loss rate of p_w. This is because according to Theorem 2.2.1, underutilization implies $rtt = RTT_{min}$, $p = p_w$ and $T =$

$\frac{kS}{RTT_{min}\sqrt{p_w}} = T_b$.

Let us now consider the case with two TCP/TFRC connections from sender s to receiver r in Figure 2.2. It is easy to see that p_w for each of the two TCP/TFRC connections remain unchanged from the case with one TCP/TFRC connection. Thus the throughput upper bound for each of the two TCP/TFRC connections is $\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w) = T_b(1-p_w)$, and the aggregate throughput upper bound for both of them is $2\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w) = 2T_b(1-p_w)$, which is smaller than $B_w(1-p_w)$, implying channel under-utilization. Hence $rtt = RTT_{min}$, $p_c = 0$, and thus $p = p_w$. The throughput for each connection is then $\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w)$. Consequently, the total throughput for both connections is $2\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w)$ with packet loss rate at p_w.

A similar argument can be repeated with three TCP/TFRC connections, except that the wireless channel is no longer under-utilized and $rtt > RTT_{min}$. Furthermore, if the buffer on node 2 overflows then p_c will no longer be zero and hence using Equation (2.2) we get $p > p_w$. In this case the wireless link is still fully utilized at $T(1-p) = B_w(1-p_w)$, but round trip time is no longer at the minimum value RTT_{min}, and overall packet loss rate p could exceed p_w, i.e. the overall packet loss rate in the two connections case.

In general, given B_w, p_w, and the packet size S for each connection, it can be shown that when full wireless channel utilization occurs, the optimal number of connections, n_{opt}, satisfies:

$$\begin{aligned} B_w(1-p_w) &= n_{opt}\,\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w) \\ \Rightarrow\ n_{opt}S &= B_w\,\frac{RTT_{min}\sqrt{p_w}}{k} \end{aligned} \tag{2.5}$$

Thus what really matters is the product of n_{opt} and S, and it is always possible to achieve full wireless channel utilization by choosing n_{opt} to be an integer, and by selecting S accordingly[1]. It is also possible to analyze the case with different packet sizes for different connections, but this is harder to do, and it is not fundamentally different from the case with the same packet size for all connections. For the case with the fixed packet size at S, the optimal number of connections is given by

$$\left\lfloor B_w\,\frac{RTT_{min}\sqrt{p_w}}{kS}\right\rfloor \equiv \hat{n}_{opt} \tag{2.6}$$

[1]Of course p_w may also change when packet size changes, but for the sake of simplicity, we assume p_w is stable as packet size changes. Analysis can be extended given a relation between p_w and S. The point here is to change packet size to achieve finer granularity in increase/decrease.

resulting in throughput of $\hat{n}_{opt}\frac{kS}{RTT_{min}\sqrt{p_w}}(1-p_w)$ and packet loss rate of p_w.

To show that opening more than n_{opt} connections results in larger rtt, or possibly higher end-to-end packet loss rate, assume n_{opt} and S lead to the optimal performance, and consider opening $n_{opt}+\delta n$ connections, where δn is a positive integer. Denoting the end-to-end packet loss rate as p' for this case, the overall throughput is given by $(n_{opt}+\delta n)\frac{kS}{rtt\sqrt{p'}}(1-p') = B_w(1-p_w)$ and hence

$$(n_{opt}+\delta n)S = B_w\,\frac{1-p_w}{1-p'}\frac{rtt\sqrt{p'}}{k} \tag{2.7}$$

Comparing the above equation with Equation (2.5), and taking into account that the right hand sides of Equations (2.5) and (2.7) are monotonically increasing functions with respect to overall packet loss rate and round trip time, we conclude that either $rtt > RTT_{min}$ and/or $p' > p_w$.

The intuition here is that as number of connections exceeds n_{opt}, the sending rate of each connection has to decrease. Thus by Equation (2.1), the product $rtt\sqrt{p}$ has to increase, so either rtt increases or p increases, or they both increase. In practice, as the number of connections exceeds n_{opt}, initially p remains constant and rtt increases due to the increase on queueing delay at node 2, i.e. $rtt > RTT_{min}$; if the number of connections keeps increasing and buffer on node 2 overflows, rtt will then stop increasing, and p begins to increase. Eventually we get both $rtt > RTT_{min}$ and $p > p_w$.

To summarize, if there are too few TCP/TFRC connections so that the aggregate throughput is smaller than $B_w(1-p_w)$, wireless channel becomes under-utilized. If the number of connections is chosen optimally based on Equation (2.5), then wireless channel becomes fully utilized, the total throughput becomes $B_w(1-p_w)$, the $rtt = RTT_{min}$, and the overall packet loss rate is at the lower bound p_w. However, if the number of connections exceeds n_{opt}, even though the wireless channel continues to be fully utilized at $B_w(1-p_w)$, the rtt will increase beyond RTT_{min} and later on packet loss rate can exceed the lower bound p_w.

Simulations and Experimental Verification

To validate the above conclusions, we carry out both NS-2 [47] simulations and actual experiments over Verizon Wireless 1xRTT CDMA data network. The topology for NS-2 simulations is the same as the one shown in Figure 2.2 with the following settings:

$B_w = 1\ Mbps$, $RTT_{min} = 168\ ms$, $S = 1000$ bytes, and p_w varying from 0.0 to 0.16. Also, no cross traffic is introduced for illustration purposes. Within NS-2, we stream 1, 2, 4, 8, 16 and 32 TFRC connections from a fixed host to mobile hosts for 1000 seconds. The wireless link is modeled as a wired link with an exponential random packet loss model.

The results of NS-2 simulations indicating throughput, packet loss rate and round trip time as a function of wireless channel error rate, p_w, for different number of connections, are shown in Figure 2.4. There are three observations to be made. First, for a given p_w, throughput increases with the number of connections up to a point, after which there is a saturation effect. For example, for $p_w = 0.04$ we need to open at least 4 connections to maximize the throughput. Second, for a fixed p_w, opening too many connections results in either higher packet loss rate, or higher round trip time than RTT_{min}, or both; for instance, as seen from Figure 2.4, at $p_w = 0.04$, opening 8 connections results in increase in round trip time but not in packet loss rate; however, opening 16 or 32 connections results in packet loss rate to be higher than 0.04, and larger round trip time. Third, given B_w, p_w, RTT_{min}, and S, there is an "optimal" number of connections with the highest throughput and the lowest packet loss rate; for example, for $p_w = 0.04$, the optimal number of connections is around 4 or 5.

Similar experiments are carried out on Verizon Wireless 1xRTT CDMA data network. The 1xRTT CDMA data network is advertised to operate at data speeds of up to 144 kbps for one user. As we explore the available bandwidth for one user using UDP flooding, we find the highest average available bandwidth averaged over 30 minutes to be between 80 kbps and 97 kbps. In our experiments, we stream for 30 minutes from a desktop on wired network in EECS department at U.C. Berkeley to a laptop connected via 1xRTT CDMA modem using 1, 2 and 3 connections with packet size of $S = 1460$ bytes. We measure the total throughput, packet loss rate and round trip time as shown in Table 2.1. Clearly, the optimal number of connections is 2. Specifically, the loss rate is slightly higher for 3 connections than for 2, while the throughput is more or less the same for 2 and 3 connections.

Based on the above analysis and experiments on the simple case, one intuitive strategy leading to good performance can be described as follows:

Keep increasing the number of connections until an additional connection results in increase of end-to-end round trip time or packet loss rate.

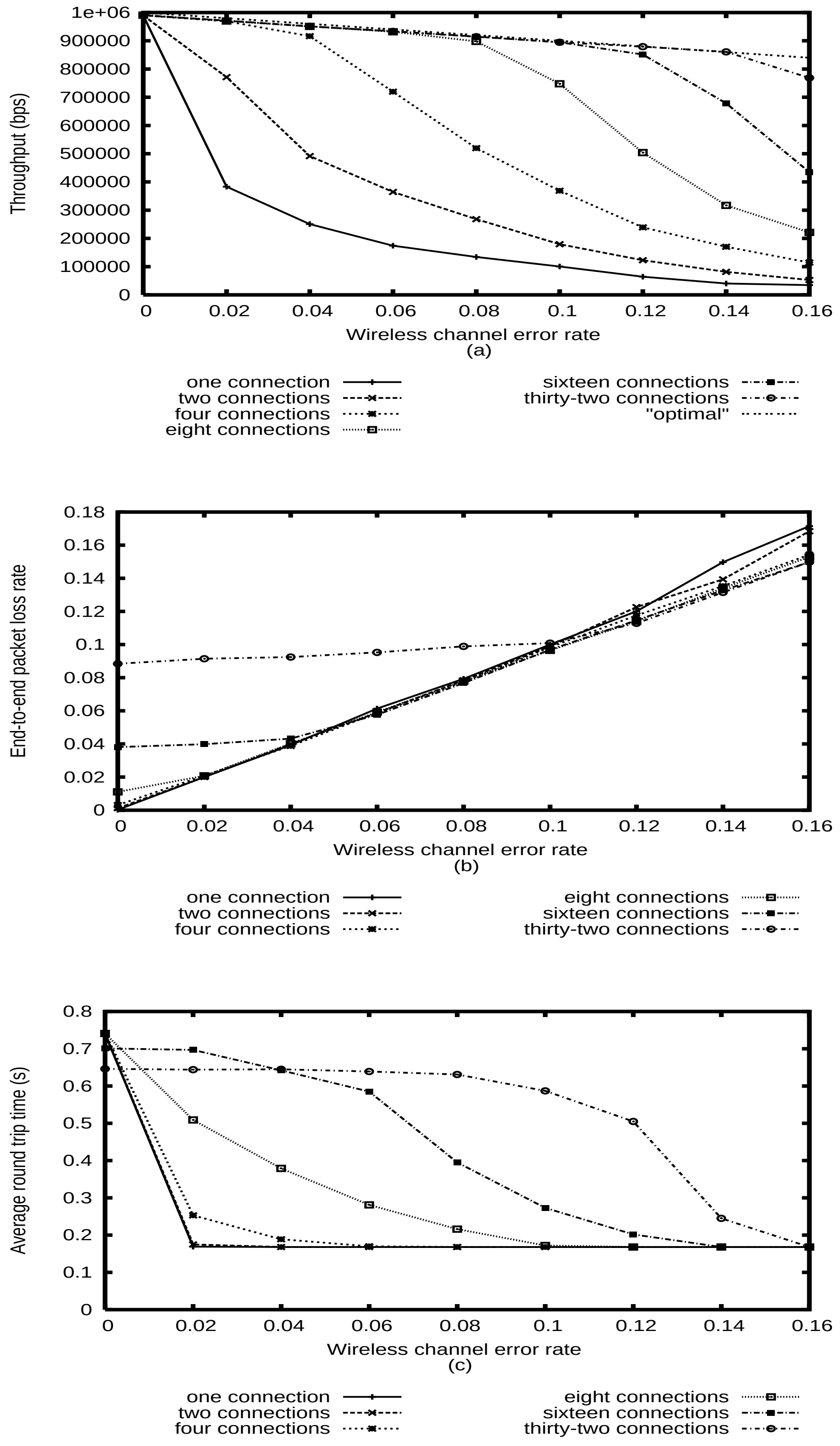


Figure 2.4: NS-2 simulations of the simple case study: (a) End-to-end throughput, (b) packet loss rate, and (c) round trip time as a function of wireless channel error rate, p_w, for different number of connections.

Table 2.1: Experimental results for Verizon Wireless 1xRTT CDMA data network.

number of conn.'s	**throughput (kbps)**	**rtt (ms)**	**pkt loss rate**
one	57	1357	0.018
two	48.2+45.6=94	2951	0.032
three	33.2+31.9+27.8=93	2863	0.046

In this chapter, we have clarified the available design space, and have developed insights on how to address the underutilization problem of flow control in wireless, by only adjusting the number of parallel TCP/TFRC connections. Although the analysis is based on a simple scenario and is very limited, the insights we gain are in fact is general, and can be extended to arbitrary scenarios. In the next chapter, we formulate the problem rigorously and derive solutions, which achieve all of desired goals and provide performance guarantees. We will see the proposed solution shares the same basic characteristics we have shown in the current chapter.

Chapter 3

Problem Formulation and Proposed Solution

In this chapter, we first review classical framework and modeling of flow control problem, and discuss the problem of TCP/TFRC over wireless, from a theoretical point of view. Then we propose a new solution, demonstrate its connections to the existing framework, and show that it meets all of our desired flow control goals. Some practical concerns, such as the effects of delay and incremental deployment of the scheme, are also addressed by theoretical analysis. Finally, we show the solution is an application of a general two timescale framework for flow control. In this framework, network and users' dynamics evolve in two different time scales, and control laws are designed for the dynamics in each timescale. Sufficient conditions for the dynamics to converge to a desired equilibrium are characterized. One advantage of this framework is that it potentially allows us to solve any flow control problem without modifying underlying network infrastructure or transport layer protocol stack.

3.1 Overview of The Flow Control Framework and TCP Modeling over Wireless

Consider a network with a set J of *resources*, i.e. links, and let C_j be the finite capacity of resource j, for $j \in J$. Let R be the set of routes, where a *route* r is a non-empty subset of J associated with a positive round trip delay T_r. We assume T_r is fixed; this is a quite natural and common assumption, at least for the time scales we are interested in, as argued in [30] and [35]. Set $a_{jr} = 1$ if $j \in r$, and set $a_{jr} = 0$ otherwise. This defines a 0-1 routing matrix $A = (a_{jr}, j \in J, r \in R)$, indicating the connectivity of the network.

Associate a route r with a user, i.e. a pair of sender and receiver, and assume users behave independently; furthermore, endow a user with a sending rate $x_r \geq 0$ and a utility function $U_r(x_r)$, which is assumed to be increasing, strictly concave, and continuously differentiable. For convenience, we also define x to be a vector of users' sending rates, i.e. $x = [x_1, x_2, \ldots]^T$.

Assume utilities are additive, so that the aggregate utility of the entire system is $\sum_{r\in R} U_r(x_r)$. The flow control problem under a deterministic fluid model, first introduced by Kelly et. al. [30] and later refined in [39], is a concave optimization problem maximizing the net utility[1]:

$$\max_{x\geq 0} \sum_{r\in R} U_r(x_r) - \sum_{j\in J} \int_0^{\sum_{s:j\in s} x_s} p_j(z)\, dz, \tag{3.1}$$

where $\int_0^{\sum_{s:j\in s} x_s} p_j(z)\, dz$ can be considered as the cost incurred at link j; $p_j(z)$ is called the price function and is required to be non-negative, continuous, increasing, and not identically zero. With these assumptions on $p_j(z)$, the objective function in Equation (3.1), the sum of users' utilities minus the costs associated with using the links, is strictly concave. One common price function used in practice is the packet loss rate, which is zero if there is no

[1]It is easy to show that this is nothing but a penalty relaxation solving the sum utility maximization with capacity constrains, namely:

$$\begin{array}{ll} \max_{x\geq 0} & \sum_{r\in R} U_r(x_r) \\ \text{s.t.} & Ax \leq C, \ \text{(i.e. } \sum_{s:j\in s} x_s \leq C_j, j \in J) \end{array}$$

congestion, and concavely increases otherwise:

$$p_j(z) = \frac{(z - C_j)^+}{z} = \begin{cases} 0, & z \le C_j; \\ 1 - \frac{C_j}{z}, & z > C_j. \end{cases}, \tag{3.2}$$

where C_j is the capacity of link j. For ease of use in the remainder of the paper, define the aggregate rate arriving at link j as follows:

$$y_j(t) = \sum_{s:j \in s} x_s(t), \quad j \in J.$$

In the rest of this thesis, we assume $p_j(y_j(t))$ is small enough such that the end-to-end packet loss rate for user r, i.e. $1 - \prod_{j \in r}(1 - p_j(y_j(t)))$, is approximately $\sum_{j \in r} p_j(y_j(t))$.

Under these settings, Kelly [39] has shown TCP Reno [2] to be a primal-like algorithm, with $x_r(t)$ satisfying:

$$\dot{x}_r(t) = \frac{1}{2S}\left(\frac{2S^2}{T_r^2} - x_r^2(t)\sum_{j \in r} p_j(y_j(t))\right), \quad r \in R, \tag{3.3}$$

and solving the optimization problem in Equation (3.1) with $U_r(x_r) = -\frac{2S^2}{T_r^2 x_r}$ where S is the TCP packet size, T_r is the end-to-end round trip time, and $p_j(y)$ takes the form in Equation (3.2). Rewriting Equation (3.1) with these quantities, the net utility function becomes:

$$\max_x \left\{ -\sum_{r \in R} \frac{2S^2}{T_r^2 x_r} - \sum_{j \in J} \int_0^{\sum_{s:j \in s} x_s} \frac{(z - C_j)^+}{z}\, dz \right\}, \tag{3.4}$$

Thus, TCP can be viewed as a discrete version of the steepest gradient descent algorithm which solves the optimization problem in Equation (3.4).

From control perspective, a network with TCP users can be modeled as a feedback control system, as shown in Figure 3.1. TCP user r uses its sending rate $x_r(t)$ as an input into the network, and adjusts $x_r(t)$ dynamically according to the feedback, i.e. congestion loss in Figure 3.1.

Equation (3.3) is a differential equation describing the time evolution of sending rate $x_r(t)$, whereby the user exploits only the aggregate packet loss information along its path. While routers drop packets and feedback the packet loss rate to TCP users based on only the difference between aggregate incoming rates and outgoing links capacities, both

[2]In the rest of the thesis, we use this version of TCP.

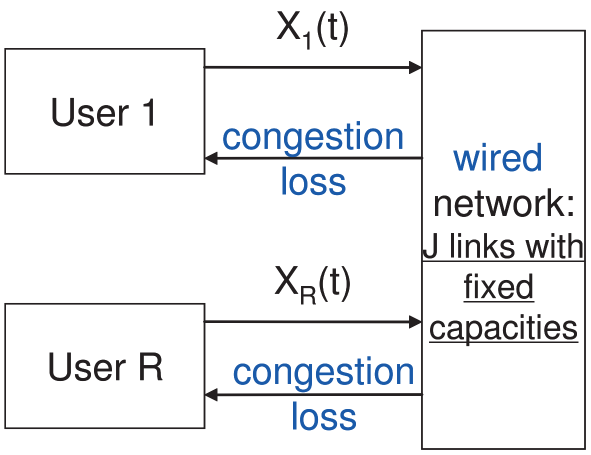


Figure 3.1: Feedback control modeling: TCP over wired networks

users and routers operate based on only local information; hence, distributed algorithms are possible to carry out the operations. Specifically, users use TCP, and routers use DropTail or RED.

In the analysis, the same flow $x_r(t)$ is assumed to be presented to all links $j \in r$, even though the flow in downstream links slightly shrinks due to loss at upstream links. This is a direct implication of our previous assumption that the packet loss rate on link j, i.e. $p_j(\sum_{s:j\in s} x_s)$, is small.

Kelly [39] showed the system in Equation (3.3) has a unique equilibrium, to which all trajectories converge, as follows:

$$x_r^o = \frac{\sqrt{2}S}{T_r\sqrt{\sum_{j\in r} p_j\left(y_j^o\right)}}, \quad r \in R; \tag{3.5}$$

where $y_j^o = \sum_{s:j\in s} x_s^o$. Equation (3.5) is similar to the well known TCP steady state throughput equation as described by Mahdavi and Floyd in [48]. TCP in wired networks achieves all desired flow control goals:

- Algorithms of users and routers are distributed.
- Users' sending rate, controlled by TCP, converge to a unique equilibrium exponentially fast [41].
- Upon the unique equilibrium, the optimization problem in Equation (3.1) is solved, and net utility is maximized.

- Upon the unique equilibrium, all bottleneck links are fully utilized. This can be seen from Equation 3.5, where finite value of x_r^o implies positivity of end-to-end packet loss rate $\sum_{j\in r} p_j\left(y_j^o\right)$, which in terms indicates the bottleneck of route r must be fully utilized and congested.

- Upon the unique equilibrium, there is roughly α-fairness [42] among users with $\alpha = 2$, implying the users are allocated rates that roughly maximize a sum of utility of the form $x_r^{1-\alpha} = x_r^{-1}$.

3.2 TCP and TFRC over Wireless

For wireless networks, we assume the links $j \in J$ are associated with not only a fixed capacity but also a packet loss rate caused by the physical channel errors, necessarily nonnegative, denoted by $\epsilon_j \geq 0, j \in J$. Nevertheless, TCP over wireless is still associated with the same concave optimization problem as with the wired networks, shown in Equation (3.4). This is because neither the utility function of users in Equation (3.4), i.e. $U_r(x_r)$, nor the cost associated with using network resources, i.e. $\int_0^{y_j} p_j(z)\,dz$, is a function of $\epsilon_j \geq 0, j \in J$. In effect, ϵ_j results in the price functions fed back to users to be inaccurate, since it now includes loss both due to congestion and physical channel errors. Hence TCP algorithms still aim to address the same optimization problem shown in Equation (3.4), but with inaccurate prices fed back from network.

From control perspective, as shown in Figure 3.2, presented wireless channel error makes the feedback (price) from network to users inaccurate. Users control sending rates based on the inaccurate feedback; as a result, the controlled sending rates might converge to an equilibrium that causes underutilization in links.

This inaccurate feedback price function, denoted by $q_j(y_j(t))$, is the sum of ϵ_j and $p_j(y_j(t))$, under the assumption that ϵ_j is small

$$q_j(y_j(t)) = p_j\left(y_j(t)\right) + \epsilon_j \geq \epsilon_j, \quad j \in J. \tag{3.6}$$

When the link is not congested, $q_j(y_j(t)) = \epsilon_j$ since all packet loss are caused by channel error; $q_j(y_j(t))$ gradually increases otherwise. With this inaccurate price, TCP now adjusts

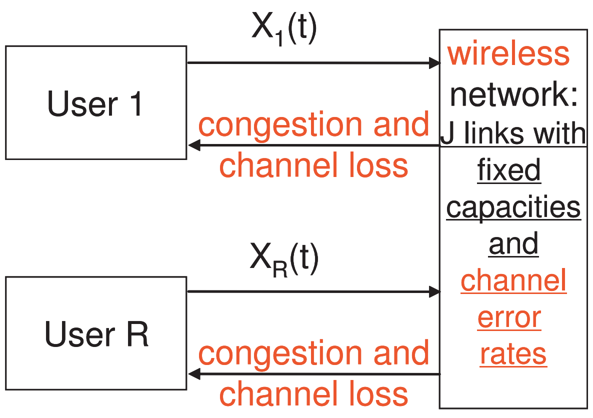

X1(t)
User 1
congestion and channel loss
XR(t)
User R
congestion and channel loss
wireless network: J links with fixed capacities and channel error rates

Figure 3.2: Feedback control modeling: TCP over wireless networks

the sending rates as:

$$\dot{x}_r(t) = \frac{1}{2}\left(\frac{2S^2}{T_r^2} - x_r^2(t)\sum_{j\in r} q_j(y_j(t))\right), \quad r \in R. \tag{3.7}$$

Following a similar analysis in [39], one can show the system in Equations (3.7) and (3.6) has a new unique equilibrium, to which all trajectories converge, as follows:

$$\bar{x}_r = \frac{\sqrt{2}S}{T_r\sqrt{\sum_{j\in r} q_j\left(\bar{y}_j\right)}} \leq \frac{\sqrt{2}S}{T_r\sqrt{\sum_{j\in r}\epsilon_j}}, \quad r \in R, \tag{3.8}$$

where $\bar{y}_j = \sum_{s:j\in s}\bar{x}_s$. Although it can be shown that there is roughly α-fairness among users with $\alpha = 2$, $\bar{x} \triangleq [\bar{x}_r, r \in R]$ is a suboptimal solution as it is different from the unique optimal one x^o. Furthermore, user r could suffer underutilization if $\sum_{j\in r}\epsilon_j$ is sufficiently large. For instance, in the one user one bottleneck network, underutilization happens if and only if $\frac{\sqrt{2}S}{T_r\sqrt{\epsilon}} < C$, where C is the bottleneck bandwidth and ϵ represents the aggregate packet loss rate caused by wireless channel error experienced by the user, as shown in previous chapter. Hence the main problem with TCP over wireless is underutilization of the wireless channel; in fact, similar analysis shows that it is also the main problem with any flow control method that uses packet loss rate as price function, such as TFRC.

One straightforward solution is to provide user r with the accurate price $\sum_{j\in r} p_j(y_j(t))$, and apply it to the control law of $x_r(t)$; this could be done by either end-to-end estimation with or without cross layer information, or by hiding the wireless loss from users via local retransmissions. In fact, most existing approaches belong to this class of solutions, thus requiring modifications either to the transport protocols or to the network infrastructure, making them hard to deploy in practice.

In essence, the main challenge of TCP optimization problem shown in Equation (3.4) for the wireless network setting is to accurately estimate price $\sum_{j \in r} p_j(y_j(t))$ from the noisy measurements $\sum_{j \in r} q_j(y_j(t))$. One way to overcome this problem is to specifically choose a slightly different utility maximization problem, resulting in a new solution that requires measurements that are easy to obtain in a practical networking setup. In the next section, we will show that by selecting a different utility to maximize, there exists a new, stable and optimal solution, which requires only one bit of end-to-end measurement.

3.3 Proposed solution

3.3.1 A New Class of Solutions

Motivated by insights provided by previous discussion in Section 2.2, we now propose a new approach to flow control based on adjusting the number of connections for user r, denoted by $n_r(t)$. This is an end-to-end application layer based scheme, and requires no modification to the network infrastructure or the transport protocol stack.

In Section 2.2, multiple TFRC connections are used to transmit one video stream. Sending rate of individual connections is controlled by TFRC itself. We have argued that the number of connections $n_r(t)$ should be controlled to pursue an optimal value, and one strategy would be to increase $n_r(t)$ until congestion is observed. The NS-2 simulations and actual experiments in Section 2.2 show the performance of such an approach. Motivated by this insight, our proposed approach is to dynamically adjust both $x_r(t)$ and $n_r(t)$.

In our approach, user r's rate $x_r(t)$ is the aggregate rate of $n_r(t)$ TCP connections user r opens. The dynamics of $x_r(t)$ is similar to that of an individual TCP, except it is more aggressive upon no packet loss and less conservative upon packet loss. Specifically, when there is no packet loss, $x_r(t)$ increases by $n_r(t)S/T_r$ per round trip time T_r rather than by S/T_r per T_r as is the case with a single TCP. When a packet loss is observed, $x_r(t)$ decreases by a factor of $1/2n_r(t)$ rather than by $1/2$ as in a single TCP connection case. Note the number of lost packets user r observes during T_r is $\frac{1}{S}x_r(t)T_r \sum_{j \in r} q_j(y_j(t))$ when packet loss rate $\sum_{j \in r} q_j(y_j(t))$ is small; hence, $x_r(t)$ changes approximately $\frac{Sn_r(t)}{T_r}$ −

$\frac{x_r(t)}{2n_r(t)S} x_r(t) T_r \sum_{j \in r} q_j(y_j(t))$ in a round trip time T_r. As a result, the control law of $x_r(t)$ can be expressed as follows:

$$\dot{x}_r(t) = \frac{1}{2Sn_r(t)} \left(\frac{2S^2 n_r^2(t)}{T_r^2} - x_r^2(t) \sum_{j \in r} q_j(y_j(t)) \right), r \in R \tag{3.9}$$

The basic idea of controlling $n_r(t)$ is to increase it if there is no congestion, and decrease it otherwise. In our approach, we apply Inverse Increase and Multiplicative Decrease (IIMD) for $n_r(t)$, i.e. $n_r(t)$ is increased inversely when there is no congestion, and decreases multiplicatively otherwise. The explicit control law is shown below:

$$\dot{n}_r(t) = c_r \left(\frac{1}{n_r(t)} - n_r(t) I_r[\sum_{j \in r} p_j(y_j(t))] \right), \qquad r \in R \tag{3.10}$$

where $c_r, r \in R$ are nonnegative constants indicating how fast $n_r(t), r \in R$ are adjusted, and $I_r(\sum_{j \in r} p_j(y_j(t)))$ is an indicator function implying the congestion status of route r:

$$I_r[\sum_{j \in r} p_j(y_j(t))] = I \left(\sum_{j \in r} \frac{(y_j(t) - C_j)^+}{y_j(t)} \right) \tag{3.11}$$

$$= \begin{cases} 1, & \text{if route } r \text{ is congested at time } t, \\ & \text{i.e. } \sum_{j \in r} \frac{(y_j(t) - C_j)^+}{y_j(t)} > 0; \\ 0, & \text{otherwise.} \end{cases}$$

As we will see later, this law leads the system to a stable equilibrium that meets all of our design goals.

As shown in Equation (3.9) and explained above, the control law of the aggregate rate for user r, i.e. $x_r(t)$, can be understood as the sum of rates from $n_r(t)$ individual connections, with each connection being controlled using the standard TCP Reno algorithm. Therefore, our approach does not require any modifications to the TCP protocol. On the other hand, as seen in Equation (3.10), the system tries to achieve full utilization by adjusting the number of connections $n_r(t)$ accordingly. In particular,

- If a route r is underutilized, then $I_r[\sum_{j \in r} p_j(y_j(t))] = 0$; this implies that the number of connections $n_r(t)$ will increase in order to boost the user's rate $x_r(t)$, pursuing full utilization on any route r;

- If the route r is fully utilized, i.e. one of its links is congested, then $I_r[\sum_{j \in r} p_j(y_j(t))] = 1$, lowering $n_r(t)$, and hence $x_r(t)$, to prevent the system from further congestion.

The intuition behind our approach is as follows: when loss rate caused by channel error increases, individual connection's sending rate is lowered, thus users need to open more connections to increase the aggregate throughput. The $I_r(\cdot)$ is the one bit of information required from the end-to-end measurements. In practice, lots of techniques can be applied to estimate $I_r(\cdot)$ using end-to-end statistics [7–10, 12, 13, 20–24, 26]. In particular, we estimate the queuing delay by comparing current round trip time with the propagation delay, and set $I_r(\cdot) = 1$ if the queuing delay is positive, and $I_r(\cdot) = 0$ otherwise.

The system in Equations (3.9) and (3.10) is a coupled, discontinuous, nonlinear system. In order to verify that this system actually meets our design goals, we need to analyze the existence of a unique equilibria, its stability and its optimality in the sense of solving a utility maximization problem.

3.3.2 Discontinuity Approximation and The Two Timescale Decomposition

The discontinuities of functions $I_r[\sum_{j\in r} p_j(y_j(t))]$ and $p_j(y_j(t))$ hinder the analysis of the equilibria. To carry out the analysis, we first approximate these discontinuous functions using continuous functions, in order to generate an approximate continuous system to the original discontinuous system[3]: $\forall j \in J, r \in R$,

$$p_j(y_j(t)) \approx \frac{1}{\beta} \ln \left(1 + e^{\beta \frac{y_j(t) - C_j}{y_j(t)}} \right) \triangleq g_j(y_j(t)), \tag{3.12}$$

$$I_r[\sum_{j\in r} p_j(y_j(t))] \approx \frac{e^{\beta \sum_{j\in r} g_j(y_j(t))} - 1}{e^{\beta \sum_{j\in r} g_j(y_j(t))} + 1} \triangleq f(\sum_{j\in r} g_j(y_j(t)), \tag{3.13}$$

where β is a nonnegative constant. It should be clear that $f(\sum_{j\in r} g_j(y_j(t)) \to I_r(\sum_{j\in r} g_j(y_j(t))$ and $g_j(y_j(t)) \to p_j(y_j(t))$ as $\beta \to \infty$.

Thus, an approximate continuous version of the original system in Equations (3.9) and (3.10) is given by: $\forall r \in R$,

$$\begin{cases} \dot{x}_r(t) = \frac{1}{2Sn_r(t)} \left(\frac{2S^2 n_r^2(t)}{T_r^2} - x_r^2(t) \sum_{j\in r} [\epsilon_j + g_j(y_j(t))] \right), \\ \dot{n}_r(t) = c_r \left(\frac{1}{n_r(t)} - n_r(t) f(\sum_{j\in r} g_j(y_j(t))) \right). \end{cases} \tag{3.14}$$

[3]We will discuss the relationship between the approximate system and the original system, and performance of the actual implementation later.

Since the approximate system in Equation (3.14) is continuous, we can analyze its equilibrium and stability for arbitrary values of β. As $\beta \to \infty$, the system in Equation (3.14) approaches the original system in Equations (3.9) and (3.10). Therefore, the equilibria and the stability results for the approximate system also correspond to the results for the original system in the Filippov sense [49, 50].

The approximate system in Equation (3.14), though continuous, is difficult to analyze in general. It is a nonlinear, coupled, multivariate system, and the two equations are not exactly symmetric even though they might appear to be so. Hence, we introduce a two time scale assumption to analyze the approximate system: *The number of connections, $n_r(t)$, changes in a timescale much slower than the source rate, $x_r(t)$.* This assumption is the key not only for carrying out the equilibria and stability analysis, but also for extending the results and opening the door to a general two time scale framework for flow control, as we will see later. It is also reasonable in practice, where the sending rates are expected to change on the order of tens of milliseconds, while number of connections is expected to change at a much slower rate, e.g. tens of seconds.

Under the above assumption, system in Equation (3.14) fits into the classical singular perturbation framework [49] [50], and therefore can be decoupled into a fast timescale and a slow timescale system. The fast timescale system is described by Equation (3.9) with the corresponding $n_r(t), r \in R$ being constant, namely boundary system: $\forall r \in R$,

$$\begin{cases} \dot{x}_r(t) = \frac{1}{2Sn_r(t)}\left(\frac{2S^2n_r^2(t)}{T_r^2} - x_r^2(t)\sum_{j\in r}[\epsilon_j + g_j(y_j(t))]\right), \\ n_r(t) = \text{constant}. \end{cases} \tag{3.15}$$

The slow timescale system is described by Equation (3.10) along the equilibrium manifold defined by the stationary solution of Equation (3.9), namely reduced system: $\forall r \in R$,

$$\begin{cases} x_r(t) = \frac{n_r(t)\sqrt{2}S}{T_r\sqrt{\sum_{j\in r}[\epsilon_j + g_j(y_j(t))]}}, \\ \dot{n}_r(t) = c_r\left(\frac{1}{n_r(t)} - n_r(t)f(\sum_{j\in r} g_j(y_j(t)))\right). \end{cases} \tag{3.16}$$

Under the two timescale setting, the behavior of the system can be described as follows. On the fast timescale, $n(t)$ can be thought of as being constant since its dynamics happens at a slow timescale. The entire system can then be expressed as a boundary system shown in Equation (3.15), and only dynamics of $x(t)$ is considered. As the boundary system is similar to Kelly et. al.'s control system on wired networks [39] except for the constant $n(t)$, and the price function $p_j(y_j(t))$ being replaced by $\epsilon_j + g_j(y(t)$, the behavior of the boundary system can be easily inferred from the known results for the system in Equation (3.3). Specifically, at the fast timescale, $x(t) \triangleq [x_r(t), r \in R]$ globally exponentially converges to the equilibrium manifold defined as follows [39, 41]:

$$x_r(t) = \frac{n_r(t)\sqrt{2}S}{T_r\sqrt{\sum_{j\in r}\left[g_j\left(y_j(t)\right) + \epsilon_j\right]}}, \quad r \in R. \tag{3.17}$$

This can be easily shown be to a one-to-one mapping between $x(t)$ and $n(t) \triangleq [n_r(t), r \in R]$:

Lemma 3.3.1. *The equilibrium manifold defined by Equation (3.17), is a one-to-one mapping between $x(t)$ and $n(t)$.*

Proof. On one hand, it is easy to see directly from Equation (3.17) that one set of $x(t)$ results in one set of $n(t)$. On the other hand, given a set of $n(t)$, Equation (3.17) is the solution solving the following strictly concave optimization problem:

$$\max_x \left\{ -\sum_{r\in R} \frac{2\,n_r^2(t)\,S^2}{T_r^2 x_r} - \sum_{j\in J}\int_0^{\sum_{s:j\in s} x_s} \left[\epsilon_j + g_j\left(y_j(t)\right)\right]\,dz \right\}, \tag{3.18}$$

and the solution must be unique. Therefore, one set of $n(t)$ only results in one set of $x(t)$, and the mapping is one-to-one. □

On the slow timescale, $x(t)$ has already converged to the above equilibrium manifold, and now the system collapses into the reduced system described in Equation (3.16). $x(t)$ has already converged onto the equilibrium manifold, and only dynamics of $n(t)$ are explicitly considered, whose behavior determines how the approximate system evolves at the slow timescale. Therefore, together with boundary system, it fully characterizes behavior of the system for all possible timescales. For illustration purposes, we have handdrawn the time dynamics of singular perturbation system comprising of a single user with sending rate $x_1(t)$, and $n_1(t)$ connections in Figure 3.3. As shown, given any initial condition, the system first converges rapidly onto the equilibrium manifold, representing the fast timescale indicated by its boundary system. On the manifold, the system's behavior follows its reduced system; if the reduced system has the equilibrium as the globally asymptotically stable equilibrium, then the trajectories, along the manifold, will converge to the equilibrium as time goes to infinity.

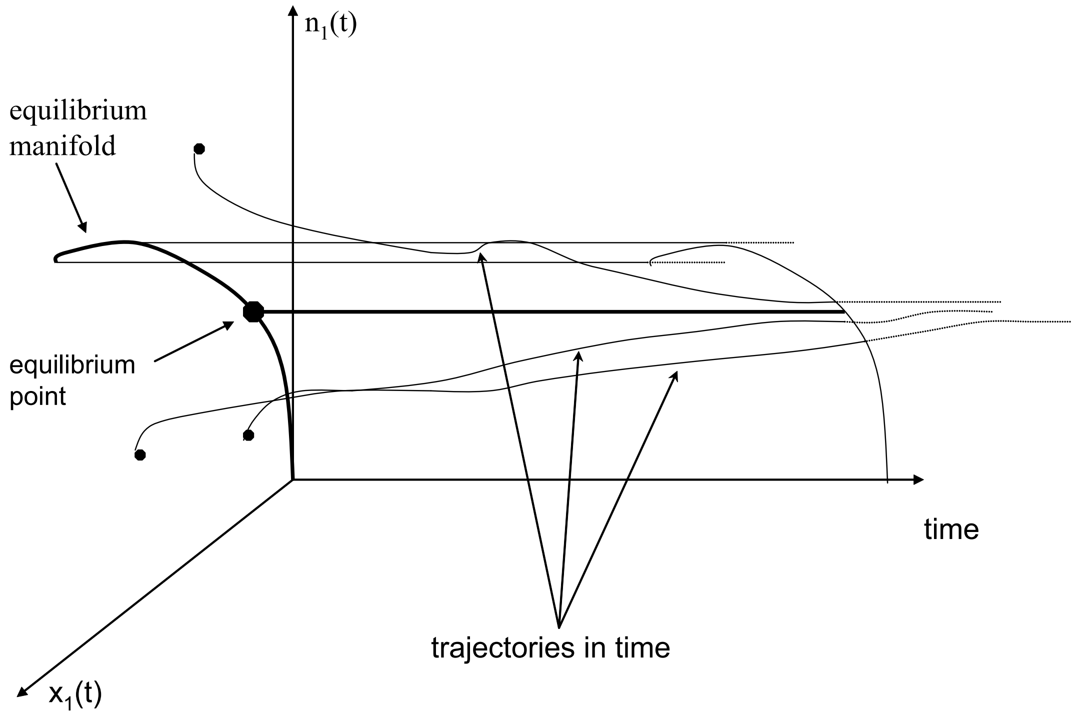


Figure 3.3: Time dynamics of a singular perturbation system.

In the next subsection, we present the results on the existence of a unique optimal equilibrium of the system and its stability.

3.3.3 The Existence of A Unique Optimal Equilibrium and Its Stability

Given the system in Equation (3.14), the first question to answer is whether it has any equilibria, and if so how many. The second question is the local and global stability of these equilibria. These two questions are important in the sense that they describe the behavior of the system as time evolves, predicting the system's performance in actual implementations in practice. For instance, if the system in Equation (3.14) has no equilibrium, the users' sending rates would not converge, making the system undesirable to implement.

First, we define four notions of stability to be used in the following theorems and lemmas, as follows:

Definition 3.3.1. ***Local Stability:*** *an equilibrium ξ^* of a continuous system $\dot{\xi} = \varphi(\xi, t, \xi_0)$ is locally stable, if $\exists\delta$, such that all system trajectories starting from any ξ_0, satisfying $|\xi_0 - \xi^*| < \delta$, converge uniformly and exponentially to ξ^*.*

Definition 3.3.2. ***Global Exponential Stability:*** *an equilibrium ξ^* of a continuous system $\dot{\xi} = \varphi(\xi, t, \xi_0)$ is globally exponentially stable, if all system trajectories starting from any ξ_0, converge uniformly and exponentially to ξ^*.*

Definition 3.3.3. ***Semi Global Exponential Stability:*** *an equilibrium ξ^* of a continuous system $\dot{\xi} = \varphi(\xi, t, \xi_0)$ is semi globally exponentially stable, if all system trajectories, starting from any ξ_0, converge exponentially but not uniformly to ξ^*. That is, the convergence rate depends on the initial condition ξ_0.*

Definition 3.3.4. ***Global Asymptotical Stability:*** *an equilibrium ξ^* of a continuous system $\dot{\xi} = \varphi(\xi, t, \xi_0)$ is globally asymptotically stable, if all system trajectories, starting from any ξ_0, asymptotically converge to ξ^*.*

We now show that the system in Equation (3.14) has only one unique equilibrium, which is locally exponentially stable, and semi globally exponentially stable, i.e. it is exponentially stable as long as $n(t)$ and $x(t)$ are constrained to lie in a compact set[4]. Further, the unique equilibrium solves a concave optimization problem. We show this through the following theorem.

Theorem 3.3.1. *For arbitrary $\beta > 0$, the approximate system in Equation (3.14) has a unique equilibrium, denoted by (x^*, n^*), given by*

$$\begin{aligned} n_r^* &= \frac{1}{\sqrt{f(\sum_{j\in r} g_j(y_j^*))}}, \qquad r \in R; \\ x_r^* &= \frac{\sqrt{2}S}{\sqrt{f(\sum_{j\in r} g_j(y_j^*))}T_r\sqrt{\sum_{j\in r}\left[g_j(y_j^*)+\epsilon_j\right]}}, \qquad r \in R. \end{aligned} \tag{3.19}$$

[4]Note $n(t)$ and $x(t)$ are typically constrained in practice.

Further, this unique equilibrium solves the following concave optimization problem

$$\max_{x \geq 0} \sum_{r \in R} U_r(x_r) - \sum_{j \in J} \int_0^{y_j} g_j(z)\, dz, \tag{3.20}$$

with U_r being concave function:

$$U_r(x_r) = \int_0^{x_r} h_r^{-1}\left(\frac{2S^2}{T_r^2 \nu^2}\right) d\nu, \quad r \in R,$$

where h_r^{-1} is the inverse of a monotonically increasing function h_r:

$$h_r(z) \triangleq \left(\sum_{j \in r} \epsilon_j + z\right) f(z) = \left(\sum_{j \in r} \epsilon_j + z\right) \frac{e^{\beta z} - 1}{e^{\beta z} + 1}, \quad r \in R.$$

Proof. Refer to Appendix A. □

Two observations to be made from Theorem 3.3.1 are the following. First, for large β, bottleneck links for every user are almost fully utilized at the equilibrium shown in Equation (3.19). This is because at the equilibrium, $\dot{n}_r^* = 0, r \in R$; hence, $f(\sum_{j \in r} g_j(y_j^*)) = 1/(n_r^*)^2 > 0, r \in R$, from Equation (3.14). This implies that at the equilibrium $f(\sum_{j \in r} g_j(y_j^*))$ is strictly positive. Taking into account Equation (3.13) for large β, $f(\sum_{j \in r} g_j(y_j^*)) > 0$ indicates $\sum_{j \in r} g_j(y_j^*) > 0$, i.e. the packet loss rate caused by congestion is also strictly positive. This implies that at least at one link j along user r's path, the aggregate rate y_j^* passing through link j is close to or lager than the link capacity C_j, hence bottleneck being almost fully utilized.

Second, the unique equilibrium for the system in Equation (3.14) in wireless scenario solves a concave optimization problem similar to the one TCP Reno solves in the wired network. They have the same form as Equation (3.1) but with different users' utility functions $U_r(x_r)$. The only difference is that the $U_r(x_r)$ in the wired case is only a function of x_r, while in wireless scenario it is also a function of $\sum_{j \in r} \epsilon_j$, i.e. the packet loss rate associated with the route r. In fact, if we let $\beta \to \infty$ and $\epsilon_j = 0, \forall j \in J$, i.e. in the wired network scenario, we have $h_r(z) = z$, and the optimization problem in Equation (3.20) becomes identical to the TCP optimization problem in Equation (3.4). In this case, the equilibrium (x^*, n^*) is exactly the same as x^o, implying TCP optimization problem in the wired network is merely a special case of that in Equation (3.20).

Given the system in Equation (3.14) has a unique optimal equilibrium, an important question to answer is that whether it is stable, i.e. will the users' rates converge to it. The following two theorems explore the answer to this question.

Theorem 3.3.2. *For arbitrary $\beta > 0$, under the two timescale assumption, the unique equilibrium of the reduced system in Equation (3.16) is locally exponentially stable. Also, the unique equilibrium of the approximate system in Equation (3.14), (x^*, n^*), is locally exponentially stable.*

Proof. Refer to Appendix B. □

Theorem 3.3.2 implies that if the number of connections $n(t)$ is initially in a small ball around the equilibrium n^*, then the entire system will converge exponentially fast to the equilibrium. As no convergence can be faster than exponential, this is the best result one can expect in a local region around the equilibrium.

How about when $n(t)$ starts far away from the equilibrium n^*? To answer this question, we explore the global stability of the equilibrium.

We state three lemmas that are needed for proving the global exponential stability. The first lemma explores an interesting structure of the vector field of the boundary layer system and the reduced system.

Lemma 3.3.2. *There exists a compact set, denoted by Ω_1, for $n(t)$ in the reduced system in Equation (3.16) with arbitrary $\beta > 0$, such that any compact set containing it is a positively invariant one. The same observation is also true for $x(t)$ in the boundary layer system Equation (3.15), and the corresponding compact set is defined as $\Omega_2(n)$, a function of n.*

Proof. Refer to Appendix C. □

A positive invariant set is a set with all trajectories on its boundary pointing inwards; as such, no trajectories inside the set will ever move out.

The next lemma investigates the global asymptotical stability of the equilibrium in the reduced system. The particular non-linear shape of the vector field for $\dot{n}_r(t)$ shown in Equation (3.16) makes the search for a suitable Lyapunov function, or a function on which to apply the La Salle principle, a challenging task. We have found that none of the techniques applied in [30] and [43] work in this case. We therefore believe that our functions for applying the La Salle principle, in the proof of following lemma, may provide new insight to searching Lyapunov functions, or functions for the La Salle principle, in similar or general cases.

Lemma 3.3.3. *The unique equilibrium of reduced layer system in Equation (3.16), with arbitrary $\beta > 0$, is a globally asymptotically stable one.*

Proof. Refer to Appendix D. □

The third lemma states that for continuous systems, local exponential stability and global asymptotical stability is equivalent to semi-global exponential stability. We should not here that this lemma is quite general and as such, its use is not restricted to the use in particular problem discussed in this thesis.

Lemma 3.3.4. *Consider a system $\dot{\xi} = \varphi(\xi, t, \xi_0)$ satisfying the following assumptions:*

- *it has a unique equilibrium at 0 that is locally exponentially stable and globally asymptotically stable;*
- *$\varphi(\xi, t, \xi_0)$ is continuous.*

Then the equilibrium of the system is semi-globally exponentially stable.

Proof. Refer to Appendix E. □

In Lemma 3.3.4, semi-global exponential stability of an equilibrium of the system implies starting from arbitrary initial point, the trajectory of the system will always converge to the equilibrium exponentially fast. However the convergence rate depends on the initial point, i.e. the convergence is not uniform. Clearly, semi-global exponential stability is a much stronger than local exponential stability, but it less strong than global exponential stability in the sense that the semi-globally exponential convergence is not a uniform one.

These three lemmas enable us to assert the semi-globally exponential stability of the equilibrium of the system in Equation (3.14), as follows:

Theorem 3.3.3. *The unique equilibrium of singularly perturbed system in Equation (3.14) with arbitrary $\beta > 0$ is semi-globally exponentially stable.*

Proof. Refer to Appendix F. □

From practical point of view, the above theorem implies that for any number of users in a network with any initial sending rates, users' rates will converge to a unique equilibrium exponentially fast.

The intuition behind both the local and semi-global convergence of the entire system is as follows: $x_r(t)$ first converges in the fast timescale to the equilibrium of the boundary system in Equation (3.15), defined by the equilibrium manifold as in Equation (3.17); then in a slow timescale , $n_r(t)$ and $x_r(t)$ follow the control laws of the reduced system in Equation (3.16) to converge to the optimal equilibrium along the manifold. An important consequence of this convergence argument is that, a combination of control law in Equation (3.10) on $n_r(t)$ and any flow control method on $x_r(t)$ resulting in the same equilibrium manifold as TCP, will retain the convergence behavior shown in Theorems 3.3.2 and 3.3.3. Therefore, it is possible to extend all these results to TFRC since it has been shown in [2] that TFRC has the same stationary behavior as TCP.

In summary, we have shown in Theorems 3.3.1, 3.3.2 and 3.3.3 that:

- For arbitrary topology, arbitrary number of users, and arbitrary initial sending rates, the sending rates controlled by proposed solution with arbitrary β converge to a unique equilibrium exponentially fast.

- For large β, all bottlenecks are fully utilized at the equilibrium; users' rates are allocated according to Equation 3.19.

- At the equilibrium, a net utility shown in Equation 3.20 is maximized.

We also know the proposed solution can be implemented based on local information only, and is distributed. Therefore, we have achieved all of our desired flow control goals.

Theorems 3.3.1, 3.3.2 and 3.3.3 state the existence of a unique optimal equilibrium, and ensure its stability for the continuous approximate system in Equation (3.14), for arbitrary values of β. In the limit as $\beta \to \infty$, the approximate system approaches the original discontinuous system in Equations (3.9) and (3.10). Therefore, for extremely large β, we expect the approximate system to behave quite similarly to the original system, except at the discontinuities $y_j(t) = C_j$.

As seen in the next section, in actual implementation of the proposed system in Equations (3.9) and (3.10), it is necessary to discretize continuous quantities. For instance, controlling $n_r(t)$ is implemented by adjusting the number of connections, which has to be an integer number; controlling $x_r(t)$ is implemented by TCP to adjust the finite number of packets to be sent out in a time interval. Therefore, it is highly unlikely for the system to operate at discontinuous points. From this point of view, the analysis based on the approximate system is accurate enough to predict and interpret the performance of the actual implementation of the algorithm in practice.

3.4 Discussions on The Proposed Solution

In the proposed solution, c_r can be chosen in a distributed fashion in the system shown in Equation (3.14), as long as the two timescale assumption holds. Practically, this implies that each user can adjust $n_r(t)$ according to a different rate. Specifically, a global setting among all the users is not necessary. Furthermore, allowing some of the c_r to be zero represents a scenario according to which the proposed scheme coexists with TCP. In this situation, all theorems still hold, except for a modification to Theorem 3.3.1. More precisely, we have the following Corollary:

Corollary 3.4.1. *For arbitrary topology, arbitrary number of users running either TCP or proposed solution, and arbitrary initial sending rates, the following holds:*

- *Sending rates $x(t)$ converge to a unique equilibrium:*
  - *Users running TCP:*

$$x_r^* = \frac{\sqrt{2}S}{T_r\sqrt{\sum_{j\in r}\left[g_j\left(y_j^*\right)+\epsilon_j\right]}} \tag{3.21}$$

  - *Users running proposed solution as in Equation (3.14):*

$$x_r^* = \frac{\sqrt{2}S}{\sqrt{f(\sum_{j\in r} g_j(y_j^*))}T_r\sqrt{\sum_{j\in r}\left[g_j\left(y_j^*\right)+\epsilon_j\right]}} \tag{3.22}$$

- *At the equilibrium, all bottlenecks are fully utilized for large β, and the following concave optimization problem is solved:*

$$\max_{x\geq 0}\sum_{r\in R}U_r(x_r)-\sum_{j\in J}\int_0^{y_j} g_j(z)\,dz, \tag{3.23}$$

  with U_r being concave function:

  - *Users running TCP:*

$$U_r(x_r) = -\sum_{r\in R}\frac{2S^2}{T_r^2 x_r} - x_r\sum_{j\in r}\epsilon_j,$$

  - *Users running proposed solution as in Equation (3.14):*

$$U_r(x_r) = \int_0^{x_r} h_r^{-1}\left(\frac{2S^2}{T_r^2\nu^2}\right)d\nu,$$

where h_r^{-1} is the inverse of an monotonically increasing function h_r:

$$h_r(z) \triangleq \left(\sum_{j\in r}\epsilon_j+z\right)f(z) = \left(\sum_{j\in r}\epsilon_j+z\right)\frac{e^{\beta z}-1}{e^{\beta z}+1}, \quad r\in R.$$

An observation on the fairness among users is that at the equilibrium, users running proposed solution are fair to users running TCP in the following sense:

- In the case that TCP operates in capacity-limited region, i.e. all bottleneck links are fully utilized even if all users running proposed solution only open one connection, for large values of β, the approximated indicator function $f(\cdot)$ takes the value 1. Users running proposed solution will open only one connection, and their sending rates will converge to the same value as if they were running TCP. In this case, fairness among users running TCP and proposed solution is the same as fairness among TCP users.

- In the case that TCP operates in channel-error-limited region, i.e. the bottleneck links of some routers are underutilized, if users running proposed solution open one connection, for large values of β, approximated indication function $f(\cdot)$ takes on a value between 0 and 1, and approximated congestion loss function $g(\cdot)$ takes on a very small positive value, as seen from definition of $f(\cdot)$ in Equation 3.13. Consequently, as seen from users' equilibrium rates in the above Corollary, users running proposed solution will get the residual bandwidth without affecting TCP users's throughput, i.e. users running TCP get throughput close to what they would have gotten if no other users were running the proposed solution.

Above results imply that in a network where our schemes coexist with TCP, all users' rates will again converge, semi-globally exponentially, to a unique optimal and optimistically fair equilibrium. Note this claim is true for arbitrary topology, arbitrary number of users, and arbitrary initial sending rates. These observations on stability and scalability encourage incremental deployment of our scheme in the current Internet where TCP is dominant, and addresses a major concern. The simulations in Chapter 5 partially support this observation.

Since all analysis and results hold regardless of the values of $\epsilon_j, j \in J$, it implies that our proposed solution works in both wired and wireless scenarios. In the wired scenario, because the indicator function $I(\cdot)$ takes value 1, proposed solution ends up with opening one connection and reduces to traditional TCP. Hence, we can afford to design only one practical scheme, following Equations (3.9) and (3.10), for both wireless and wired networks, with the latter corresponding to the case where $\epsilon_j = 0, j \in J$.

Table 3.1: MATLAB Simulation setting.

Simulation setting	Value
Capacity of link 1: C_1	100 Bps
Capacity of link 2: C_2	160 Bps
Wireless packet loss rate on link 1: p_w^1	0.001
Wireless packet loss rate on link 2: p_w^2	0.0005
Packet size: S	1000 Bytes
Round trip time of user 1: T_1	2 ms
Round trip time of user 2: T_2	1 ms
Round trip time of user 3: T_3	3 ms
Adjusting rate of the number of connections of user 1: c_1	0.005
Adjusting rate of the number of connections of user 2: c_2	0.01
Adjusting rate of the number of connections of user 3: c_3	0.015
Simulation time interval	20 ms

3.5 Illustrated MATLAB Simulations

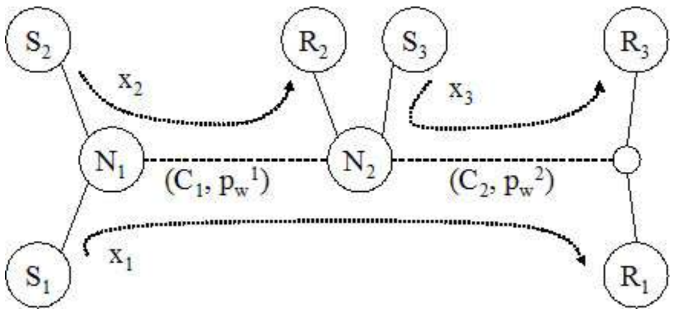
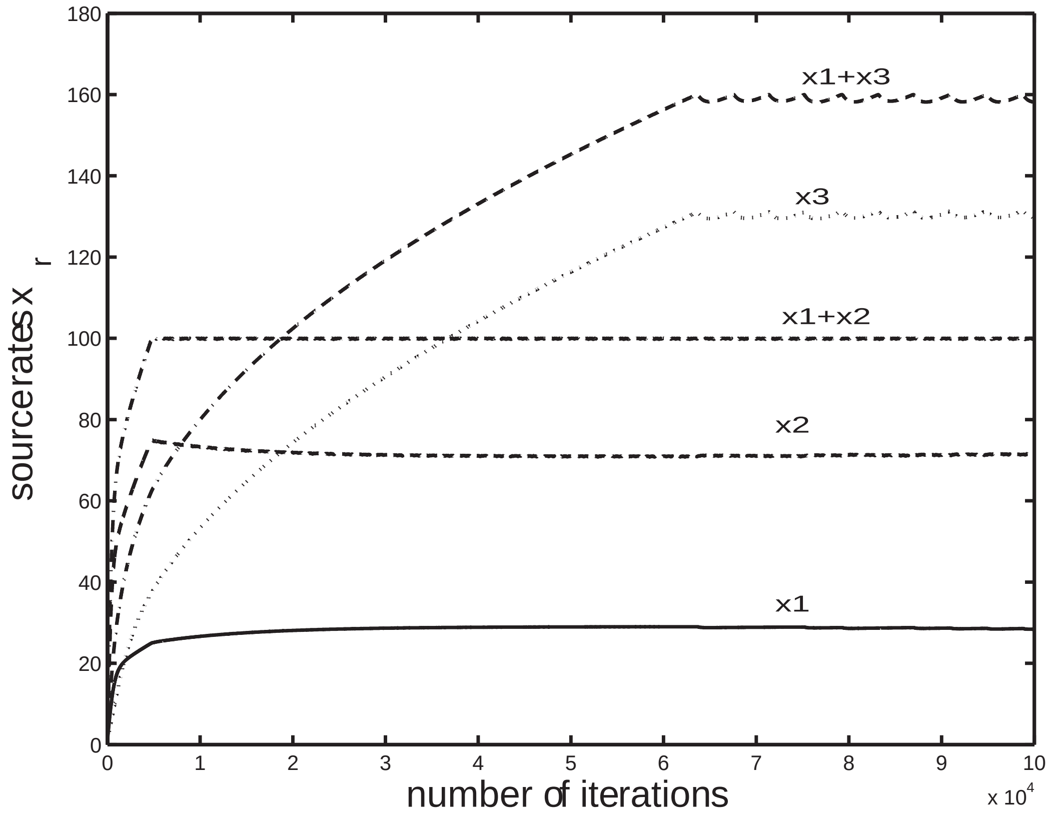


Figure 3.4: MATLAB Simulation topology.

To visually illustrate the system dynamics described in previous sections, we carry out MATLAB simulations for the topology shown in Figure 3.4 with settings shown in Table 3.1. As seen, c_1, c_2, c_3 are chosen to be very small, in order to satisfy the two timescale assumptions.

In simulations, we control sending rates for user i along path s_i to r_i, $i = 1, 2, 3$, using proposed scheme in Equations (3.9) and (3.10). The simulation results are shown in Figure 3.5, including sending rates of users and the number of connections. As predicted in the analysis in Section 3.3, proposed solution achieves full utilization of links, and users' sending rates converge nicely. Around full utilization of links, there are some oscillations of



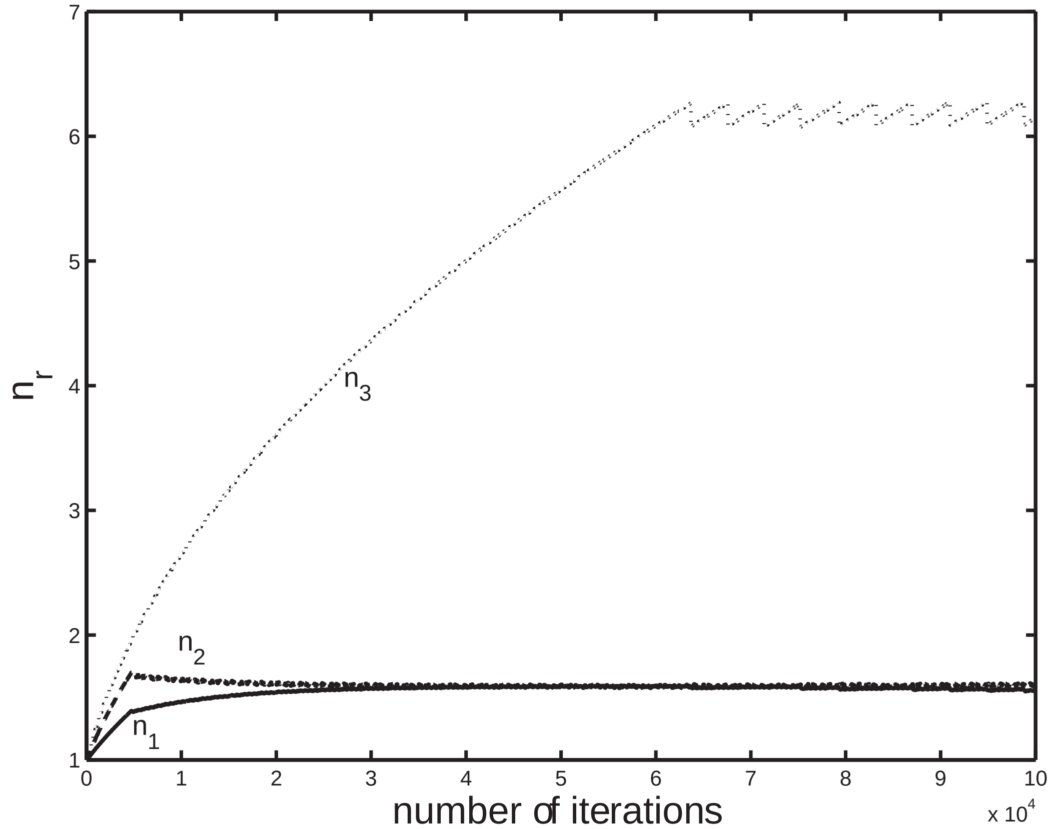


Figure 3.5: Illustrated MATLAB simulation results: sending rates and the number of connections.

aggregate rates, e.g. aggregate rate passing through link 2 is $x_1(t) + x_3(t)$. This oscillation is a direct consequence of the gradual increase and sharp decrease of the number of connections, according to IIMD control law shown in Equation (3.10). Although only results of one instance of simulation are shown, the same convergence behavior of sending rates and the number of connections appears in all other simulations with different initial conditions on sending rates and the number of connections.

3.6 A General Two Timescale Framework for Flow Control

In proposed solution, in order to converge to a desired equilibrium, two control laws for both number of connections and sending rate of individual connection are designed. Sending rate of individual connection is controlled by TCP to converge to an equilibrium manifold exponentially fast, on a fast timescale. Number of TCP connections is controlled to converge to the desired equilibrium exponentially fast along the manifold, on a slow timescale. We have also argued that TFRC can be applied to replace TCP to control sending rate of individual connection, and the entire system still converges to the desired equilibrium exponentially fast.

This implies a general two timescale framework as indicated from our proposed solution. In this framework, it is sufficient to solve any flow control problem by the following process:

- Based on the desired flow control goals, choose an equilibrium of sending rates that satisfy all specific goals, e.g. full utilization of bottleneck bandwidth and fairness among users.
- On a fast timescale, fix number of connections, design a control law for sending rate of individual connections, such as the one shown in Equation (3.7), to converge to a equilibrium manifold containing the desired equilibrium exponentially fast.
- On a slow timescale, design a control law for number of connections, such as the one shown in Equation (3.10) to converge to the desired equilibrium exponentially fast, along the equilibrium manifold.

If we follow the above process, then users' sending rates will converge to the desired equilibrium exponentially fast. Theoretically, this framework is an application of the singular perturbation theorem in [50]. Since no convergence faster than exponential is possible, this is in fact the best convergence shape one can expect.

Our proposed solution for problem of flow control in wireless networks, follows this general framework implicitly. By theorems and lemmas in this chapter, the proposed solution chooses an optimization problem to solve, of which the optimal equilibrium achieves all flow control goals. On the fast timescale, the proposed solution applies TCP to control sending rate of individual connection to converge to an equilibrium manifold containing the desired equilibrium. On the slow timescale, the proposed solution applies IIMD control law to control the number of TCP connections, in order to converge to the desired equilibrium along the equilibrium manifold. Eventually, the sending rates controlled by the proposed solution converge to the desired equilibrium exponentially fast, as expected.

By always using TCP as the control law for sending rate in fast timescale, and designing new control law for number of connections in slow timescale, one can potentially address any flow control problem without changing today's transport layer protocol. Further, if the control law for the number of connections utilizes information that today's infrastructure can readily provide, then the problem is solved without modifying network infrastructure as well.

One significant advantage of this two timescale framework is the separation of control laws in two timescales. Control law in each timescale can be designed *independently* of the control law in the other timescale. As long as the control law can guarantee exponential convergence in its timescale, the general exponential convergence result holds. This implies that we can replace one control law in one timescale, without affecting the one in the other timescale, and the general convergence result still holds. For example, as argued in previous chapter, we can use TFRC rather than TCP to control sending rate of individual connections, and still achieve all convergence properties. This is analogous to upgrading a car by changing the tires without changing the engine.

3.7 A Variant to Proposed Solution

In this section, we describe a variant to proposed solution, by using a different control law for the number of connections in a slow timescale. This flexible design shows the power of the general two timescale framework.

The variant solution uses TCP to control sending rate of individual connection, and uses Inversely Increase and Additively Decrease (IIAD) to control the number of connections. That is, the number of connections increases inverse proportionally upon no congestion, and decreases additively upon congestion. The only difference between this variant solution and proposed solution is the control law for the number of connections, i.e. IIAD vs IIMD. The approximated version of the variant solution is the following:

$$\begin{cases} \dot{x}_r(t) = \frac{1}{2Sn_r(t)}\left(\frac{2S^2n_r^2(t)}{T_r^2} - x_r^2(t)\sum_{j\in r}[\epsilon_j + g_j(y_j(t))]\right), \\ \dot{n}_r(t) = c_r\left(\frac{1}{n_r(t)} - f(\sum_{j\in r} g_j(y_j(t)))\right). \end{cases} \tag{3.24}$$

Applying two timescale decomposition, we derive the boundary layer system in fast timescale:

$$\begin{cases} \dot{x}_r(t) = \frac{1}{2Sn_r(t)}\left(\frac{2S^2n_r^2(t)}{T_r^2} - x_r^2(t)\sum_{j\in r}[\epsilon_j + g_j(y_j(t))]\right), \\ n_r(t) = \text{constant}. \end{cases} \tag{3.25}$$

and the reduced order system in slow timescale:

$$\begin{cases} x_r(t) = \frac{n_r(t)\sqrt{2}S}{T_r\sqrt{\sum_{j\in r}[\epsilon_j+g_j(y_j(t))]}}, \\ \dot{n}_r(t) = c_r\left(\frac{1}{n_r(t)} - f(\sum_{j\in r} g_j(y_j(t)))\right). \end{cases} \tag{3.26}$$

Again, the question here is whether the variant solution leads to a system with a unique exponential stable equilibrium, at which all flow control goals are achieved. Following similar analysis of proposed solution, we have the following theorem for existence and uniqueness of the optimal equilibrium:

Theorem 3.7.1. *For arbitrary $\beta > 0$, the approximate system in Equation (3.24) has a unique equilibrium, denoted by (x^*, n^*) as*

$$\begin{aligned} n_r^* &= \frac{1}{f(\sum_{j\in r} g_j(y_j^*))}, \qquad r \in R; \\ x_r^* &= \frac{\sqrt{2}S}{f(\sum_{j\in r} g_j(y_j^*))T_r\sqrt{\sum_{j\in r}\left[g_j(y_j^*)+\epsilon_j\right]}}, \quad r \in R. \end{aligned} \tag{3.27}$$

Further, this unique equilibrium solves the following concave optimization problem

$$\max_{x\geq 0} \sum_{r\in R} U_r(x_r) - \sum_{j\in J}\int_0^{y_j} g_j(z)\,dz, \tag{3.28}$$

with U_r being concave function:

$$U_r(x_r) = \int_0^{x_r} h_r^{-1}\left(\frac{2S^2}{T_r^2\nu^2}\right) d\nu, \quad r \in R, \tag{3.29}$$

where h_r^{-1} is the inverse of a monotonically increasing function h_r:

$$h_r(z) \triangleq \left(\sum_{j\in r}\epsilon_j + z\right) f^2(z) = \left(\sum_{j\in r}\epsilon_j + z\right)\left(\frac{e^{\beta z}-1}{e^{\beta z}+1}\right)^2, \quad r \in R.$$

Proof. Refer to Appendix G. □

At the equilibrium, as compared to proposed solution, all observations about $x_r(t)$ are the same, except a slightly different net utility shown in Equation (3.29) is maximized.

Since only the reduced order system is modified, according to the general two timescale framework, we need to show reduced order system has an exponentially stable equilibrium, in order to show the entire system converges to the desired equilibrium exponentially fast. The following theorem shows the exponential stability of the equilibrium of reduced order system:

Theorem 3.7.2. *For arbitrary $\beta > 0$, the unique equilibrium of the reduced system in Equation (3.26) is*

- *locally exponentially stable,*
- *globally asymptotically stable,*
- *semi globally exponentially stable.*

Proof. For local exponential stability, the proof follows exactly the same idea, technique, and procedure of the proof of Theorem 3.3.2 in Appendix B, with the exception of few equations. So we will not go over the details here.

For globally asymptotically stability, the proof follows exactly the same idea, technique, and procedure of the proof of Lemma 3.3.3 in Appendix D, except for using a different La Salle function in the proof, as follows:

$$\begin{aligned} V(n,z) &= -\sum_{r\in R} c_r \left[\int_{\epsilon_r}^{z_r} \frac{1}{\sqrt{y} n_r} dy - \int_{\epsilon_r}^{z_r} \frac{1}{\sqrt{y}} f(y-\epsilon_r) dy \right. \\ & \left. + \int_0^{n_r} \frac{1}{y^2} \phi_r(\frac{1}{y}) dy \right], \end{aligned} \tag{3.30}$$

where function $\phi_r(\cdot)$ is defined in Equation (D.2). So we will not go over the details here.

Applying Lemma 3.3.4, we get the desired semi globally exponential stability result. □

Similar to the proposed solution in Equation (3.14), in the variant solution shown in Equation (3.24), c_r can be chosen in a distributed fashion, as long as the two timescale assumption holds. Practically, this implies that each user can adjust $n_r(t)$ according to the same control law but with a different rate. Specifically, a global setting among all the users is not necessary. Furthermore, allowing some of the c_r to be zero represents a scenario according to which the proposed scheme coexists with TCP. In this situation, all theorems still hold, except for a modification to Theorem 3.7.1. More precisely, we have the following Corollary:

Corollary 3.7.1. *For arbitrary topology, arbitrary number of users running either TCP or the variant solution shown in Equation (3.24), and arbitrary initial sending rates, the following holds:*

- *Sending rates $x(t)$ converge to a unique equilibrium:*
  - *Users running TCP:*

$$x_r^* = \frac{\sqrt{2}S}{T_r \sqrt{\sum_{j\in r} \left[g_j \left(y_j^* \right) + \epsilon_j \right]}} \tag{3.31}$$

– *Users running the variant solution as in Equation (3.24):*

$$x_r^* = \frac{\sqrt{2}S}{f(\sum_{j\in r} g_j(y_j^*))T_r\sqrt{\sum_{j\in r}\left[g_j\left(y_j^*\right)+\epsilon_j\right]}} \tag{3.32}$$

- *At the equilibrium, all bottlenecks are fully utilized for large β, and the following concave optimization problem is solved:*

$$\max_{x\geq 0}\sum_{r\in R} U_r(x_r) - \sum_{j\in J}\int_0^{y_j} g_j(z)\,dz, \tag{3.33}$$

with U_r being concave function:

– *Users running TCP:*

$$U_r(x_r) = -\sum_{r\in R}\frac{2S^2}{T_r^2 x_r} - x_r\sum_{j\in r}\epsilon_j,$$

– *Users running the variant solution as in Equation (3.24):*

$$U_r(x_r) = \int_0^{x_r} h_r^{-1}\left(\frac{2S^2}{T_r^2\nu^2}\right)d\nu,$$

where h_r^{-1} is the inverse of an monotonically increasing function h_r:

$$h_r(z) \triangleq \left(\sum_{j\in r}\epsilon_j + z\right)f^2(z) = \left(\sum_{j\in r}\epsilon_j + z\right)\left(\frac{e^{\beta z}-1}{e^{\beta z}+1}\right)^2, \quad r\in R.$$

Similar to proposed solution coexisting with TCP, an observation on the fairness among users is that at the equilibrium, users running the variant solution are fair to users running TCP in the following sense:

- In the case that TCP operates in capacity-limited region, i.e. all bottleneck links are fully utilized even if all users running proposed solution only open one connection, for large values of β, the approximated indicator function $f(\cdot)$ takes the value 1. Users running the variant solution will open only one connection, and their sending rates will converge to the same value as if they were running TCP. In this case, fairness among users running TCP and the variant solution is the same as fairness among TCP users.

- In the case that TCP operates in channel-error-limited region, i.e. the bottleneck links of some routers are underutilized, if users running the variant solution open one connection, for large values of β, approximated indication function $f(\cdot)$ takes on a value between 0 and 1, and approximated congestion loss function $g(\cdot)$ takes on a very small positive value, as seen from definition of $f(\cdot)$ in Equation 3.13. Consequently, as seen from users' equilibrium rates in the above Corollary, users running the variant solution will get the residual bandwidth without affecting TCP users's throughput, i.e. users running TCP get throughput close to what they would have gotten if no other users were running the proposed solution.

Chapter 4

Proposed Practical Solutions

In this chapter, we design practical solutions for flow control in wireless networks, following the insights gained from analysis in the previous Chapter. In particular, we will first discuss the guidelines for implementing the proposed solution in Equations (3.9) and (3.10) in practice. Then we design two practical solutions following the guidelines, namely Enhanced Multiple TCP (E-MULTCP) for data transmission and Enhance Multiple TFRC (E-MULTFRC) for multimedia streaming. We discuss how the parameters in E-MULTCP and E-MULTFRC are chosen to achieve the desired performance, and to balance bandwidth utilization and responsiveness to changes in network conditions. We also discuss the quantization drawback of E-MULTFRC related to operating multiple connections, and design a variant called Enhanced ALL-IN-ONE TFRC (E-AIOTFRC) to address it. Results of NS-2 simulations and actual experiments are provided in Chapter 5 characterizing the performance of all above schemes.

Table 4.1: One-to-one correspondence between congestion status and queuing delay.

Status	Queuing Delay
Congested	> 0
Not Congested	$= 0$

4.1 Design Guideline

We conceptually rewrite the proposed solution in Equations (3.9) and (3.10) as follows:

$$\begin{cases} \dot{x}_r(t) = n_r(t) \text{ connections of } TCP, \\ \dot{n}_r(t) = IIMD : \begin{cases} c_r[\frac{1}{n_r(t)} - n_r(t)], & \text{if route } r \text{ is congested,} \\ \frac{c_r}{n_r(t)}, & \text{otherwise.} \end{cases} \end{cases} \tag{4.1}$$

Above equations show design guidelines of practical schemes:

- Use TCP to control sending rate of individual connection, based on observed overall end-to-end packet loss rates;
- On a timescale much slower than the one at which TCP operates, control the number of TCP connections according to IIMD control law, based on additional *one bit of information* indicating whether user's route is congested or not.

Since TCP operates on a timescale from milliseconds to seconds, we choose to adjust the number of connections on the order of tens of seconds, in order to satisfy the two timescale assumption.

One bit of information indicating congestion status of user's route is also needed to design practical schemes. To get the information from end-to-end measurement, it is well-known that there is one-to-one correspondence between congestion status of a route and the positivity of queuing delay along it, as shown in Table 4.1. Hence, we can measure an average queuing delay in the interval, over which the number of connections is adjusted.

As $n_r(t)$ shown in Equation (4.1) is not required to be an integer, in practice one can either quantize it to the closest integer, or mimic sending rate of $n_r(t)$ connections by opening one TFRC connection with rate equal to that of $n_r(t)$ connections. Alternatively, it is possible to open an integer number of connections with varying packet sizes, such that the aggregate sending rate is the same as that of $n_r(t)$ connections with standard packet size. For example. if we want to mimic sending rate of 1.5 connections with standard packet size, we can open 2 connections with packet size chosen to be 0.75 of the standard one.

Following these guidelines, we design practical schemes for data transmission, as well as for multimedia streaming, in wireless networks.

4.2 E-MULTCP for Data Transmission

The framework of our proposed E-MULTCP is shown in Figure 4.1. As seen, there are two components in the system: *RTT* Measurement Subsystem (RMS), and Connections Controller Subsystem (CCS).

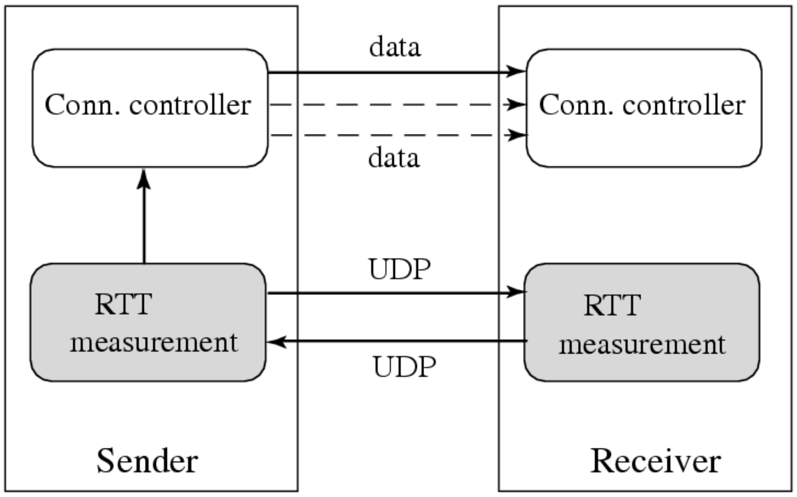


Figure 4.1: E-MULTCP system framework.

4.2.1 RMS

The gray blocks in Figure 4.1 represent RMS. They measure round trip time samples between sender and receiver, denoted by rtt_{sample}, by computing difference between the time sender emits a packet, and the time it receives the ACK from receiver. These packets are sent through the forward and backward UDP connections between sender and receiver.

To reduce overhead and to prevent congestion collapse, the rate at which RTT-measurement packets are sent, is set to be the same as that of data packets, which is estimated on sender side by measuring the number of data packets sent in the previous round trip time interval. The RTT measurement packets contain only an IP header and a timestamp, a total of 28 bytes. In the case when TCP uses typical packet size of 1500 bytes, the RTT measurement overhead is 2% of the total throughput.

After waiting for a fixed interval, denoted by τ, RMS computes the running average, ave_rtt, of these rtt_{sample}s, and reports it to the CCS. As such, $1/\tau$ is the frequency of adjusting number of connections; τ has to be large enough to ensure the frequency is much lower than that of changing the source rate, which is typically of the order of round trip time. In our current implementation, we choose τ to be 20 seconds. It should be noted here that our formulation in Section 3.4 allows variable settings of τ, rather than a fixed value, such as 20 seconds.

4.2.2 CCS

The CCS is shown as the white blocks in Figure 4.1. Its basic functionality is to properly control the number of connections n, following the control law shown in Equation (4.1). CCS at the sender adjusts the number of connections n roughly by IIMD law, based on route congestion status. Specifically, ave_rtt is reported to CCS by RMS, CCS sets the rtt_min as the minimum over all ave_rtt seen so far, and then adapts the number of connections n as follows:

$$n = \begin{cases} \beta n + \alpha/n, & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ n + \alpha/n, & \text{otherwise.} \end{cases} \tag{4.2}$$

where $\alpha = 1 - \beta < 1$ and γ is a preset parameter. This is nothing but a discrete implementation of the IIMD control law shown in Equation (4.1), with $c_r = \alpha/\tau$. The congestion status is estimated by comparing the measured queuing delay $ave\_rtt - rtt\_min$ with a dynamic threshold γrtt_min. For a given route, the rtt_min is a constant representing the minimum observed round trip time for that route, approximating physical propagation delay. As such, $ave\_rtt - rtt\_min$ corresponds to current queuing delay, and γrtt_min is a threshold on the queuing delay that E-MULTCP can tolerate before it starts to decrease the number of connections. Therefore, under ideal conditions, E-MULTCP keeps increasing the number of connections to make *ave_rtt* as close as possible to $(1+\gamma)rtt\_min$ without exceeding it.

4.2.3 Discussion

In the increasing stage, i.e. $I(ave\_rtt - rtt\_min > \gamma rtt\_min) = 0$, E-MULTCP has an increasing rate of $\dot{n}(t) = \alpha/(\tau n(t))$ according to Equation 3.10. Hence, to increase n from N_2 to N_1, assuming $N_1 > N_2$, it roughly takes E-MULTCP $\frac{\tau}{2\alpha}(N_1^2 - N_2^2)$ seconds. α is defined as the normalized increasing rate, i.e. the increasing rate $\dot{n}(t)$ normalized by $\tau n(t)$. It is a crucial design parameter independent of $n(t)$ and τ, representing how fast E-MULTCP shifts from channel-error-limited region to capacity-limited region. The larger α is, the faster E-MULTCP increases its overall rate.

The decreasing rate for the decreasing stage is roughly $\dot{n}(t) = -\frac{\alpha n(t)}{\tau}$, and hence it takes E-MULTCP $\frac{1}{\alpha}\tau \ln(N_1/N_2)$ seconds to decrease n from N_1 to N_2. Thus, E-MULTCP is conservative in adding connections, but much aggressive in closing them.

In a simple topology with one E-MULTCP system over one wireless link, we can use the above results to compute bandwidth utilization ratio. At steady state, E-MULTCP periodically increases the number of connections n to the optimal n^* that fully utilizes the wireless bandwidth, and proportionally decreases n upon reaching n^*. The approximated continuous version of this process is demonstrated in Figure 4.2. In the plot, T is the time for $n(t)$ to increases from βn^* to n^*, and hence is $(n^*)^2(1-\beta^2)\tau/2\alpha$. The number of connections is given by $n(t) = \sqrt{(t-kT)2\alpha/\tau + (\beta n^*)^2}$ for $kT \le t \le (k+1)T$ and $k \in Z^+$.

The utilization ratio, at steady state, is then the ratio between the average number of connections and n^*, as follows:

$$\frac{1}{Tn^*}\int_0^T n(\delta)d\delta = \frac{2}{3}\frac{1+\beta+\beta^2}{1+\beta}. \tag{4.3}$$

This ratio is at least 2/3, only depends on β, and is independent of wireless packet loss rate, τ, and n^*. The larger β is, the high utilization ratio is. There is clearly a trade-off between above utilization ratio and normalized increasing rate, i.e. $\alpha = 1-\beta$, as shown in Figure 4.3. It is up to designer to select β to get the best balance for targeting scenario.

One of our main target scenarios for E-MULTCP is cell phones connecting to Internet via commercial wireless networks and downloading data or video from servers. As described later, we have empirically found that 2 or 3 parallel connections are sufficient to

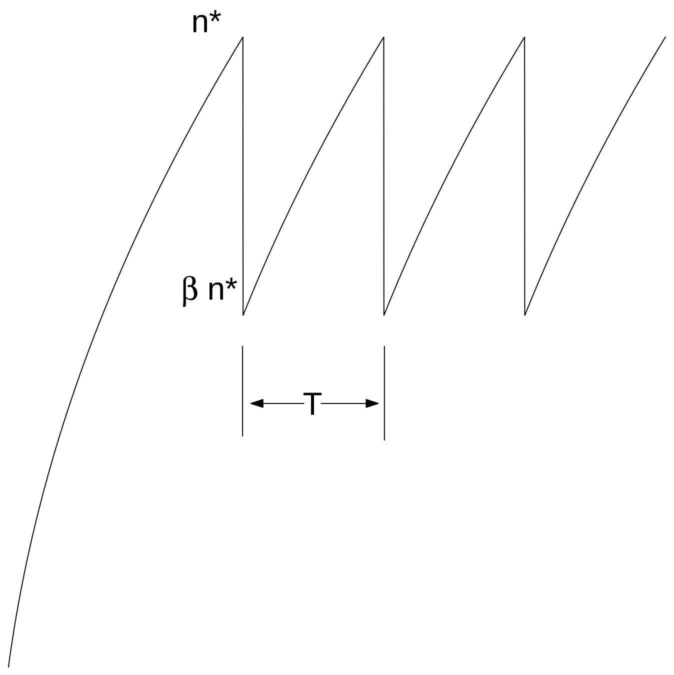


Figure 4.2: Demonstration of the change on the number of connections n, controlled by single E-MULTCP over single wireless link.

fully utilized 1xRTT CDMA and EVDO wireless bandwidth, which are around 110 kbps and 380 kbps, respectively [3]. As the wireless bandwidth is a scarce resource and limited in nature, and the optimal number of connections n^* in most practical situations is small, it is more important to achieve high utilization than high increasing rate. Moreover, most cell phones are limited in power, so it would be better if β is selected to make E-MULTCP control procedure computationally efficient, by replacing multiplication operations by binary shifts.

As such, we have selected $\beta = 0.75$ for E-MULTCP, to get a utilization ratio of 0.88, and a normalized increasing rate of 0.25. This means it takes E-MULTCP $2\tau(N_1^2 - N_2^2)$ seconds to increase from N_2 connections to N_1 connections. The IIMD of n requires binary shifts and additions only, and is therefore computationally efficient.

When there is a route change either due to change in the wireless base station, or due to route change within the wired networks, the value of *rtt_min* changes significantly, affecting the performance of E-MULTCP. Under these conditions, it is conceivable to use route change detection tools such as traceroute [51] at the sender to detect the route change, in order to reset *rtt_min* to a new value. Furthermore, it can be argued that the overall

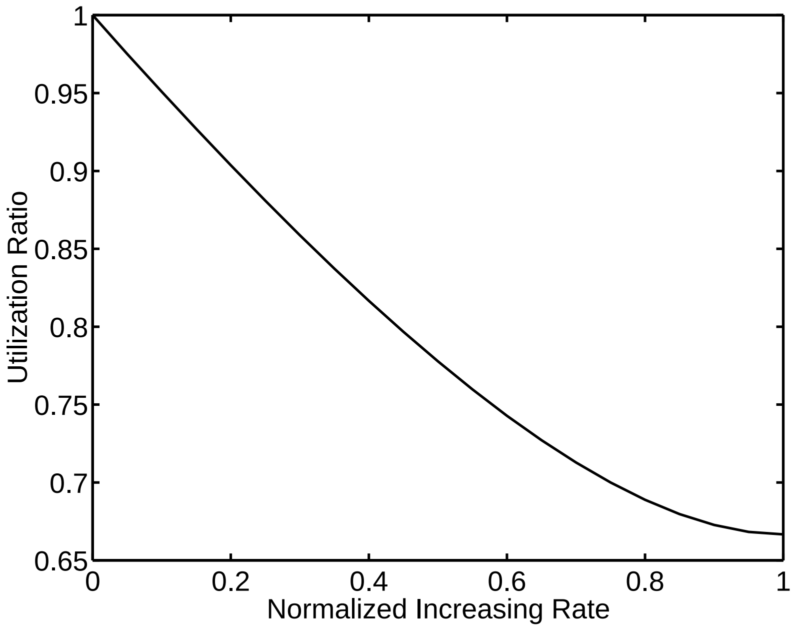


Figure 4.3: A trade off between utilization ratio and normalized increasing rate α.

throughput of E-MULTCP will not go to zero, resulting in starvation; this is because E-MULTCP always keeps at least one connection open.

Since the data stream in E-MULTCP is transmitted using multiple connections, the receiver could potentially receive out of order packets. However, reordering application packets from multiple TCP connections using a receive buffer is a rather mature technology widely used in peer-to-peer applications, such as BitTorrent file sharing, e.g. Kazza [52], and peer-to-peer streaming, e.g. PPlive [53]. As discussed in later chapters, we have also implemented and successfully tested a file downloading software on an actual BREW [54] cellular telephone, combining E-MULTCP and a simple packets reordering procedure. We refer interested readers to details of reordering technology in literature [55].

In summary, in the E-MULTCP system, the number of connections is controlled according to a discrete version of Equation (3.10) at the sender; the sending rate of each TCP connection is adjusted automatically by itself. The rate of change of number of connections is expected to be much slower than that of sending rate satisfying the two time scale assumption in the previous section. Thus, the optimality and stability analysis in the previous section for the dynamic system in Equations (3.9) and (3.10) applies to

E-MULTCP, indicating that E-MULTCP results in a stable, yet fully utilized network as shown in Equation (3.20).

4.3 E-MULTFRC for Multimedia Streaming

Although the analysis in previous section is based on TCP model, it can be extended to TFRC, since it has been shown TFRC has the same stationary behavior as TCP [2]. As we are interested in rate control for streaming over wireless, we design a practical scheme for that, based on our analysis and control law in Equation (4.1), by adjusting the number of TFRC connections.

The framework of our proposed system which we refer to as E-MULTFRC is shown in Figure 4.4. As seen, the system is quite similar to that of E-MULTCP shown in Figure 4.1. There are two components in the system: *RTT* measurement subsystem (RMS), and connections controller subsystem (CCS).

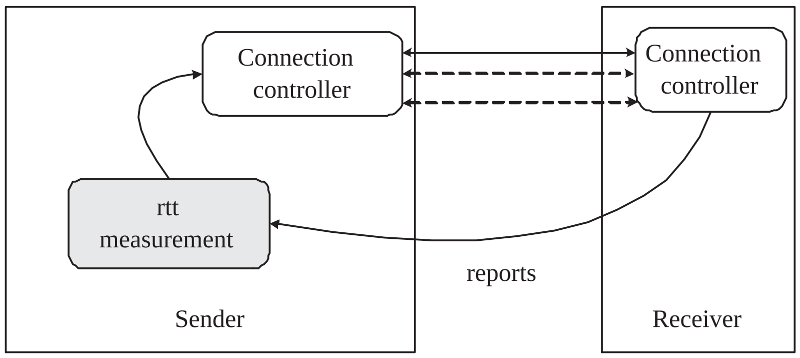


Figure 4.4: E-MULTFRC system framework.

4.3.1 RMS

The gray block in Figure 4.4 represents RMS that resides at the sender; it receives reports from receiver every round trip time, containing average round trip time rtt_{sample} measured in the past round trip time window. After waiting for a fixed interval, denoted by τ, RMS computes the running average, *ave_rtt*, of these rtt_{sample}s, and reports it to the CCS. Here, $1/\tau$ defines the frequency of adjusting number of connections; it has to be large enough to ensure the frequency is much lower than that of changing the source rate, which is typically of the order of round trip time. In our current implementation, we choose τ to be 20 seconds.

4.3.2 CCS

The CCS subsystem in E-MULTFRC is the same as the one in E-MULTCP, hence we will not repeat the details here.

4.4 Multiple TFRC (MULTFRC) for Multimedia Streaming: A Variant of E-MULTFRC

As seen in previous section, E-MULTFRC follows the proposed solution in Chapter 3. In this section, we develop a scheme called MULTFRC that is a variant of proposed solution, as shown in Section 3.7, and a variant of E-MULTFRC.

MULTFRC shares the same design principles as E-MULTFRC, expect that it applies IIAD instead of IIMD to control the number of TFRC connections. That is, MULTFRC also has a RMS and a CCS subsystem, but the CCS system controls the number of connections n, as follows:

$$n = \begin{cases} n-1, & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ n + \alpha/n, & \text{otherwise.} \end{cases} \tag{4.4}$$

Similar to the analysis in Section 4.2.3, adaptation rate and utilization of MULTFRC in a simple topology with one MULTCP system over one wireless link can be analyzed as follows.

In the increasing stage, MULTFRC has the same increasing rate as E-MULTFRC, as they share the same increase law on n. For example, to increase the number of connections n from N_2 to N_1, assuming $N_1 > N_2$, it roughly takes MULTFRC $\frac{\tau}{2\alpha}(N_1^2 - N_2^2)$ seconds. On the other hand, the decreasing rate of MULTFRC for the decreasing stage is roughly $\dot{n}(t) = -\frac{1}{\tau}$, and hence it takes MULTFRC $\tau(N_1 - N_2)$ seconds to decrease n from N_1 to N_2.

The utilization ratio of MULTFRC, at steady state, is then the ratio between the average number of connections and n^*. Note increasing from $n^* - 1$ to n^* takes $T = \tau(2n^* - 1)/(2\alpha)$, and $n(t) = \sqrt{2[\alpha t/\tau + n^2(0)]}$ in the increasing stage, the utilization ratio can be computed as follows:

$$\frac{1}{Tn^*}\int_0^T n(\delta)d\delta = \frac{2}{3}\left(1 - \frac{1}{n^*}\frac{3n^* - 2}{6n^* - 3}\right) > 1 - \frac{1}{2n^*}. \tag{4.5}$$

This ratio is at least $(2/3)^2$, only depends on n^*, and is independent of α and τ. As $n^* \to \infty$, the utilization ratio tends to be 1. Hence, large values of n^* improve the utilization ratio in this scenario. As large values of n^* typically represent poor wireless channel conditions, theoretically MULTFRC has better utilization ratio as the channel gets worse.

4.5 E-AIOTFRC for Multimedia Streaming

There are mainly two drawbacks associated with E-MULTFRC. First drawback has to do with bandwidth underutilization, and the other two are with implementation complexity. We will begin with utilization drawback. As we will show in the next chapter, E-MULTFRC may not fully utilize the wireless bandwidth when wireless channel error rate is small. This suboptimal performance has two causes. First one is the control behavior described in Equation (4.2): as described, n is decreased when the full utilization of bottlenecks is detected, and is inversely increased until the next full utilization is detected. During this period, bottlenecks stay underutilized, resulting in suboptimal average throughput. It is impossible to remove this sub-optimality determined by the control law without changing the law. The second reason for bandwidth underutilization is the "quantization effect" in E-MULTFRC whereby in practice the number of connections is forced to be an integer. This loss of granularity typically results in bandwidth underutilization. For example, if the optimal number of connections has been determined to be 1.5, then n is forced to take fractional values between 1 and 3, e.g. 1, 1.25, 1.45, 2.14, 1.14, ..., as dictated by Equation (4.2). E-MULTFRC then quantizes n to the closest integer to oscillate between one and two, resulting in loss of throughput granularity. This effect can be eliminated by avoiding the quantization step, as described below.

The second drawback of E-MULTFRC is of a more practical nature. Operating multiple connections in one application could potentially consume too much system resources. For example, each TFRC connection uses a different port to send out data packets, carries out individual feedback process, and updates the loss event rate and RTT even though they are highly correlated for these TFRC connections. Clearly, there is unnecessary overhead associated with operating multiple connections, in terms of computation, processing power, memory, and ports, particularly for today's low power, resource-limited handheld devices.

In this section we propose a new scheme E-AIOTFRC, in order to address these drawbacks, while retaining the same control law for n as in E-MULTFRC. To achieve this goal, we integrate the Bandwidth Filtered Loss Detection (BFLD) technique from [56], to be described shortly, together with the control law in Equation (4.1) to construct the E-AIOTFRC system. The system framework is shown in Figure 4.5. Basically, the Sink at the receiver feeds back the RTT and loss event rate to the sender. The sender then adjusts of n based on Equation (4.1), and sends out the data packets at a rate of n times that of one TFRC's sending rate. The functionalities of E-AIOTFRC senders and receivers are described as follows:

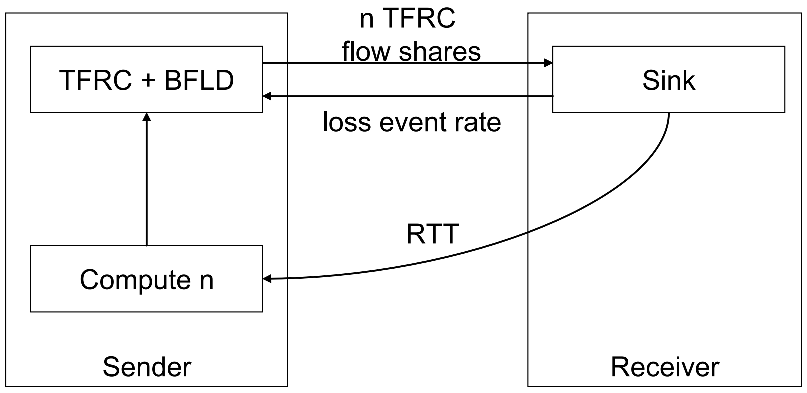


Figure 4.5: The system framework of E-AIOTFRC.

- *Sender*: There are two functional components in the sender. One component is represented by the "compute n" block. It receives the RTTs from the receiver, computes an ave_rtt, by averaging these RTT samples over a 20 second window, then updates n according to Equation (4.2) every 20 seconds, i.e. using the same law as E-MULTFRC. For E-AIOTFRC, we choose $\alpha = 1 - \beta = 0.25$, and $\gamma = 0.5$.

  The other component is represented by the "TFRC+BFLD" block, and it has two functionalities: first, it obtains the updated n from the "compute n" component, as well as the loss event rate, denoted as p_l from the receiver. It then computes the TCP friendly rate of one TFRC connection as the standard TFRC does [2], which is roughly proportional to $\frac{k}{T_r\sqrt{p_l}}$ with k being a constant and T_r being the measured round trip time, and adjusts the sending rate to be n times that of one TFRC.

The second functionality of the "TFRC+BFLD" block in the sender is to mark the headers of selected data packets before they are sent out. The data packets to mark are selected in such a way that they form a virtual single TFRC flow, and hence correspond to $1/n$ of all the outgoing packets. For example, if $n = 1.5$, then "TFRC+BFLD" evenly marks 2/3 of all outgoing packets. The reason for the marking is to facilitate the loss event rate measurement at the receiver, and will be explained shortly.

- *Receiver*: The E-AIOTFRC Sink component reports the RTT and the loss event rate of the virtual TFRC connection to the sender every RTT. The only difference between the E-AIOTFRC Sink and the original TFRC Sink is that the E-AIOTFRC Sink measures and updates the loss event rate based on the virtual TFRC flow with marked packets.

  The operation flow of E-AIOTFRC is as follows. Every RTT, the receiver sends back the measured RTT and loss event rate to the sender. Based on the RTT, the sender adjusts n according to Equation (4.2) every 20 seconds. At any moment, the sender sends at a rate equivalent to n TFRC flow shares, and marks the selected outgoing packets to form a virtual stream, which is used for the receiver to carry out loss event rate measurements.

It should also noted that all the functionalities of sender can be shifted to and implemented at receiver:

- Virtual sampling on sender side can be done instead at receiver by taking only $1/n$ of all incoming and lost packets into loss event rate measurement and update.

- Based on the measured round trip time, receiver can carry out the adjustment of n, discount the measured packet loss event rate p_l, by $1/n^2$, and send this discounted packet loss event rate p_l/n^2 to sender. Using this discount packet loss event rate and the measured round trip time T_r, sender computes the source rate as $\frac{nk}{T_r\sqrt{p_l}}$, equivalent to that of n TFRC connections. Hence, the sender will send out the packets at a rate equivalent to n TFRC connections.

By shifting all functionalities to receiver, it is possible to implement E-AIOTFRC scheme by only modifying the receiver, i.e. the client side which is typically a PC, laptop or a handheld device. This implementation requires no modification at the sender, i.e. the server side, potentially making E-AIOTFRC even easier to deploy.

We now explain the reasoning behind the marking process. It has been argued in [56] that if the sending rate of the application is adjusted to be n times that of one TFRC, then TFRC Sink at the receiver underestimates the loss event rate based on using all the received packets. TFRC Sink records the beginning of a loss event when a packet loss is detected. The loss event ends when, after a “guarding” period of one RTT, another packet loss is detected. A loss event interval is defined as the difference in sequence numbers between the above two lost packets; the loss event rate is thus estimated by taking the inverse of this difference. In the case where all the packets are used to estimate loss event rate, the number of packets received by TFRC Sink, during the guarding RTT, is n times that of one TFRC. As such, the loss event interval now could be n times of that of single TFRC, and the loss event rate is underestimated.

To overcome this problem, we sample the outgoing packets at the sender to form a single virtual TFRC stream at the sender, and modify TFRC Sink to carry out the loss event rate measurement based on this virtual stream. This is the functionality of BFLD as verified in [56].

If otherwise the deflated loss event rate is reported to the sender, the aggregate sending rate will in fact be higher than n TFRCs’ flow shares. This is because the aggregate sending rate is inversely proportional to the square root of its measured loss event rate, which is smaller than those measured by individual TFRC flow. Consequently, this aggressive sending rate could potentially cause congestion collapse or unfairness to TCP. This is also of concern to the MULTFRC-LERD scheme in [57].

We now explain the reason to choose a larger γ in E-AIOTFRC than in E-MULTFRC. E-MULTFRC achieves n TFRC flow shares by opening $\bar{n}$ independent TFRC connections, while E-AIOTFRC achieves n flow shares by opening one connection and sending at n times the sending rate of one TFRC connection. In situations where the throughput of each TFRC connection is a function of the random packet loss rate, e.g. that caused by random wireless channel error, it is well known the latter method results in a higher variance in the aggregate sending rate. Since the queueing delay along the route is a function of the aggregate sending rate, E-AIOTFRC experiences higher queuing delay variance along the path. Therefore, in order to achieve the same level of confidence in the measured queueing delay, used to adjust n in Equation (4.2), E-AIOTFRC needs a larger threshold than E-MULTFRC. In our current implementation, we have empirically chosen $\gamma = 0.5$ for E-AIOTFRC.

Chapter 5

Simulations and Experiments

In this chapter, we carry out NS-2 [47] simulations and experiments over Verizon Wireless 1xRTT and EVDO CDMA data network to evaluate the performance of E-MULTCP, E-MULTFRC and E-AIOTFRC. We use TCP NewReno implementation for TCP protocol in all simulations.

5.1 E-MULTCP: NS-2 Simulations, 1xRTT and EVDO Wireless Experiments

In this section, we carry out NS-2 [47] simulations and actual experiments over Verizon Wireless 1xRTT and EVDO CDMA data network to evaluate the performance of E-MULTCP.

5.1.1 Setup

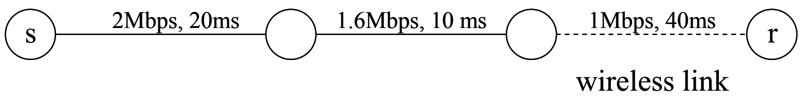


Figure 5.1: Simulation topology.

The topology used in simulations is shown in Figure 5.1. The sender denoted by s, and the receiver denoted by r, both run E-MULTCP at the application layer. The number of connections, as mentioned in the previous section, is controlled by the sender. For all simulations, the wireless bandwidth B_w is set be 1 Mbps, and is assumed to be the bottleneck. The wireless link is modeled by an exponential error model, and p_w varies from 0.0 to 0.08 in increments of 0.02. DropTail type queue is used for each node. In order to evaluate E-MULTCP's performance in the presence of wireless channel errors, we examine three issues; first, how E-MULTCP performs in terms of average throughput, average round trip time, and packet loss rate, as a function of p_w. Second, whether the number of connections is stable. Third, whether or not an E-MULTCP application can fairly share with an application using one TFRC or one TCP connection. In all the simulations, throughput is measured every second, packet loss rate is measured every 30 seconds, the average round trip time is measured every 100 packets, and the number of connections is sampled whenever there is a change.

For the actual experiments over 1xRTT, we stream from a desktop connected to Internet via 100 Mbps Ethernet in eecs.berkeley.edu domain, to a notebook connected to Internet via Verizon Wireless 1xRTT and EVDO CDMA data network. Thus it is quite likely that the last 1xRTT or EVDO CDMA link is the bottleneck for the connection. We measure the average throughput, average number of connections, and packet loss rate.

5.1.2 Performance Characterization of E-MULTCP

Following the analysis and arguments in Section 4.2.3, we use following parameters for E-MULTCP in simulations and experiments to achieve reasonable balance between utilization and adaptation rate: $\alpha = 0.25, \beta = 0.75$. We select $\gamma = 0.2$, and $\tau = 20$ seconds. Intuitively, larger τ results in more reliable estimates of round trip time, but at the expense of a lower sampling frequency, resulting in a less responsive system. Larger γ results in a system that is more robust to round trip time estimates, but at the expense of longer queues in routers, and hence a longer queueing delay.

We simulate the E-MULTCP system to send data for 9000 seconds, compute the average throughput and packet loss rate for p_w =0.0, 0.02, 0.04, 0.06 and 0.08[1], and compare

[1] It is well known that TCP's degraded performance is dominated by timeout if wireless packet loss rate p_w is higher than 0.1. As our model does not capture the timeout effect, we are more interested in cases where $p_w < 0.1$.

them to the optimal, i.e. $B_w(1 - p_w)$ for each p_w. The results for $B_w = 1\ Mbps$, $\beta = 0.75$, and $RTT_{min} = 168\ ms$ are shown in Figure 5.2. As seen, the throughput is within 25% of the optimal, and is reasonably close to the predicted one, i.e. the product of the optimal and the utilization ratio computed using Equation (4.3). The average round trip time is within 20% of RTT_{min}, the same as the expected range, i.e. γRTT_{min}. and the packet loss rate is almost identical to the optimal, i.e. a line of slope one as a function of wireless channel error rate. As expected, the average number of connections increases with wireless channel error rate, p_w. [2]

To demonstrate the trade-off between utilization ratio and normalized increasing rate of E-MULTCP with different values of β, we have also carried out simulations using the same setting as above, but with $\beta = 0.5$. The results are shown in Figure 5.3. As seen, the throughput is lower than the above simulation with $\beta = 0.75$, as expected; this is because on average fewer connections are opened. We will see later the setting $\beta = 0.5$ leads to a fast adaptation to changes of wireless channel conditions.

Considering the throughput plot in Figure 5.2, for some values of p_w, there is a significant difference between the actual and optimal throughput. This is due to the quantization effect in situations where the number of connections is small, i.e. 2 to 4. In these situations, a small oscillation around the optimal number of connections results in large variation in observed throughput. One way to alleviate this problem is to increase γ in order to tolerate larger queuing delay and hence absorb throughput fluctuations, at the expense of being less responsive. Another alternative is to use smaller packet size in order to reduce the "quantization effect" at the expense of (a) larger overhead and hence lower transmission efficiency, and (b) the slower rate of convergence to the optimal number of connections.

One might also notice that the difference between E-MULTCP's throughput and the optimal becomes larger as p_w increases. This is due to increased timeout events as p_w increases. Upon timeout, TCP slow starts again, generating bursty traffic that results in transient queuing delay. Consequently, detecting congestion based on RTT estimate becomes less reliable, decreasing the average throughput. This effect is more pronounced for large p_w and number of connections.

In order to examine E-MULTCP's adaptation rate to changes of wireless channel

[2]Note the round trip time for $p_w = 0$ is not shown in Figure 5.2 because it represents the case when the channel is error free. In this case, E-MULTCP reduces to one TCP connection.

	ramping up time (s)	predicted (s)	ramping down time (s)	predicted (s)
$\beta = 0.75$	660	650	80	73
$\beta = 0.5$	313	325	32	36

Table 5.1: Adaptation rates of E-MULTCP with different values of β

condition, we test E-MULTCP with p_w initially set at 0.02, switched to 0.06 at 3000^{th} second and switched back at 6000^{th} second. The throughput, packet loss rate, round trip time and the number of opened connections are shown in Figure 5.4 for $\beta = 0.75$, and in Fig 5.5 for $\beta = 0.5$.

As seen, the number of connections varies from around 2-3 to around 5 as p_w switches from 0.02 to 0.06. The ramp up and ramp down times from 2 to 5, and 5 to 2 connections for $\beta = 0.75$ and $\beta = 0.5$ are shown in Table 5.1. As seen, the adaptation performance is close to the theoretical predictions, which are based on analysis in Section 4.2.3. Combined with the observations in Figures 5.2 and 5.3, we can see that smaller β results in a faster adaptation rate to changes in wireless channel conditions, but a lower utilization ratio. These observations are consistent with our analysis on the trade-off between utilization ratio and adaptation rate, in Section 4.2.3.

5.1.3 Experimental Results for E-MULTCP in 2004 and 2005

Similar experiments are carried out on Verizon Wireless 1xRTT CDMA data network. The 1xRTT CDMA data network is advertised to operate at data speeds of up to 144 kbps for one user. As we explore the available bandwidth for one user using UDP flooding, we find the highest average available bandwidth averaged over 30 minutes to be between 80 kbps to 97 kbps. In our experiments, we stream for 30 minutes from a desktop on wired network in eecs.berkeley.edu domain to a laptop connected via 1xRTT CDMA modem using E-MULTCP, E-MULTFRC and TCP.

The results are shown in Table 5.2 for packet size of 1460 bytes. As seen, on average E-MULTCP opens up 1.7 connections, and results in 58% higher throughput, at the expense of a larger round trip time, and higher packet loss rate.

We have also carried out experiments over Verizon Wireless EVDO data network, by sending 5 MB files from a desktop on wired network in eecs.berkeley.edu domain to an

Table 5.2: Experimental results for E-MULTCP and E-MULTFRC systems over 1xRTT CDMA.

scheme	**throughput (kbps)**	**rtt (ms)**	**ave. # of conn.**	**throughput improvement over one TCP (%)**
one TCP	59	1954	N/A	N/A
E-MULTCP	93	2447	1.7	58
E-MULTFRC	89	2767	1.9	51

EVDO cell phone using E-MULTCP and TCP. A file size of 5 MB is chosen to be typical of a MP3 song. The results are shown in Table 5.3 for packet size of 1460 bytes. As seen, on average E-MULTCP opens up 1.8 connections, and results in 43% higher throughput, at the expense of a larger round trip time, and higher packet loss rate.

Table 5.3: Actual experimental results for E-MULTCP and TCP over commercial EVDO data networks.

scheme	**throughput (kbps)**	**rtt (ms)**	**ave. # of conn.**	**throughput improvement over one TCP (%))**
one TCP	249	702	N/A	N/A
E-MULTCP	355	946	1.8	43

5.1.4 Additional Experiments for E-MULTCP in 2006

All of the experiments in Section 5.1.3 were carried out in years 2004 and 2005. In Fall 2005, Verizon Wireless in USA launched a new data service EV-DO, which provides up to 2 Mbps data rate. We speculate that many 1xRTT users switched from 1xRTT to EV-DO network due to the higher data rates. Consequently, as subscription level to 1xRTT network dropped, it improved the overall throughput of the 1xRTT network for the remaining subscribers. We repeated experiments to evaluate E-MULTCP's performance under new channel conditions in Fall 2006 to reconfirm improved performance of E-MULTCP over TCP. The results are reported in this subsection.

We implement E-MULTCP on a laptop running Windows XP, as well as on a cell phone running BREW [54]. BREW, standing for Binary Runtime Environment for Wireless, is a development platform and coding language created by Qualcomm for cell phones. BREW is a middleware that allows developers to easily port their applications onto every cell phone using Qualcomm chipsets.

As we explore the available bandwidth for one user using UDP flooding, we find the highest average available bandwidth averaged over 30 minutes to be between 101 kbps and 112 kbps. This is considerably larger than what we observed in Fall 2004 in US. In the experiments in 2006, we send data for 1, 2, 5, and 30 minutes from a desktop on wired network in eecs.berkeley.edu domain to a laptop and to a BREW cell phone connected via 1xRTT CDMA modem using E-MULTCP and TCP. The results are shown in Table 5.4 for packet size of 1460 bytes. Each data point is averaged over 5 independent runs. As seen, on average E-MULTCP opens up around 1.4 connections, and results in 30% higher throughput, at the expense of a larger round trip time, and higher packet loss rate. Although these experiments result in smaller improvement over TCP than the one shown in Section 5.1.2, they still show that E-MULTCP outperforms TCP. The reason for the smaller improvement is the 1xRTT channel condition in 2006 being better than in 2004 and 2005. From analysis in Section 3.2 and Equation 3.8, it can be shown if aggregate packet loss rate caused by channel error for user r, i.e. $\sum_{j \in r} \epsilon_j$, becomes smaller, then utilization of TCP/TFRC becomes higher. As TCP/TFRC achieves higher utilization in 1xRTT in 2006 as compared to 2004, there is less room for improvement for E-MULTCP. Nevertheless, the insights still hold: as long as TCP/TFRC performs in channel-error-limited region, proposed solutions such as E-MULTCP always improve throughput.

Table 5.4: Actual experimental results for E-MULTCP over 1xRTT network in year 2006.

scheme and setup	**throughput (kbps)**	**rtt (ms)**	**ave. # of conn.**
one TCP (1 min)	81	712	N/A
E-MULTCP (1 min)	107	1130	1.3
Improvement (%)	32	N/A	N/A
one TCP (2 min)	84	708	N/A
E-MULTCP (2 min)	110	1340	1.3
Improvement (%)	31	N/A	N/A
one TCP (5 min)	86	723	N/A
E-MULTCP (5 min)	108	1372	1.3
Improvement (%)	25	N/A	N/A
one TCP (30 min)	85	714	N/A
E-MULTCP (30 min)	108	1436	1.3
Improvement (%)	27	N/A	N/A

5.1.5 Fairness between E-MULTCP and TCP

To investigate the fairness of E-MULTCP, we carry out NS-2 simulations based on the "dumbbell" topology shown in Figure 5.6. Senders are denoted by $si, i = 1, \ldots, 16$, and receivers are denoted by $di, i = 1 \ldots, 16$. We investigate two types of fairness: the inter-protocol fairness between E-MULTCP and TCP, and the intra-protocol fairness within E-MULTCP.

The intra-protocol fairness is defined as the fairness between E-MULTCP flows. In our simulations, we run E-MULTCP on all 16 sender-receiver pairs shown in Figure 5.6 for 5000 seconds, and compare their throughput. E-MULTCP is said to be intra-protocol fair if all receivers achieve the same throughput. The fairness ratios for $p_w = 0.01$ and $p_w = 0.04$ are shown in Table 5.5. The fairness ratio is defined as receivers' throughput divided by the average throughput; the closer to one, the more fair the E-MULTCP system is. As seen, the fairness ratio is fairly close to one, indicating E-MULTCP flows are fair to each other, at least in this simulation setting. The bandwidth utilization ratios are 96% for $p_w = 0.01$ and 98% for $p_w = 0.04$.

Table 5.5: Simulation results for intra-protocol fairness of E-MULTCP.

receiver	fairness ratio p_w=0.01	fairness ratio p_w=0.04	receiver	fairness ratio p_w=0.01	fairness ratio p_w=0.04
d1	1.06	1.03	d9	1.07	0.98
d2	1.00	1.01	d10	0.97	0.98
d3	1.06	1.03	d11	1.02	1.02
d4	1.02	1.08	d12	1.02	0.99
d5	0.90	0.98	d13	0.98	0.97
d6	0.93	0.98	d14	0.89	1.00
d7	1.02	1.04	d15	1.01	1.01
d8	0.99	1.01	d16	0.98	0.94

The inter-protocol fairness is defined as the fairness between E-MULTCP and TCP. In our simulations, we run E-MULTCP on the first 8 sender-receiver pairs, i.e. $(si, di), i = 1, \ldots, 8$, and TCP on the remaining 8 sender-receiver pairs shown in Figure 5.6; each session lasts 5000 seconds, and we compare their throughput for $p_w = 0.01$ and $p_w = 0.02$. Under the simulation settings, each E-MULTCP consumes more bandwidth than one TCP under full utilization. This is because the wireless channel error rate is large enough to make the number of virtual connections of each E-MULTCP to be larger than one. Hence, it is meaningless to define the fairness between E-MULTCP and TCP as having the same throughput.[3] As such, in our simulations, we define E-MULTCP to be fair to TCP if it does not result in a significant decrease in TCP's throughput as compared to TCP throughput in the absence of E-MULTCP. Specifically for our simulations, it implies TCP retains the same throughput whether or not it coexists with E-MULTCP under the same network setting. The throughput of E-MULTCP and TCP, as well as the total bandwidth utilization ratios for the setup shown in Figure 5.6, are shown in Table 5.6 for two scenarios: (a) 8 E-MULTCP coexisting with 8 TCP connections, (b) 16 TCP connections. Figures 5.7 and 5.8 also show the dynamics of throughput, packet loss rate, RTT, and the number of virtual connections n for the case where 8 TCP connections coexist with 8 E-MULTCP connections. Comparing E-MULTCP+TCP with TCP-alone in Table 5.6, we see the former achieves a much higher utilization of the wireless bandwidth at the expense of slightly lower TCP throughput. A careful examination of Figure 5.7 reveals that this throughput drop is caused by the higher RTT experienced by TCP connections in the E-MULTCP+TCP scenario as compared with TCP-alone scenario. For example, for $p_w = 0.01$ and $\gamma = 0.2$, The RTT for E-MULTCP+TCP is around 0.56 seconds, while for TCP-alone is 0.5 seconds, i.e. the propagation delay[4]. As TCP's throughput is known to be inversely proportional to RTT, the 12% increase in the RTT of TCP connections roughly explains the 11% decrease in the TCP's throughput shown in first row in Table 5.6.

This increase in the RTT experienced by TCP is, by design, a consequence of E-MULTCP controlling n according to Equation (4.2). As n is only decreased after the queuing delay exceeds the threshold γrtt_min, round trip time is increased when E-MULTCP increases n to achieve full utilization. One way to address this problem is to use a smaller value for γ, in order to reduce the increase in the RTT, and hence minimize the TCP's throughput drop. However, smaller values of γ also result in lower bandwidth utilization due to increased sensitivity of E-MULTCP to fluctuations in RTT measurement. As shown in Table 5.6, the drop in TCP throughput for $\gamma = 0.1$ is smaller than that of $\gamma = 0.2$. Also, $\gamma = 0.1$ results in a lower utilization than $\gamma = 0.2$. Regardless, the stability of the network

[3]Obviously, there are situations in which E-MULTCP ends up with performing similar to one TCP. An example would be E-MULTCP competing for bandwidth with TCP on wired networks. In that case, however, the fairness between E-MULTCP and TCP is reduced to the fairness between TCP connections themselves, and has been well explored in literature.

[4]In this NS simulation, TCP and E-MULTCP share the same route and hence both have the same round trip time.

Table 5.6: Simulation results for fairness between E-MULTCP and TCP.

settings	8 E-MULTCP + 8 TCP			16 TCP	
	ave. thput. (E-MULTCP) (kbps)	ave. thput. (TCP) (kbps)	utilization (%)	ave. thput. (TCP) (kbps)	utilization (%)
p_w=0.01 γ=0.2	432.1	160.3	96	179.8	58
p_w=0.01 γ=0.1	402.3	170.0	93	179.8	58
p_w=0.02 γ=0.2	468.8	107.8	94	120.4	40
p_w=0.02 γ=0.1	446.1	114.3	92	120.4	40

with mixed TCP and E-MULTCP is guaranteed by Theorem 3.3.3.

5.1.6 Slow Start

In practice, the optimal number of connections might be a large number. For instance, in a Gigabit network with undesirable hardware glitches causing packets to drop unnecessarily, sometimes more than 20 connections are needed in order to achieve full utilization of Gigabit links.

However, since E-MULTCP applies inversely increase law on the number of connections upon no congestion, it takes a long time for E-MULTCP to increase the number of connections to the optimal, in the initial starting stage. For instance, if the optimal number of connections is $n^* = 20$, using the analysis in Section 4.2.3, it takes E-MULTCP about 16,000 seconds to open 20 connections, which is about 4.5 hours. This is too conservative.

In order to speed up the initial climbing, we offer an optional technique that is implemented in E-MULTCP, namely slow start. This is named after the famous TCP slow start technique, and in fact uses the same idea as TCP's slow start. With this technique, in the initial climbing stage, E-MULTCP doubles the number of connections each 20 seconds if no congestion is observed, and halves as it otherwise, as follows:

$$n = \begin{cases} 1/2n + \alpha/n, & \text{if } ave\_rtt - rtt\_min > \gamma rtt\_min; \\ 2n, & \text{otherwise.} \end{cases} \tag{5.1}$$

By using slow start technique, the number of connections n in fact increases exponentially in its increasing stage. To reach n^* takes $\tau \log_2 n^*$ seconds, instead of $\frac{\tau}{2\alpha}[(n^*)^2 - 1]$ seconds in normal operation stage.

The initial climbing stage ends when the first sign of congestion is observed, and subsequently E-MULTCP uses the IIMD control law on the number of connections. We have carried out a NS-2 simulations showing performance of slow start implemented in E-MULTCP. The simulation topology is the same as the one shown in Figure 5.1, with $p_w = 0.05$, $B_w = 10M$ bps, $\tau = 10$ seconds, and propagation delay 80 ms. Consequently, $n^* = 24.5$. The simulation results are shown in Figure 5.9. As seen, after applying slow start, the initial climb up time is about 50 seconds, which is close to expected values $\tau \log_2 n^*$. This is much shorter than using IIMD in the initial climbing stage, which is 11,500 seconds.

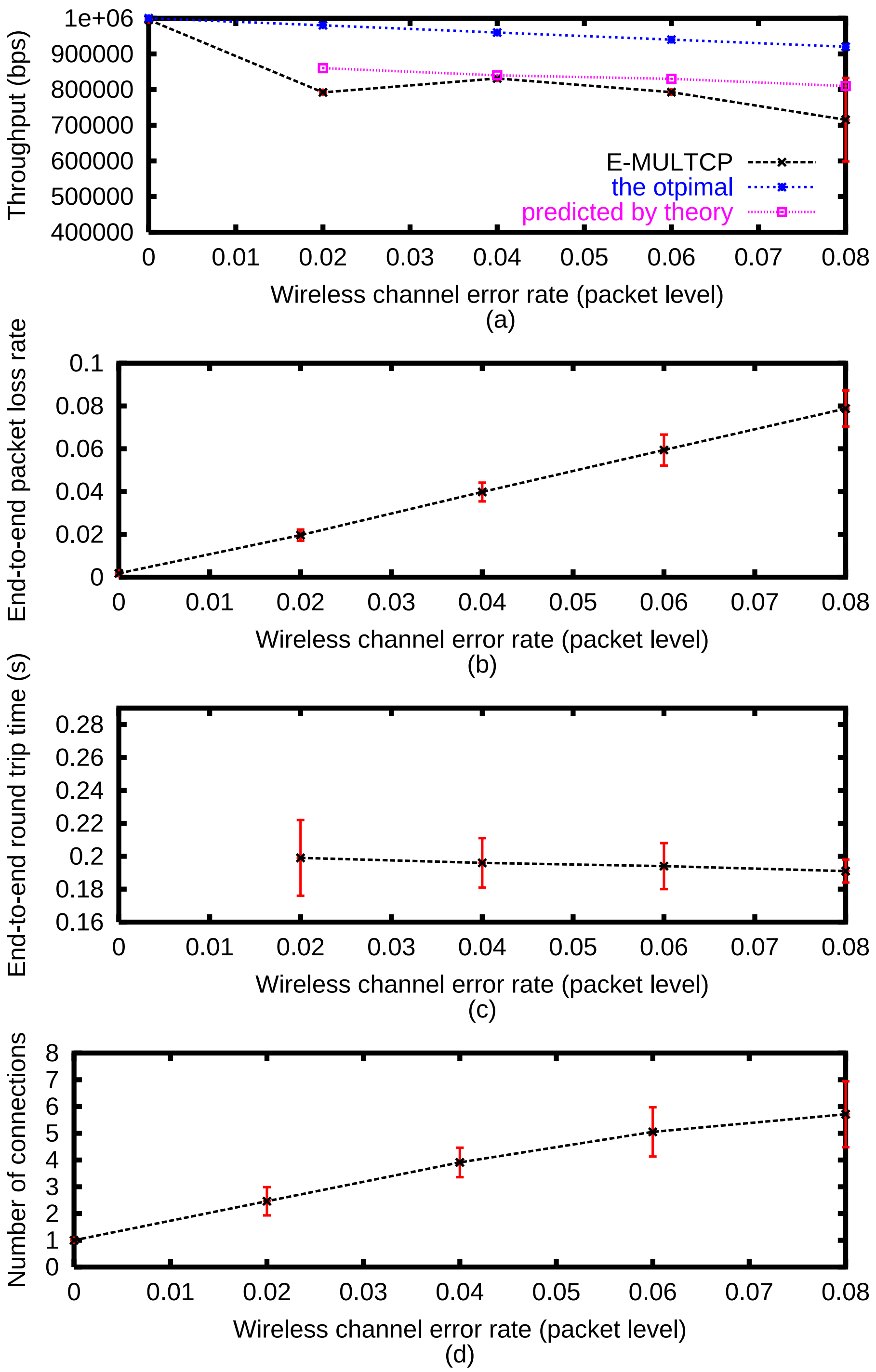


Figure 5.2: NS-2 simulations of E-MULTCP for $B_w = 1\ Mbps$ and $RTT_{min} = 168\ ms$, for $\beta = 0.75$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

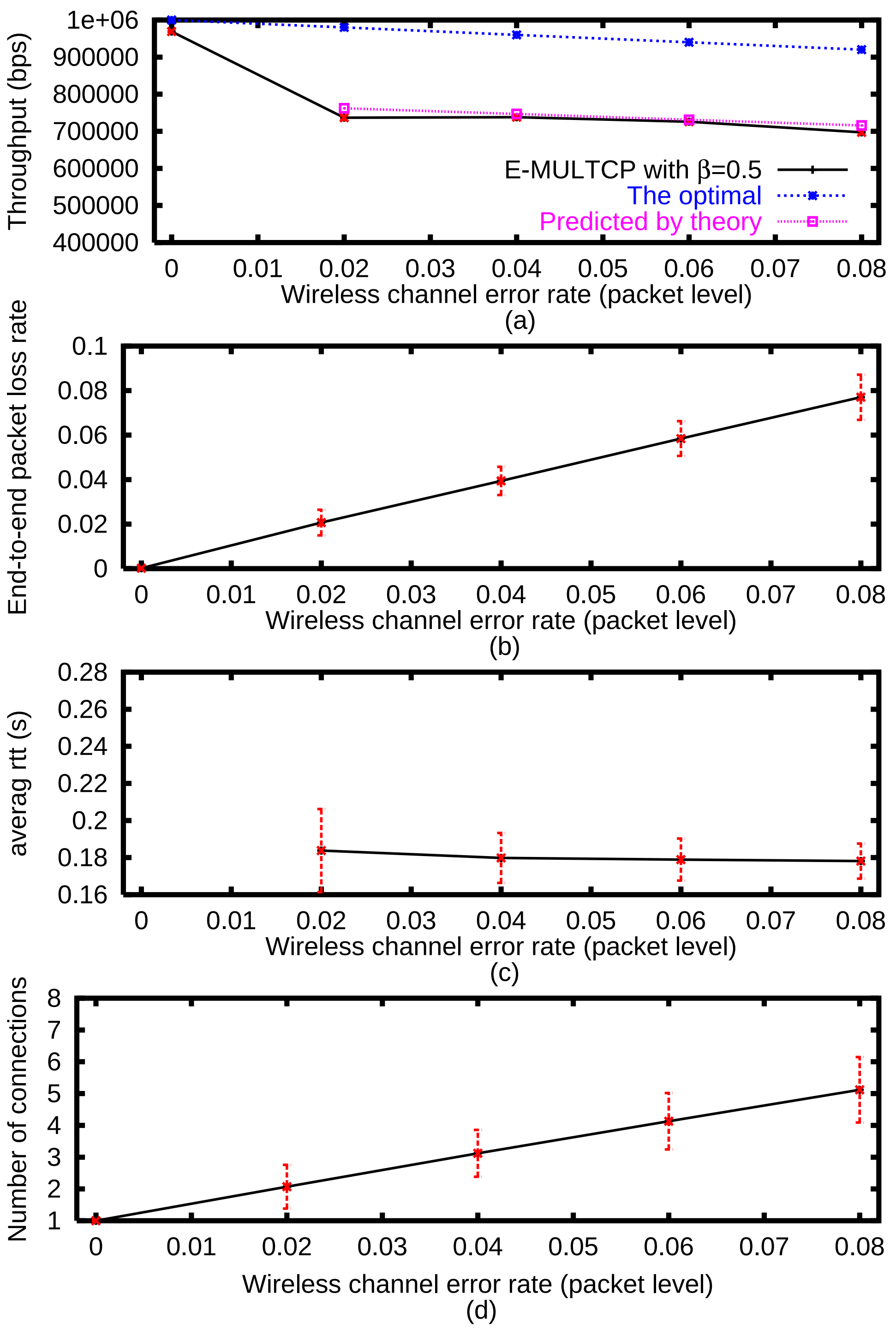


Figure 5.3: NS-2 simulations of E-MULTCP for $B_w = 1$ $Mbps$ and $RTT_{min} = 168$ ms, for $\beta = 0.5$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

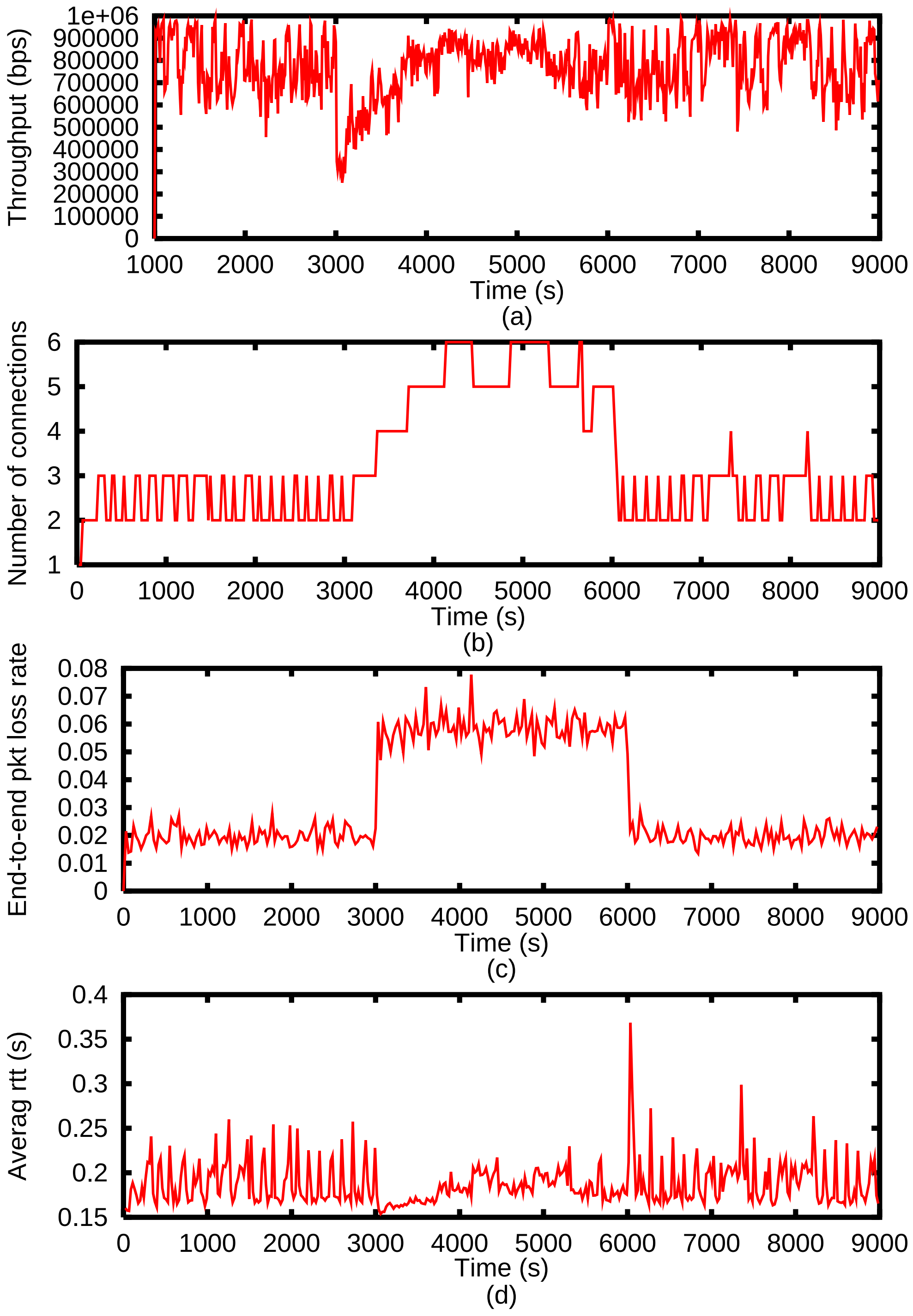


Figure 5.4: NS-2 simulation results of E-MULTCP with $\beta = 0.75$ as p_w changes from 0.02 to 0.06 and back again, for $\alpha = 0.25$; (a) end-to-end round trip time, (b) throughput, (c) number of connections, (d) end-to-end packet loss rate, all as a function of time.

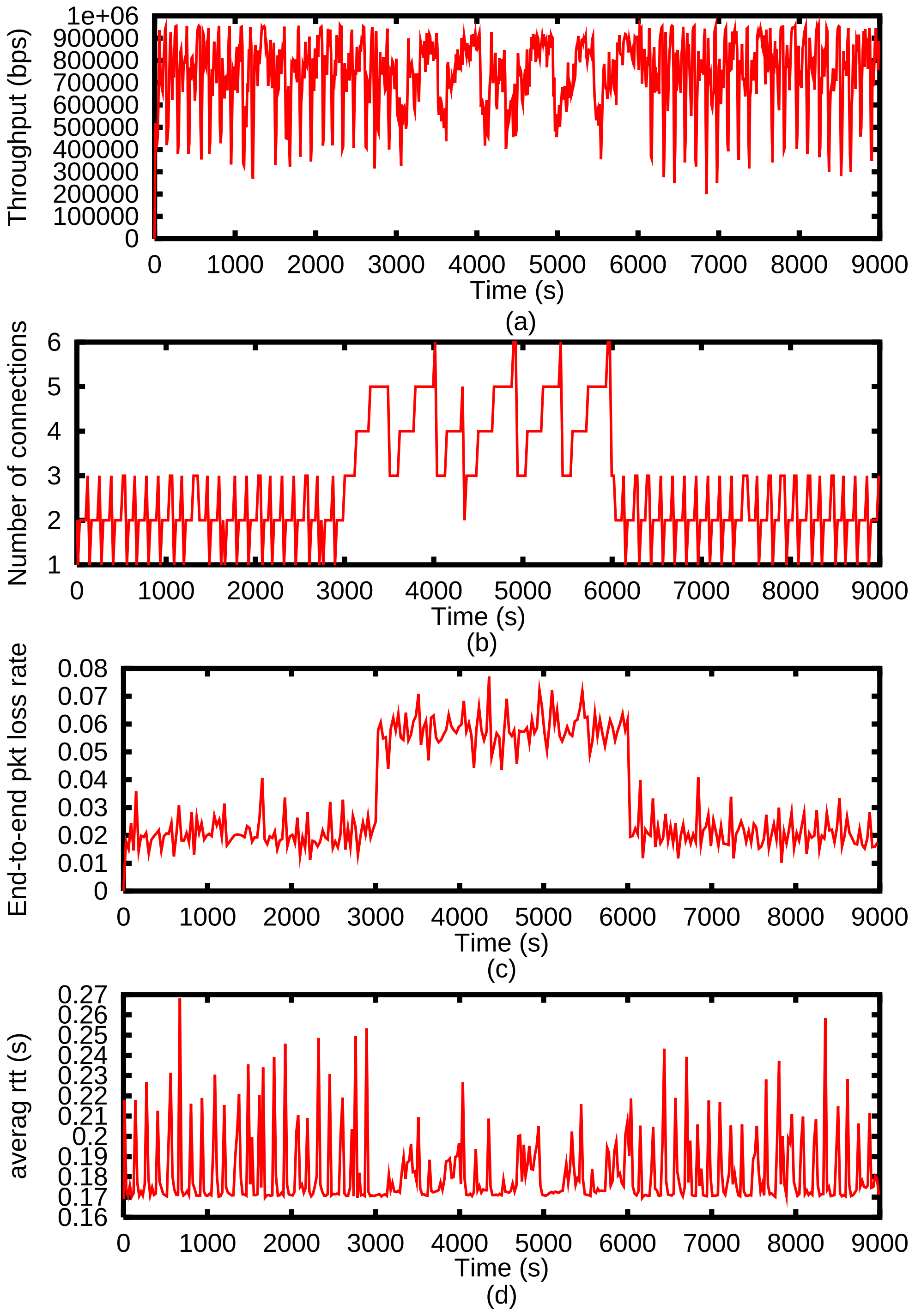


Figure 5.5: NS-2 simulation results of E-MULTCP with $\beta = 0.5$ as p_w changes from 0.02 to 0.06 and back again; (a) end-to-end round trip time, (b) throughput, (c) number of connections, (d) end-to-end packet loss rate, all as a function of time.

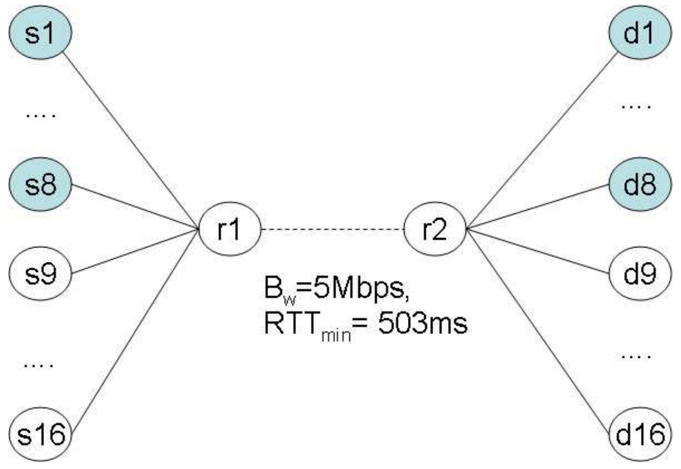
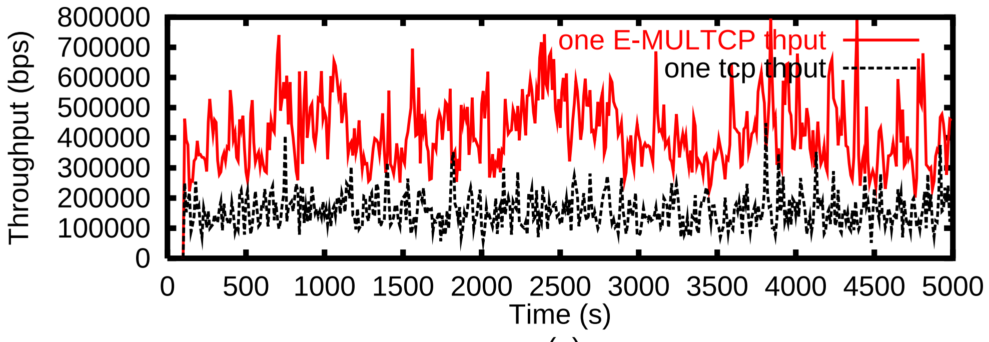
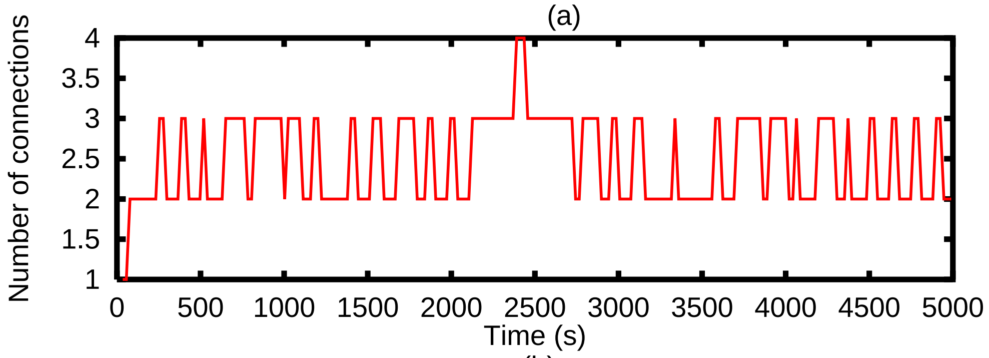


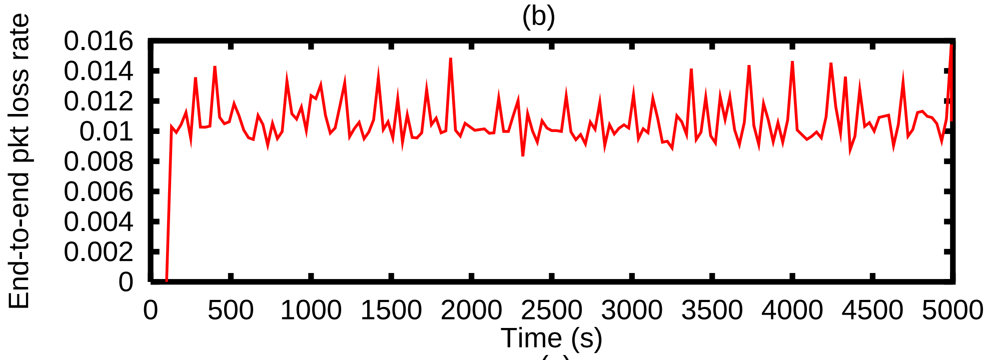
Figure 5.6: The simulation topology for E-MULTCP's fairness evaluation.



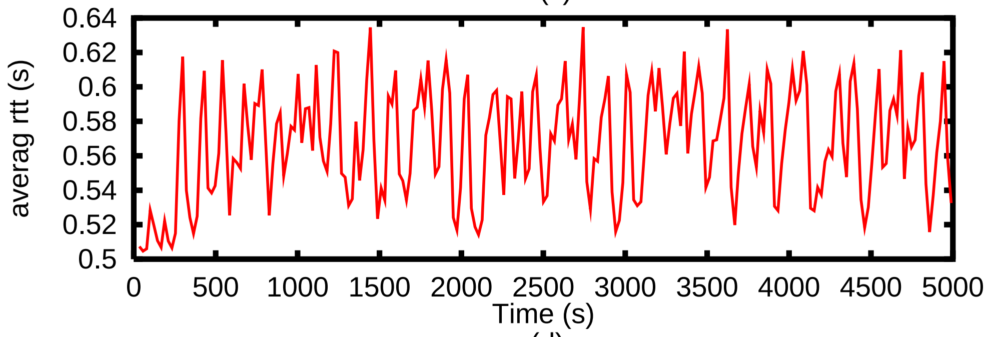
(a)



(b)



(c)



(d)

Figure 5.7: NS-2 simulation results for the case $p_w = 0.01, \gamma = 0.2$ for E-MULTCP+TCP scenario: the dynamics of (a) throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

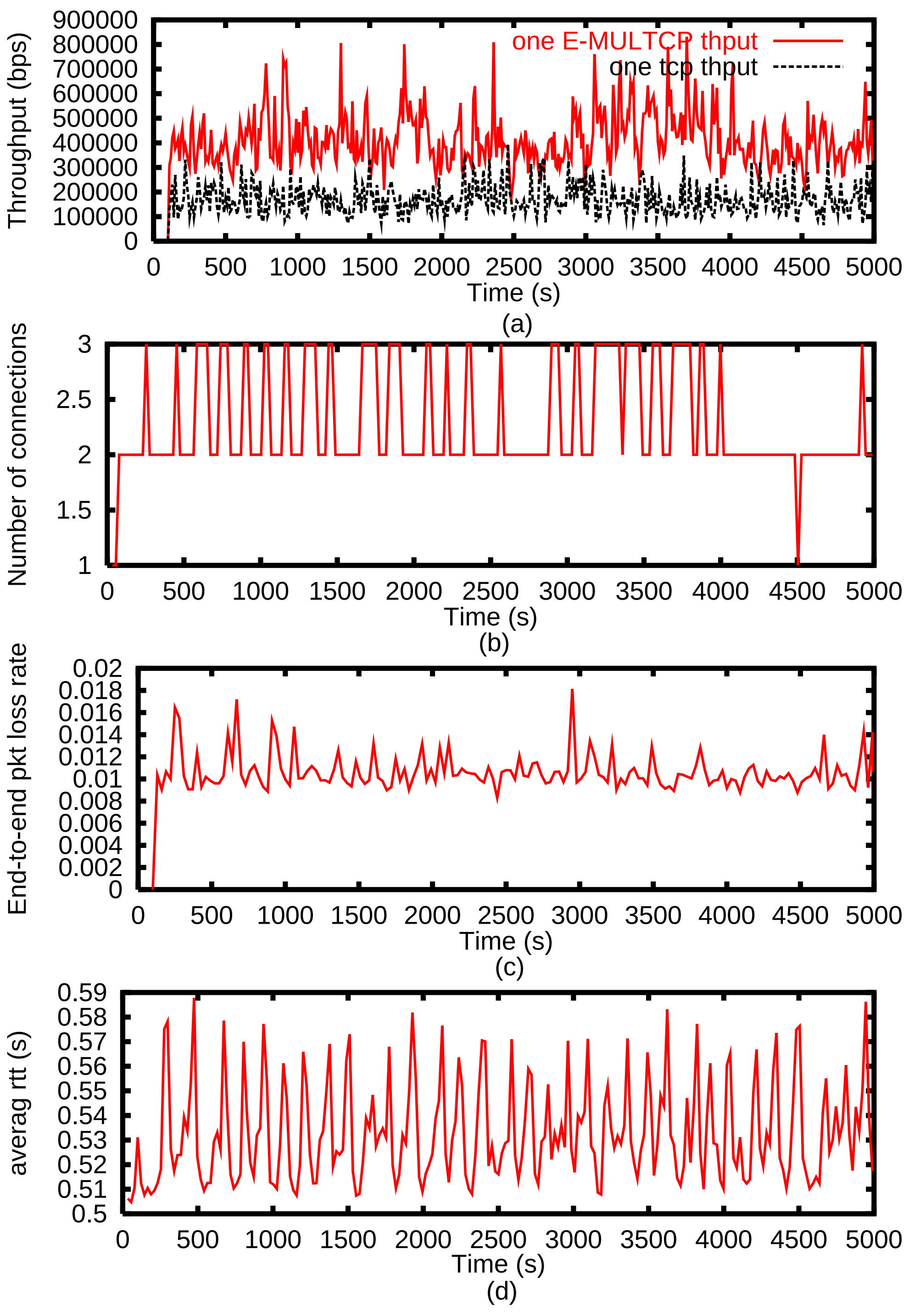


Figure 5.8: NS-2 simulation results for the case $p_w = 0.01, \gamma = 0.1$ for E-MULTCP+TCP scenario: (a) throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

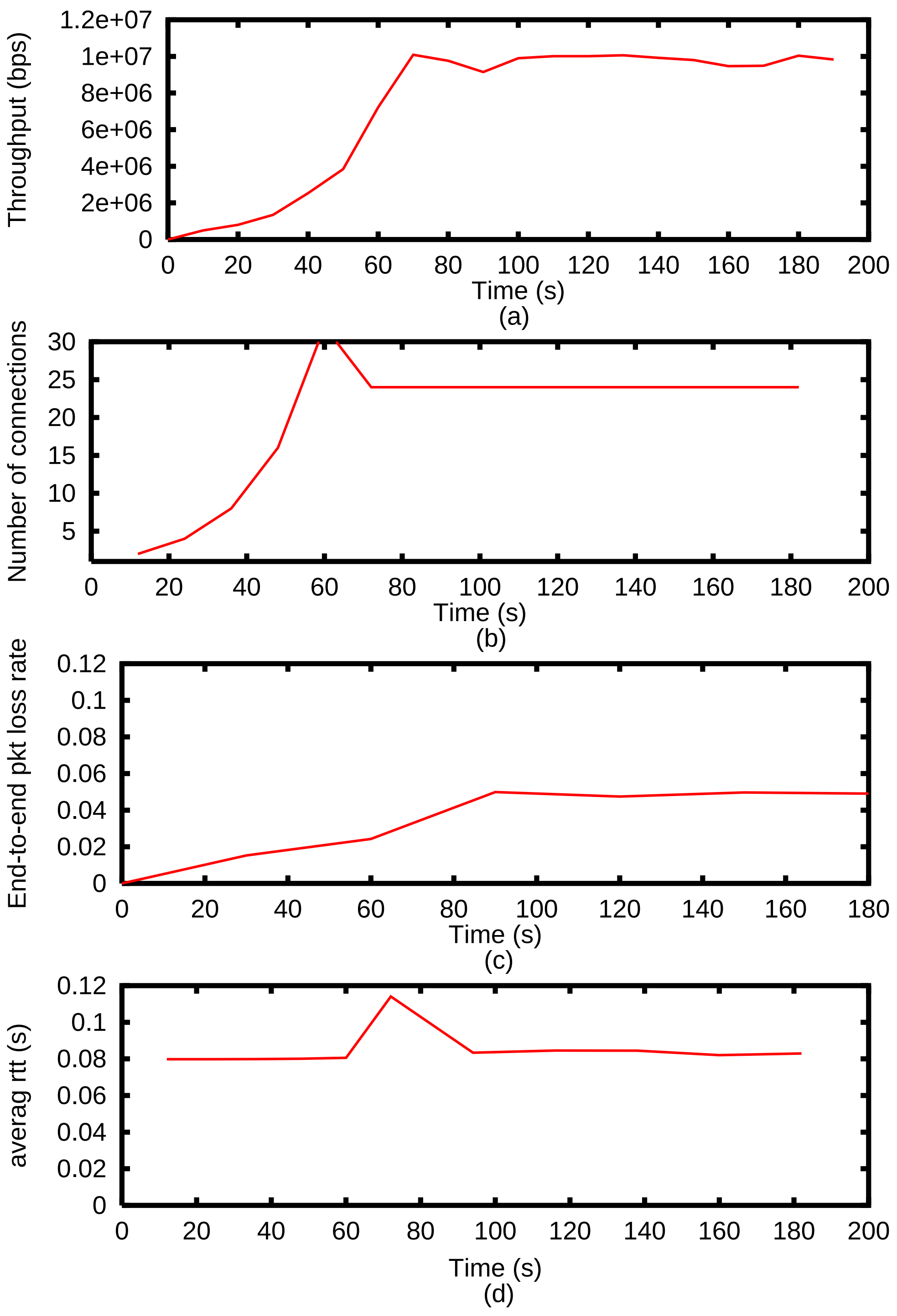


Figure 5.9: NS-2 simulation results for the case $p_w = 0.05$ for E-MULTCP with slow start: (a) throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

5.2 E-MULTFRC: NS-2 Simulations and 1xRTT Wireless Experiments

In this section, we carry out NS-2 [47] simulations and actual experiments over Verizon Wireless 1xRTT CDMA data network to evaluate the performance of E-MULTFRC.

5.2.1 Setup

The topology used in E-MULTFRC simulations is the same as E-MULTCP, as shown in Figure 5.1. The sender denoted by s, and the receiver denoted by r, both run E-MULTFRC at the application layer. Other simulation settings are the same as those in E-MULTCP simulations as well.

Similarly, in order to evaluate E-MULTFRC's performance in the presence of wireless channel errors, we examine the same three issues as in E-MULTCP simulations; first, how E-MULTFRC performs in terms of average throughput, average round trip time, and packet loss rate, as a function of p_w. Second, whether the number of connections is stable. Third, whether or not an E-MULTFRC application can fairly share with an application using one TFRC or one TCP connection.

For the actual experiments over 1xRTT, we stream from a desktop connected to Internet via 100 Mbps Ethernet in eecs.berkeley.edu domain, to a notebook connected to Internet via Verizon Wireless 1xRTT CDMA data network. The streaming takes 30 minutes.

5.2.2 Performance Characterization of E-MULTFRC

We use the same set of parameters α, β, γ, and τ as E-MULTCP: $\alpha = 0.25, \beta = 0.75$, $\gamma = 0.2$, and $\tau = 20$ seconds.

We simulate the E-MULTFRC system to stream for 9000 seconds, and compute the average throughput and packet loss rate for p_w =0.0, 0.02, 0.04, 0.06 and 0.08, and compare them to the optimal, i.e. $B_w(1 - p_w)$ for each p_w. The results for $B_w = 1$ $Mbps$ and $RTT_{min} = 168$ ms are shown in Figure 5.10. As seen, E-MULTFRC has similar performance as compared to E-MULTCP. The throughput is within 25% of the optimal. The average round trip time is within 20% of RTT_{min}, the same as the expected range i.e. γRTT_{min}. The packet loss rate is almost identical to the optimal, i.e. a line of slope one as a function of wireless channel error rate. As expected, the average number of connections increases with wireless channel error rate, p_w. [5]

Considering the throughput plot in Figure 5.10, we notice that for some values of p_w, there is a significant difference between the actual and optimal throughput. This is the "quantization effect" in E-MULTCP's simulations. One way to alleviate this problem is to increase γ in order to tolerate larger queuing delay and hence absorb throughput fluctuations, at the expense of being less responsive. Another alternative is to use smaller packet size in order to reduce the "quantization effect" at the expense of (a) lower transmission efficiency and (b) the slower rate of convergence to the optimal number of connections.

To examine the dynamics of E-MULTFRC system, we show throughput, packet loss rate, and the number of connections as a function of time for $p_w = 0.04$ in Figure 5.11. As seen, the throughput and the number of connections are quite stable; as expected, packet loss rate is around 0.04 and round trip time is low, and is in agreement with the results corresponding to $p_w = 0.04$ in Figure 5.10. Similar results are obtained for other values of p_w.

In order to examine E-MULTFRC's performance as a function of p_w, as well as the dynamics of E-MULTFRC, we use E-MULTFRC with p_w initially set at 0.02. Then at 3000^{th} second, p_w is switched to 0.06, and at 6000^{th} second switched back to 0.02. Here, we artificially change p_w to see how E-MULTFRC adapts to the change in p_w. The throughput, packet loss rate, round trip time and the number of opened connections are shown in Figure 5.12. As seen, the number of connections varies from around 2-3 to around 5 as p_w switches from 0.02 to 0.06.

5.2.3 Experimental Results for E-MULTFRC

Similar experiments are carried out on Verizon Wireless 1xRTT CDMA data network in Spring of 2005. At that time, the 1xRTT CDMA data network is found to have the highest average available bandwidth averaged over 30 minutes to be between 80 kbps to 97 kbps, by using UDP flooding. In our experiments, we stream for 30 minutes from a desktop on wired network in eecs.berkeley.edu domain to a laptop connected via 1xRTT CDMA modem using E-MULTFRC and TFRC. The results are shown in Table 5.7 for packet size

[5]Note the round trip time for $p_w = 0$ is not shown in Figure 5.10 because it represents the channel error free case in which E-MULTFRC reduces to one TFRC connection.

of 1460 bytes. As seen, on average E-MULTFRC opens up 1.9 connections, and results in 65% higher throughput, at the expense of a larger round trip time, and higher packet loss rate. Compared to E-MULTCP's results in Table 5.2, we can see they have similar performance.

Table 5.8 shows packet loss details of E-MULTFRC for one of the 30 minutes long experiments over 1xRTT. As expected, both the packet loss rate and burstiness of the loss increase as the number of connections increases.

Table 5.7: Actual experimental results for an E-MULTFRC system over 1xRTT CDMA.

scheme	**throughput (kbps)**	**rtt (ms)**	**packet loss rate**	**ave. # of conn.**	**throughput improvement over one TFRC (%)**
one TFRC	54	1624	0.031	N/A	N/A
E-MULTFRC	89	2767	0.041	1.9	65

Table 5.8: Packet loss details of E-MULTFRC

# of conn. opened	**% of time**	**pkt loss rate**	**avg. burst error length**	**std. dev.**	**max. burst length**
one	24.1	0.016	2.78	3.26	7
two	61.1	0.045	2.43	3.33	10
three	14.8	0.076	3.45	8.93	11

5.2.4 Performance Comparison of E-MULTFRC and TFRC

We also carry out simulations for the topology shown in Figure 5.13 with the following settings: $C_1 = 1\ Mbps, C_2 = 5\ Mbps$, $S = 760$ bytes, users 1, 2, 3 have round trip propagation delays of 691 ms, 651 ms, and 69 ms respectively. We use two sets of (p_w^1, p_w^2). The first set is $(0, 0)$, representing the wired scenario; the second set is $(0.02, 0.01)$, representing the wireless scenario. The wireless link is modeled as a wired link with an exponential random packet loss model.

In simulations, we stream 9000 seconds video from s_i to r_i for user i, $i = 1, 2, 3$, using E-MULTFRC scheme, or only one TFRC. The simulation results for TFRC are shown in Figure 5.14 for wireless scenario with $p_w^1 = 0.02$ and $p_w^2 = 0.01$. Those for E-MULTFRC are shown in Figure 5.15 for wireless scenario with $p_w^1 = 0.02$ and $p_w^2 = 0.01$, and in Figure 5.16 wired scenario with $p_w^1 = p_w^2 = 0$. Based on this setting, the minimum end-to-end packet loss rates for user 2 and 3 are p_w^1 and p_w^2 respectively, i.e. 0.02 and 0.01 respectively in the wireless scenario, and simply 0 in the wired scenario. The minimum end-to-end packet loss rate for user 1 can be easily computed as $1 - (1 - p_w^1)(1 - p_w^2)$, which is 0.0297 for the wireless scenario, and 0 in the wired scenario. The minimum round trip times for users 1, 2, and 3 in the wireless scenario are merely the propagation delays, i.e. 691 ms, 651 ms, and 69 ms respectively.

The results in Figures 5.14, 5.15 and 5.16 include throughput measured every 10 seconds, number of connections, end-to-end packet loss rate measured every 30 seconds and average *rtt* measured every 20 seconds. Several observations can be drawn from the Figures. First, as predicted from the analysis in Section 3.3, when $p_w^1 = p_w^2 = 0$, as shown in Figure 5.16, E-MULTFRC opens one connection, thereby being reduced to one TFRC in the wired network case. Second, comparing Figures 5.15(a) and 5.16(a), E-MULTFRC can achieve almost the same stationary sending rate in wireless and wired networks. Moreover, as seen from Figure 5.15(c), the end-to-end packet loss rates for users 1, 2, and 3 are about 0.03, 0.02, and 0.01, i.e. the minimum values for the wireless scenario, as discussed earlier. In addition, as shown in Figure 5.16(c), the packet loss rate for all 3 users in the wired scenario is zero, which is in agreement with the minimum values discussed earlier. Since at time zero, initially two connections are opened before the number of connections eventually converges to one in Figure 5.16(c), there is a spike in end-to-end packet loss rate at time zero, which eventually converges to zero. Similarly, as seen from Figure 5.15(d), the round trip times for users 1, 2, and 3 in the wireless scenario are around the minimum values, as discussed above. These are simply the results of E-MULTFRC pursuing the boundary between full utilization and underutilization. Fourth, comparing Figures 5.14(a) and 5.15(a), E-MULTFRC achieves a higher throughput than one TFRC scheme in wireless networks, at the expense of minor modification to the application layer. This confirms our analysis in Section 3.3, and shows E-MULTFRC can achieve good performance in the wireless scenario.

5.2.5 Fairness between E-MULTFRC and TCP

To investigate the fairness of E-MULTFRC, we carry out NS-2 simulations based on the "dumbbell" topology shown in Figure 5.6. Senders are denoted by $si, i = 1, \ldots, 16$, and receivers are denoted by $di, i = 1 \ldots, 16$. We investigate two types of fairness: the inter-protocol fairness between E-MULTFRC and TCP, and the intra-protocol fairness within E-MULTFRC.

The intra-protocol fairness is defined as the fairness between E-MULTFRC flows. In our simulations, we run E-MULTFRC on all 16 sender-receiver pairs shown in Figure 5.6 for 5000 seconds, and compare their throughput. E-MULTFRC is said to be intra-protocol fair if all receivers achieve the same throughput. The fairness ratios for $p_w = 0.01$ and $p_w = 0.04$ are shown in Table 5.9. The fairness ratio is defined as receivers' throughput divided by the average throughput; the closer to one, the more fair the E-MULTFRC system is. As seen, the fairness ratio is fairly close to one, indicating E-MULTFRC flows are fair to each other, at least in this simulation setting. The bandwidth utilization ratios are 96% for $p_w = 0.01$ and 98% for $p_w = 0.04$.

Table 5.9: Simulation results for intra-protocol fairness of E-MULTFRC.

receiver	fairness ratio p_w=0.01	fairness ratio p_w=0.04	receiver	fairness ratio p_w=0.01	fairness ratio p_w=0.04
d1	1.08	1.03	d9	1.04	0.98
d2	0.99	1.01	d10	0.98	0.98
d3	1.05	1.03	d11	1.03	1.02
d4	1.01	1.08	d12	1.03	0.99
d5	0.91	0.98	d13	1.00	0.97
d6	0.92	0.98	d14	0.90	1.00
d7	1.02	1.04	d15	1.02	1.01
d8	0.98	1.01	d16	1.00	0.94

The inter-protocol fairness is defined as the fairness between E-MULTFRC and TCP. In our simulations, we run E-MULTFRC on the first 8 sender-receiver pairs, i.e. $(si, di), i = 1, \ldots, 8$, and TCP on the remaining 8 sender-receiver pairs shown in Figure 5.6; each session lasts 5000 seconds, and we compare their throughput for $p_w = 0.01$ and $p_w = 0.02$. Under the simulation settings, each E-MULTFRC consumes more bandwidth than one TCP under full utilization. This is because in this case, the wireless channel error rate is large enough to make the number of virtual connections of each E-MULTFRC to be larger than one. Hence, it is meaningless to define the fairness between E-MULTFRC and TCP as having the same throughput.[6] As such, in our simulations, we define E-MULTFRC to be fair

[6]Obviously, there are situations in which E-MULTFRC ends up with performing similar to one TFRC. An example would be E-MULTFRC competing for bandwidth with TCP on wired networks. In that case,

to TCP if it does not result in a decrease in TCP's throughput as compared to the case where E-MULTFRC is absent. Specifically for our simulations, it implies TCP retains the same throughput whether or not it coexists with E-MULTFRC under the same network setting. The throughput of E-MULTFRC and TCP, as well as the total bandwidth utilization ratios for the setup shown in Figure 5.6, are shown in Table 5.10 for two scenarios: (a) 8 E-MULTFRC coexisting with 8 TCP connections, (b) 16 TCP connections. Figures 5.17 and 5.18 also show the dynamics of throughput, packet loss rate, RTT, and the number of virtual connections n for $\gamma = 0.2$ and $\gamma = 0.1$ respectively. Comparing E-MULTFRC+TCP with TCP-alone in Table 5.10, we see the former achieves a much higher utilization of the wireless bandwidth at the expense of lower TCP throughput. A careful examination of Figure 5.17 reveals that this throughput drop is caused by the higher RTT experienced by TCP connections in the E-MULTFRC+TCP scenario as compared with TCP-alone scenario. For example, for $p_w = 0.01$ and $\gamma = 0.2$, E-MULTFRC+TCP experiences around 0.55 seconds RTT, while TCP-alone only experiences 0.5 seconds RTT, i.e. the propagation delay[7]. As TCP's throughput is known to be inversely proportional to RTT, the 10% increase in the RTT of TCP connections roughly explains the 13% decrease in the TCP's throughput shown in first row in Table 5.10.

Table 5.10: Simulation results for fairness between E-MULTFRC and TCP.

settings	8 E-MULTFRC + 8 TCP			16 TCP	
	ave. thput. (E-MULTFRC) (kbps)	ave. thput. (TCP) (kbps)	utili-zation (%)	ave. thput. (TCP) (kbps)	utili-zation (%)
p_w=0.01 γ=0.2	436.59	176.67	99	200.168	65
p_w=0.01 γ=0.1	408.84	188.40	97	200.168	65
p_w=0.02 γ=0.2	472.52	127.81	98	139.674	46
p_w=0.02 γ=0.1	444.90	134.64	93	139.674	46

This increase in the RTT experienced by TCP is, by design, a consequence of

however, the fairness between E-MULTFRC and TCP is reduced to the fairness between TFRC and TCP, and has been well explored in [2].

[7] In this NS simulation, TCP and E-MULTFRC share the same route and hence both have the same round trip time.

E-MULTFRC controlling n according to Equation (4.2). As n is only decreased after the queuing delay exceeds the threshold γrtt_min, round trip time is increased when E-MULTFRC increases n to achieve full utilization. One way to address this problem is to use a smaller value for γ, in order to reduce the increase in the RTT, and hence minimize the TCP's throughput drop. However, similar to the arguments in E-MULTCP's simulations in Section 5.1.3, for smaller values of γ also result in lower bandwidth utilization due to increased sensitivity of E-MULTFRC to fluctuations in measured RTT. As shown in Table 5.10, $\gamma = 0.1$ results in a smaller negative change in the TCP's throughput, than $\gamma = 0.2$.

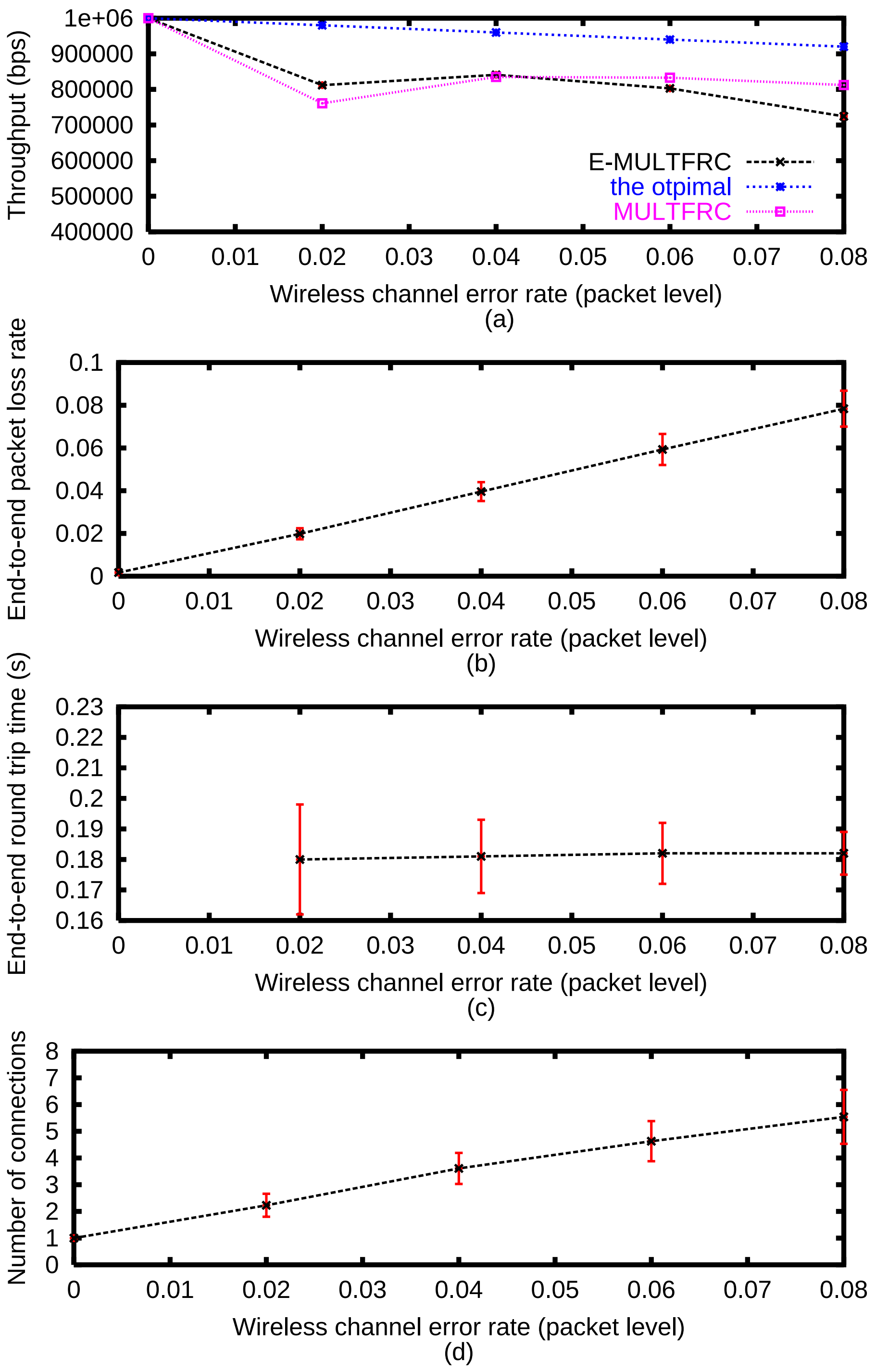


Figure 5.10: NS-2 simulations of E-MULTFRC for $B_w = 1\ Mbps$ and $RTT_{min} = 168\ ms$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

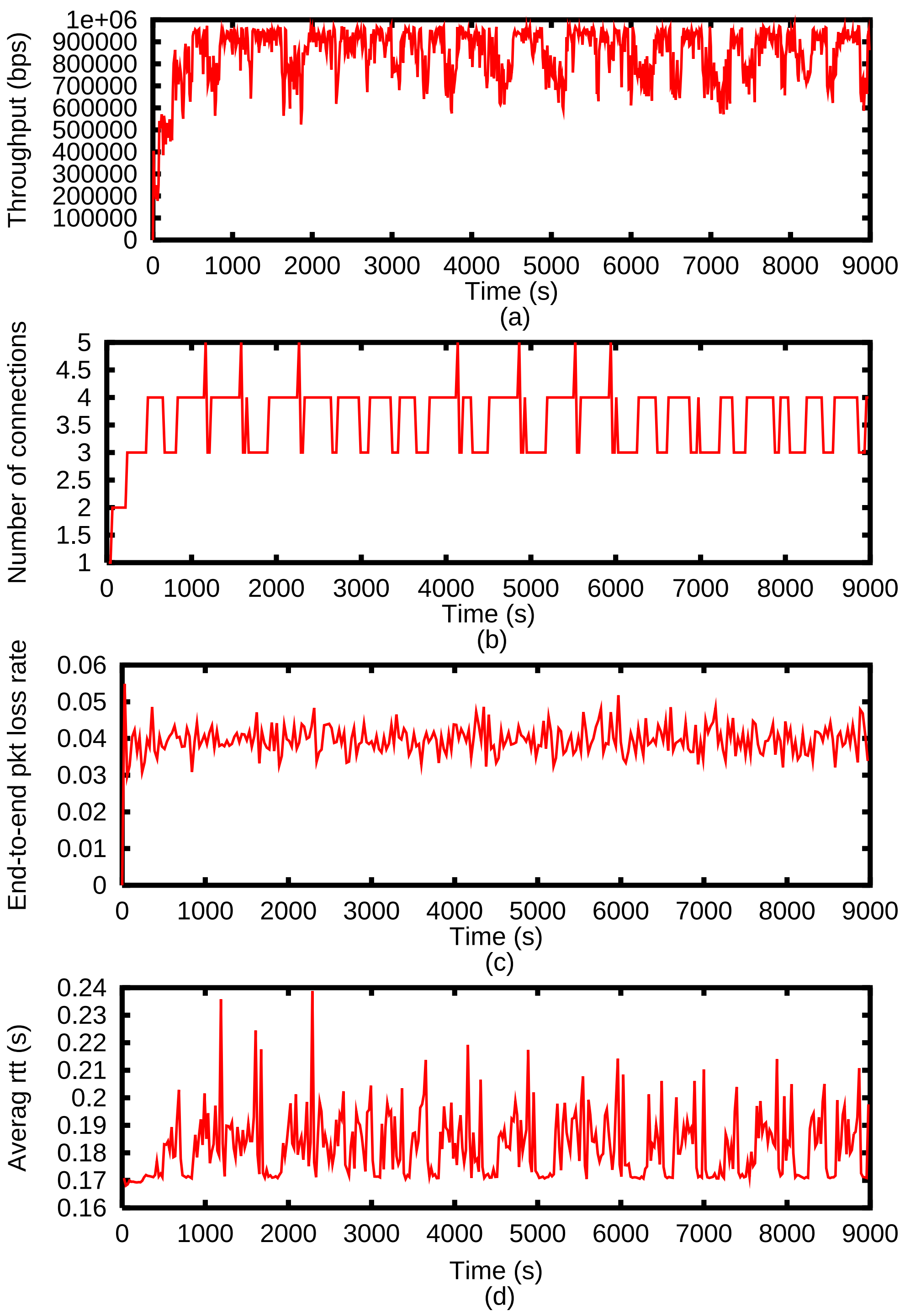


Figure 5.11: NS-2 simulations of E-MULTFRC for $B_w = 1\,Mbps$ and $p_w = 0.04$; (a) end-to-end round trip time, (b) throughput, (c) end-to-end packet loss rate, (d) number of connections, all as a function of time.

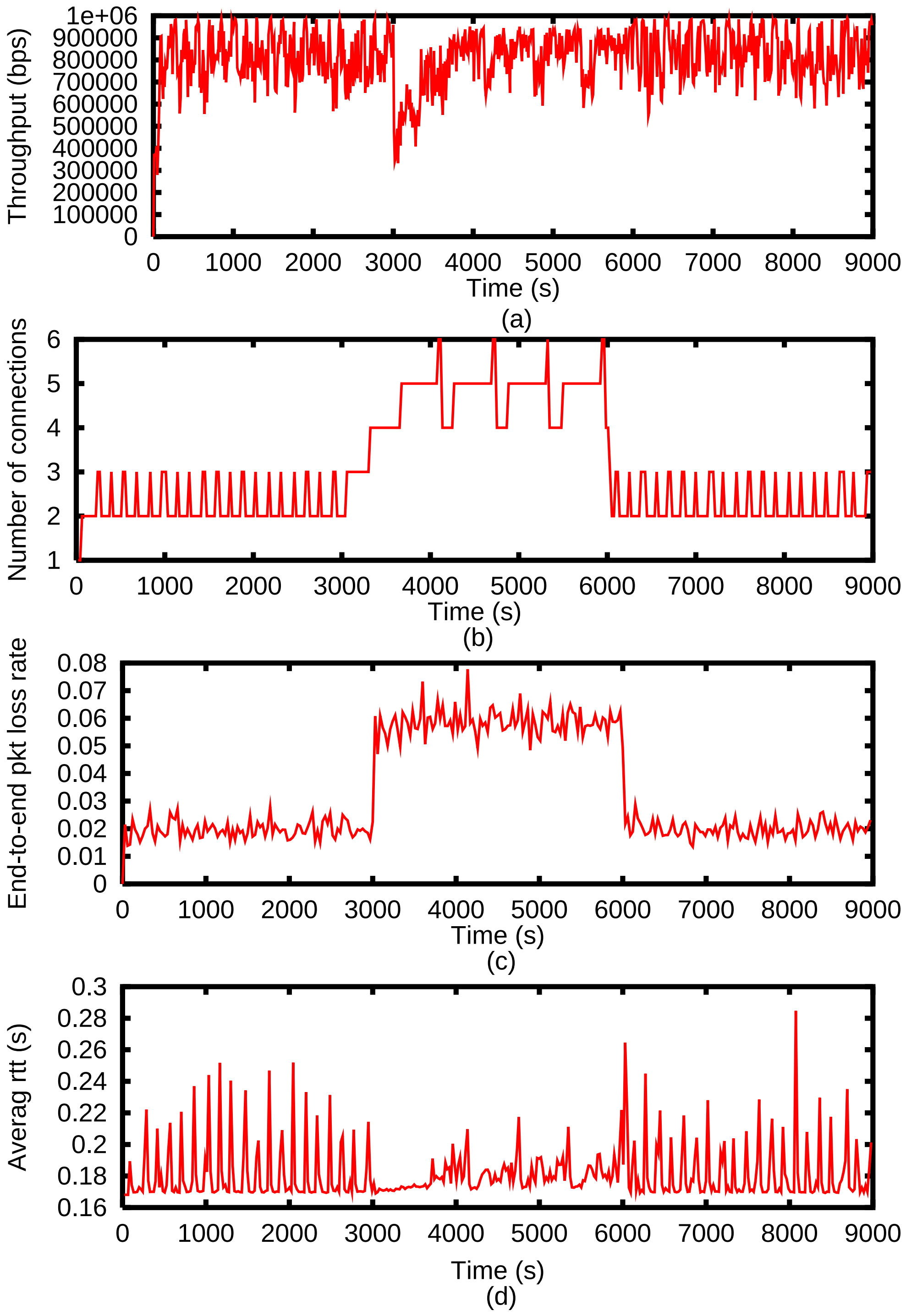


Figure 5.12: NS-2 simulation results of E-MULTFRC as p_w changes from 0.02 to 0.06 and back again; (a) end-to-end round trip time, (b) throughput, (c) numbers of connections, (d) end-to-end packet loss rate, all as a function of time.

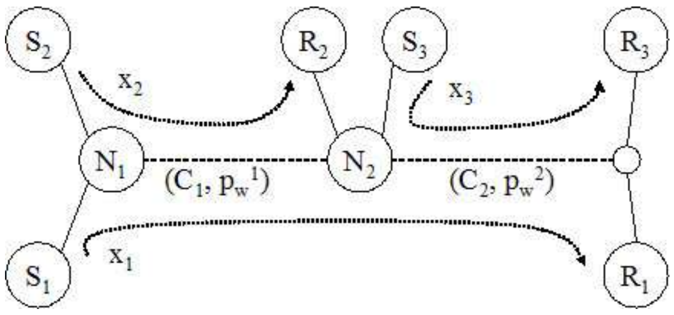


Figure 5.13: Simulation topology.

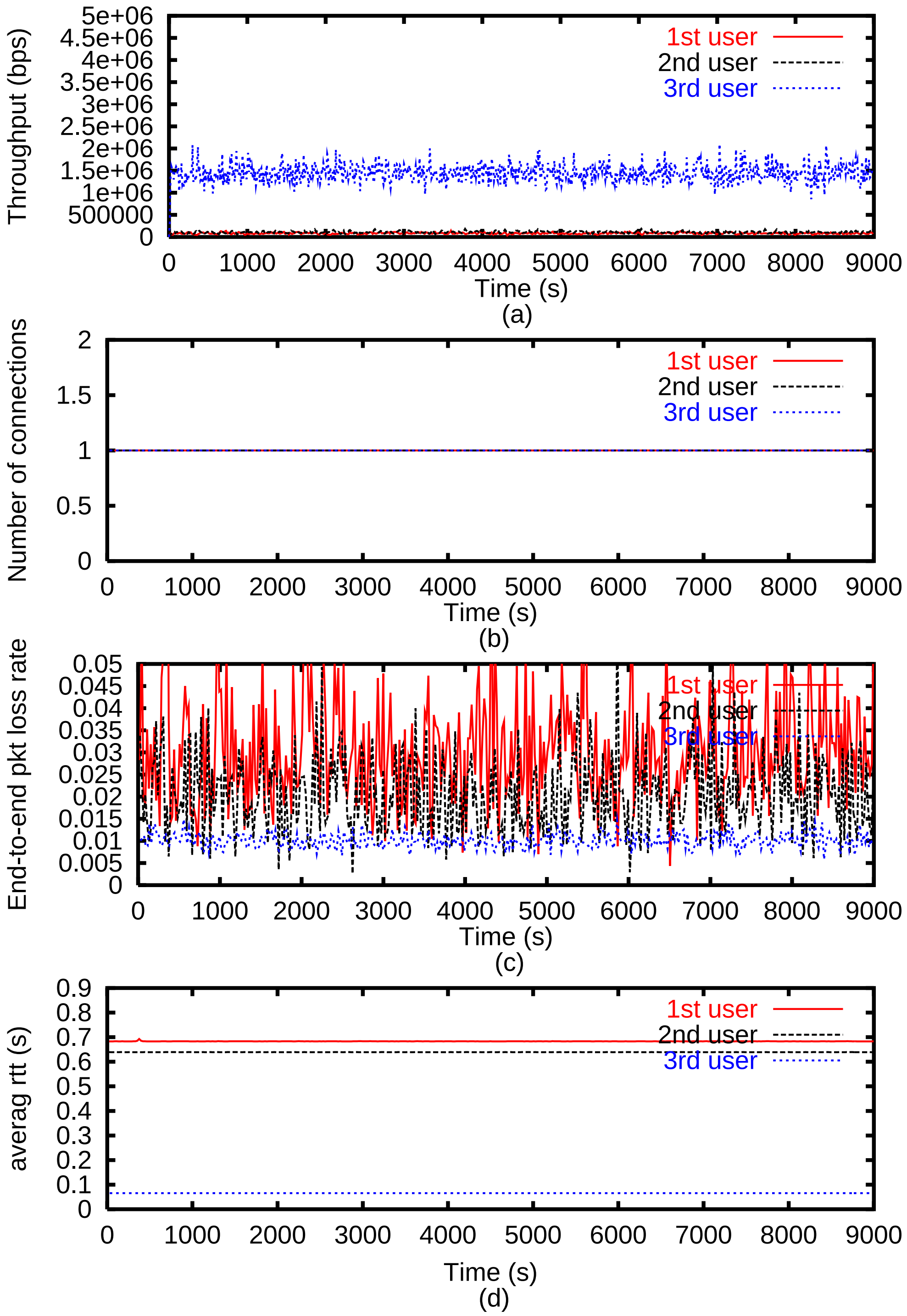


Figure 5.14: NS-2 simulations of one TFRC for the topology shown in Figure 5.13 with $p_w^1 = 0.02, p_w^2 = 0.01$: (a) throughput, (b) number of connections, (c) end-to-end packet loss rate and (d) average rtt.

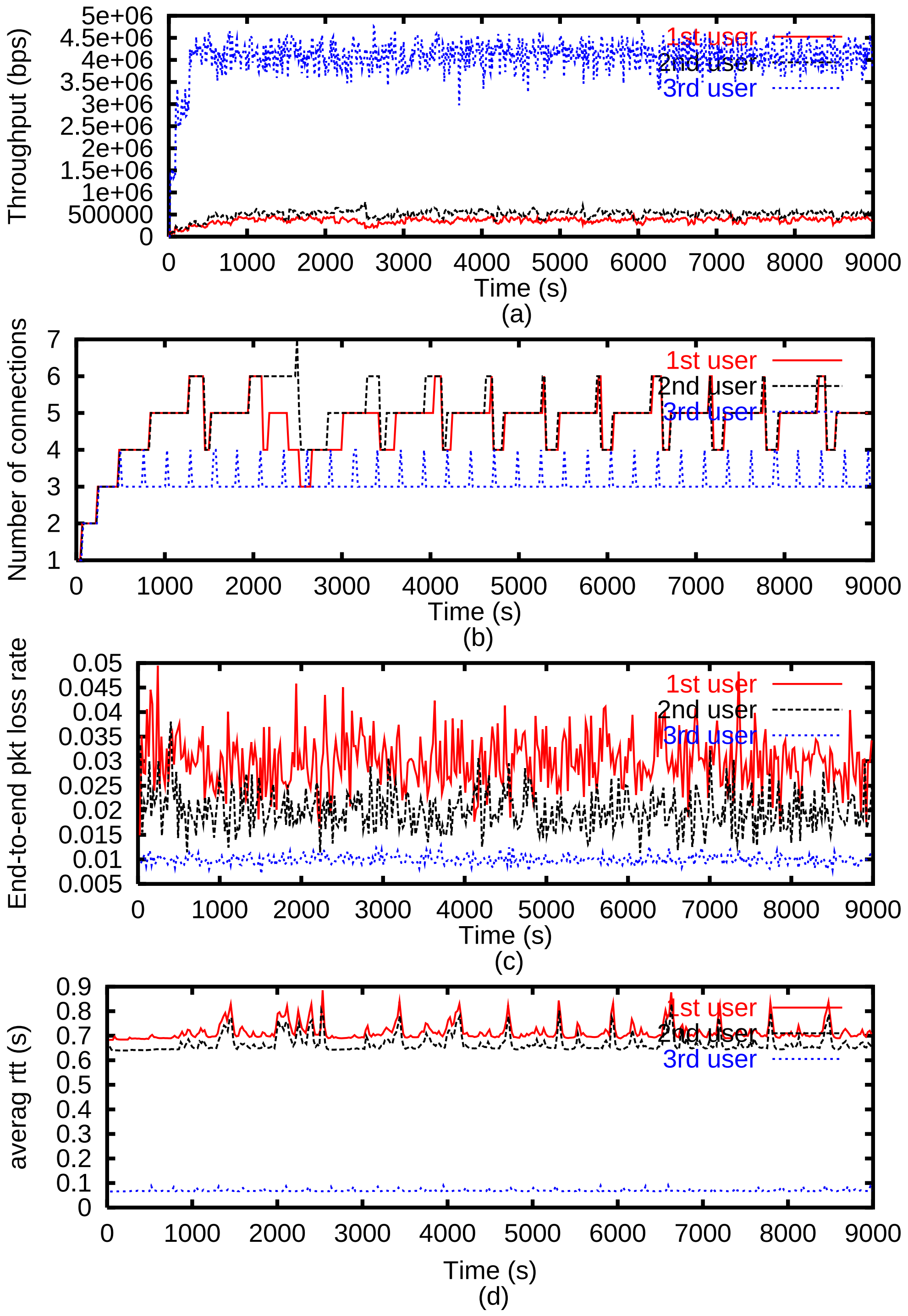


Figure 5.15: NS-2 simulations of E-MULTFRC for the topology shown in Figure 5.13 with $p_w^1 = 0.02, p_w^2 = 0.01$: (a) throughput, (b) number of connections, (c) end-to-end packet loss rate and (d) average rtt.

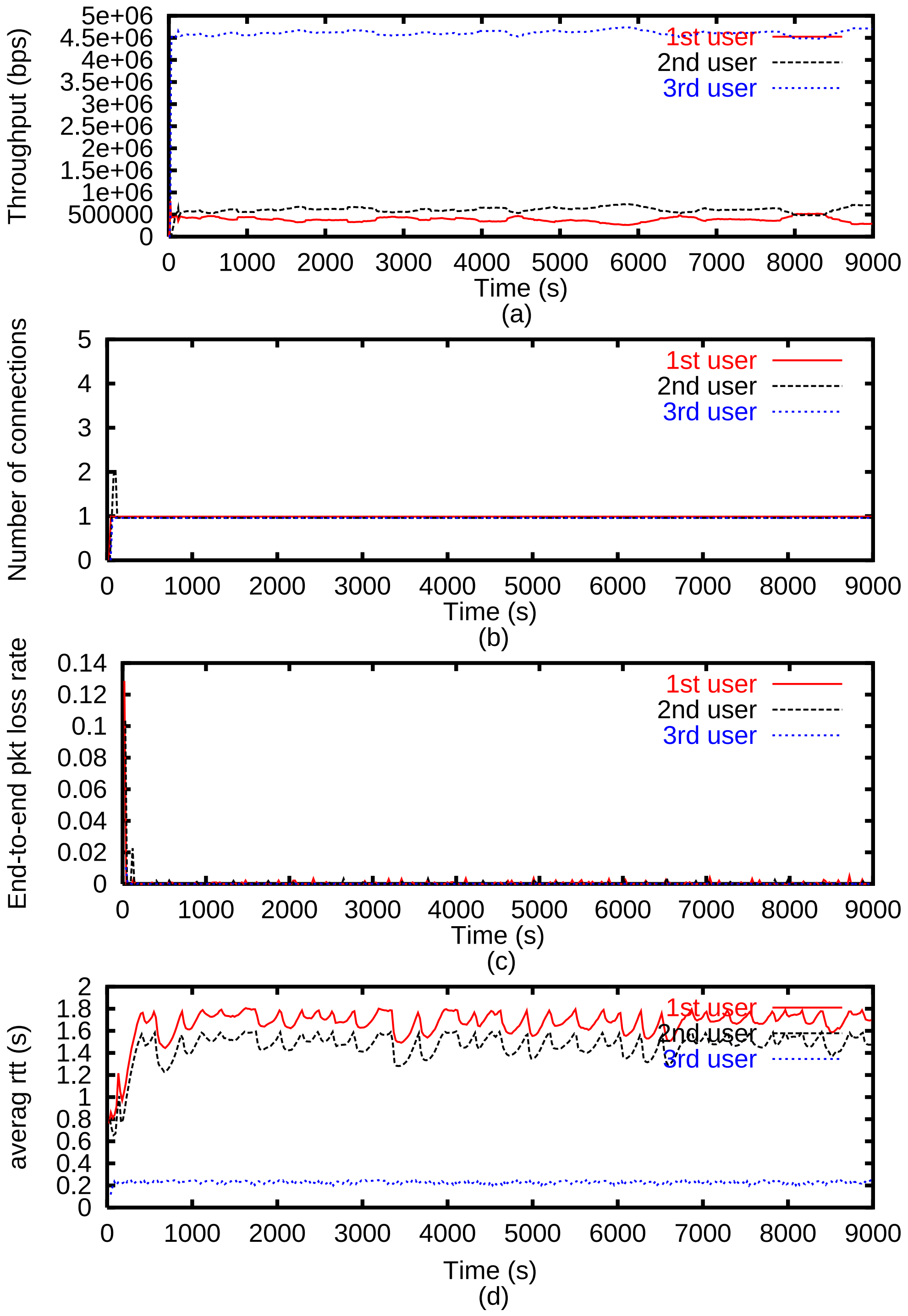


Figure 5.16: NS-2 simulations of E-MULTFRC for the topology shown in Figure 5.13 with $p_w^1 = 0.00, p_w^2 = 0.00$: (a) throughput, (b) number of connections, (c) end-to-end packet loss rate and (d) average rtt.

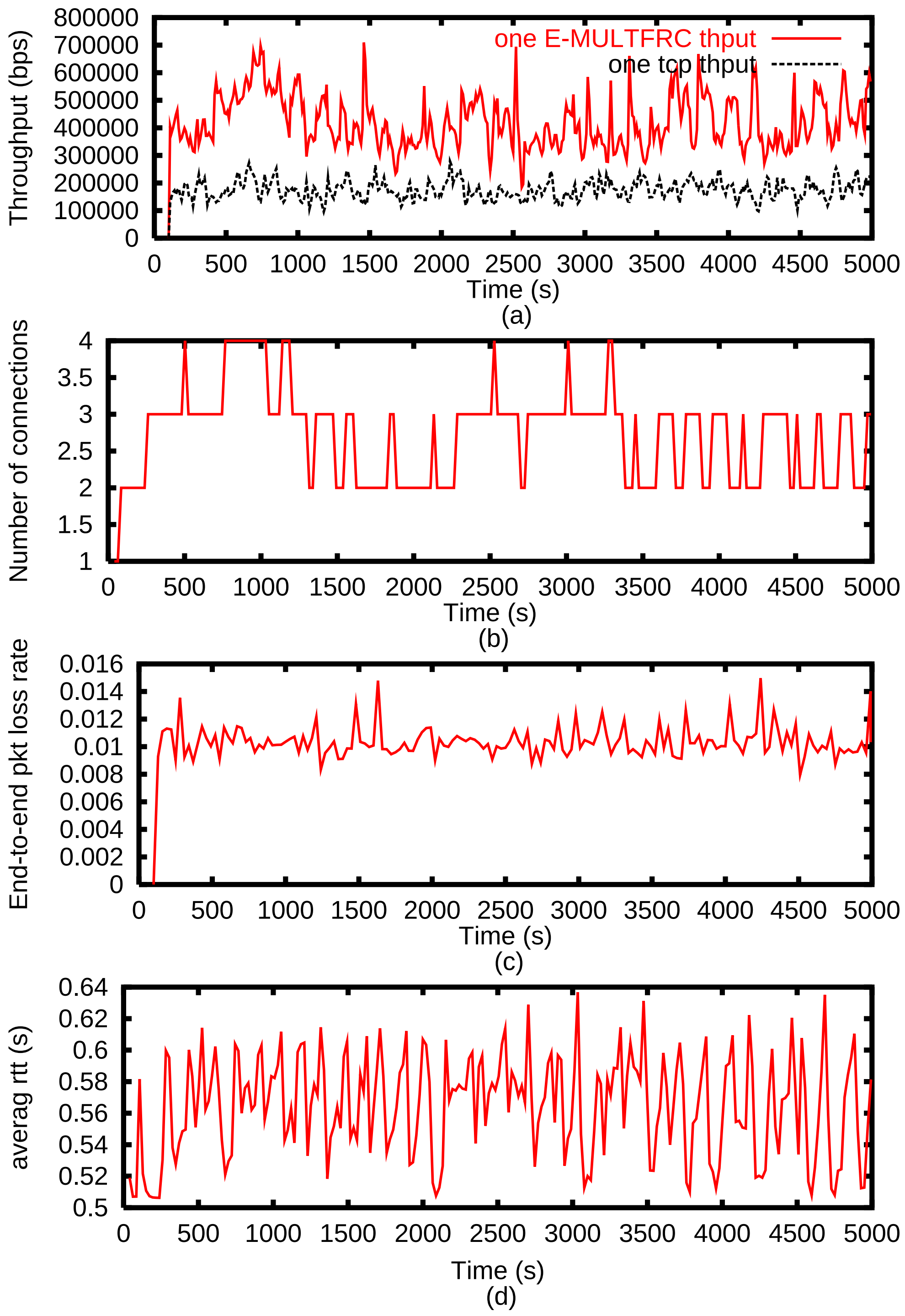


Figure 5.17: NS-2 simulation results of E-MULTFRC for the case $p_w = 0.01, \gamma = 0.2$ for E-MULTFRC+TCP scenario: the dynamics of (a)throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

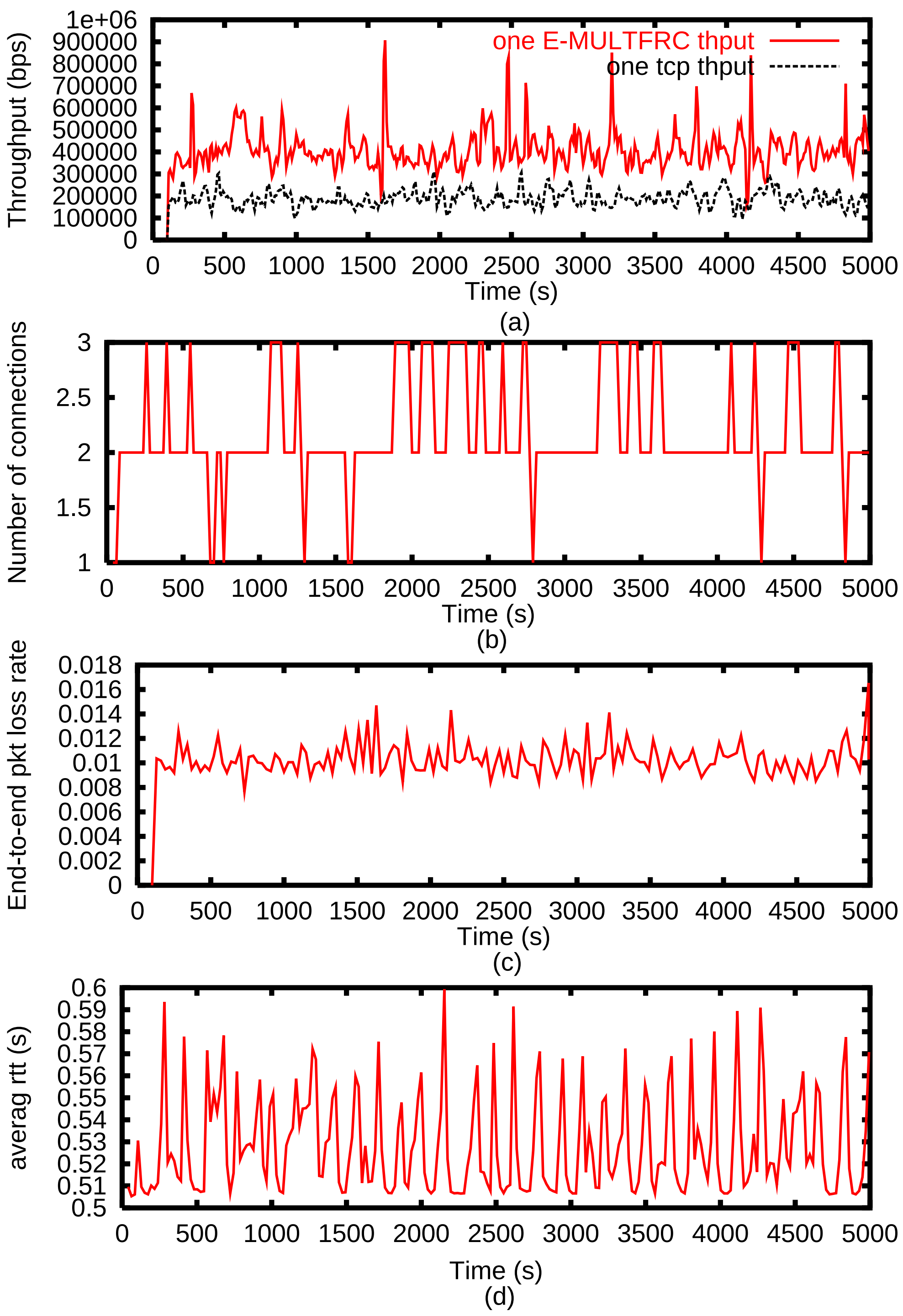


Figure 5.18: NS-2 simulation results of E-MULTFRC for the case $p_w = 0.01, \gamma = 0.1$ for E-MULTFRC+TCP scenario: (a)throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

5.2.6 Comparison Between E-MULTFRC and Video Transport Protocol (VTP)

VTP [26] is an experimental protocol with a new end-to-end rate control mechanism designed specifically for real-time streaming in wireless networks. It relies on two techniques, namely the Achieved Rate (AR) estimation and Loss Discrimination Algorithm (LDA). AR is the receiving rate measured by receiver, plus the fraction corresponding to packet loss caused by wireless error. VTP uses an end-to-end statistics based LDA called Spike that infers congestion based on the measured RTT, and switches between a congestion and an error state [26].

Similar to TCP, VTP linearly probes the available bandwidth until congestion is detected. However, VTP does not perform multiplicative decrease upon congestion, instead it reduces the sending rate to AR and avoids the drastic rate reductions for improved smoothness, at the same time maintaining the same average rate as TCP. In contrast to the highly fluctuating TCP rate, VTP reduces its rate by a lower amount but keeps this reduced rate for longer. It is easy to see that in the steady state in VTP, the average sending rate is equal to AR.

VTP belongs to the end-to-end statistics based solution for TFRC over wireless, as mentioned in Section 1.2. VTP is an entirely new rate control protocol and hence requires modifications to transport layer protocol stack. Its effectiveness has not been evaluated either in theory or over the Internet or in practical wireless networks.

In this section, we briefly compare performance of E-MULTFRC and VTP using NS-2 simulations over the topology shown in 5.1. The sender denoted by s, and the receiver denoted by r, both run E-MULTFRC with slow start option at the application layer in one simulation, and VTP at the transport layer in the other simulation. A detailed comparison is provided by Yang et. al. [58].

For all simulations, the wireless bandwidth B_w is set be 4 Mbps and is assumed to be the bottleneck. Round trip propagation delay RTT_{min} is 80 ms. We send dummy data from sender to receiver for 300 seconds, using both E-MULTFRC and VTP schemes. In one set of comparison, the wireless link is modeled by an exponential error model, and p_w is set to be 0.02. In the other set of comparison, the wireless link is modeled by a two states Markov model with following settings: duration of good state is 1 second; duration of bad state is 0.5 second; $p_w = 0$ in good state; $p_w = 0.06$ in bad state; hence the overall packet loss rate caused by wireless error is 2%; state transition matrix is

$$\begin{pmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{pmatrix}.$$

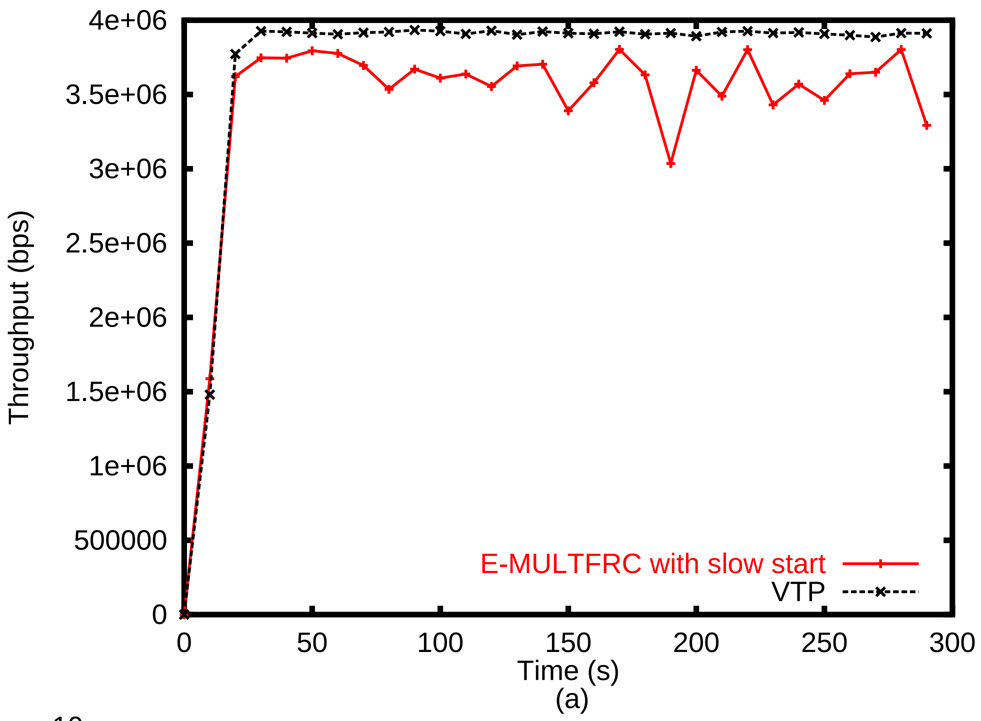
DropTail type queue is used for each node. In order to compare the performance of E-MULTFRC and VTP in the presence of wireless channel errors, we examine their throughput in the presence of simulated wireless random and burst loss. In all the simulations, throughput is measured every 10 seconds, and the number of connections is sampled whenever there is a change.

The throughput comparison in the presence of random loss and burst loss are shown in Figures 5.19 and 5.20, respectively. As seen, in the presence of random wireless error, VTP has a slightly higher throughput than E-MULTFRC and its utilization ratio is 97%, compared to that of E-MULTFRC at 91 %. VTP's result is consistent with those in literature: VTP's LDA works in a scenario with one flow, one wireless link, and random loss[8]. E-MULTFRC's performance is consistent with our analysis and simulation in previous sections.

On the other hand, in the presence of burst error, E-MULTFRC outperforms VTP all the time, and its overall utilization is 85%, as compared to VTP's 70%. This is consistent with the results in [58] in that VTP's LDA works sub-optimally in burst loss scenarios. E-MULTFRC, on the other hand, is not sensitive to burst errors happening in a timescale much faster than that of changing connections. Intuitively, this is because E-MULTFRC cares about long term average utilization over 20 seconds, and as such effects of the short term packet loss fluctuation on utilization tend to average out, and will not affect E-MULTFRC's performance in general.

In summary, as compared to VTP, E-MULTFRC performs slightly worse in presence of random loss, and much better in the presence of burst loss. Moreover, E-MULTFRC's performance has been evaluated in this thesis both theoretically and experimentally over commercial wireless networks. Meanwhile VTP's theoretical and experimental performance are subject to further research. Last but not the least, VTP is a completely new rate control/congestion control protocol, and hence requires modification to transport layer protocol stack, i.e. at the operating system level, in order to be deployed; E-MULTFRC requires no

[8]VTP's performance has not been extensively evaluated in a simulation or real scenarios with multiple users and various loss conditions. An extensive study of LDA algorithms, including VTP's, is included in [21].



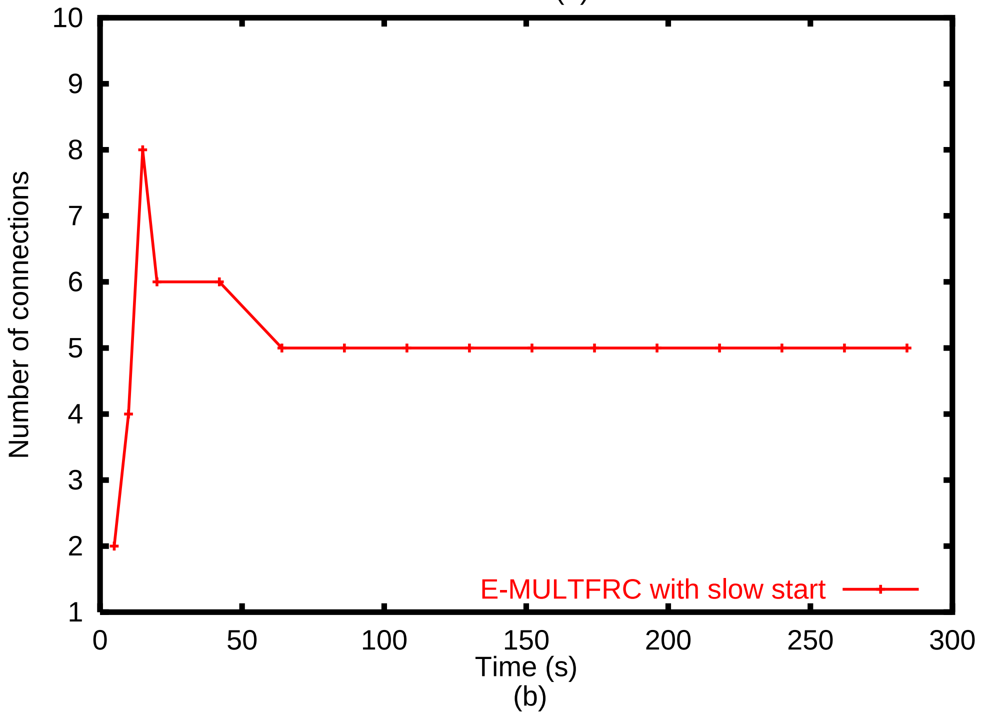
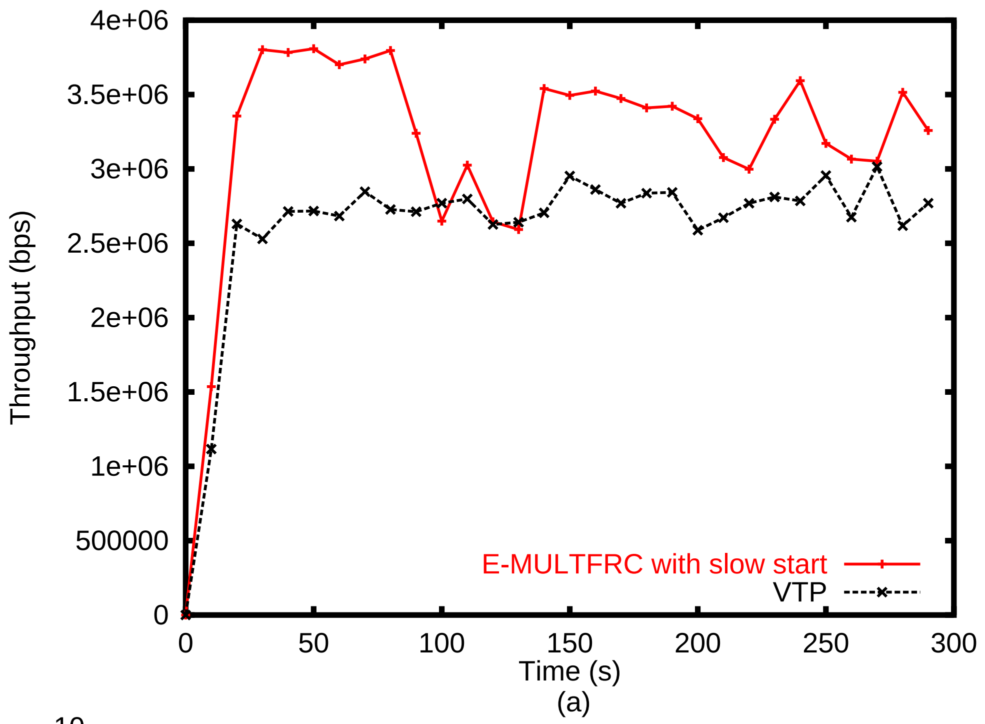


Figure 5.19: Performance comparison between E-MULTFRC and VTP, in the presence of random loss: (a) throughput; (b) the number of connections opened by E-MULTFRC.



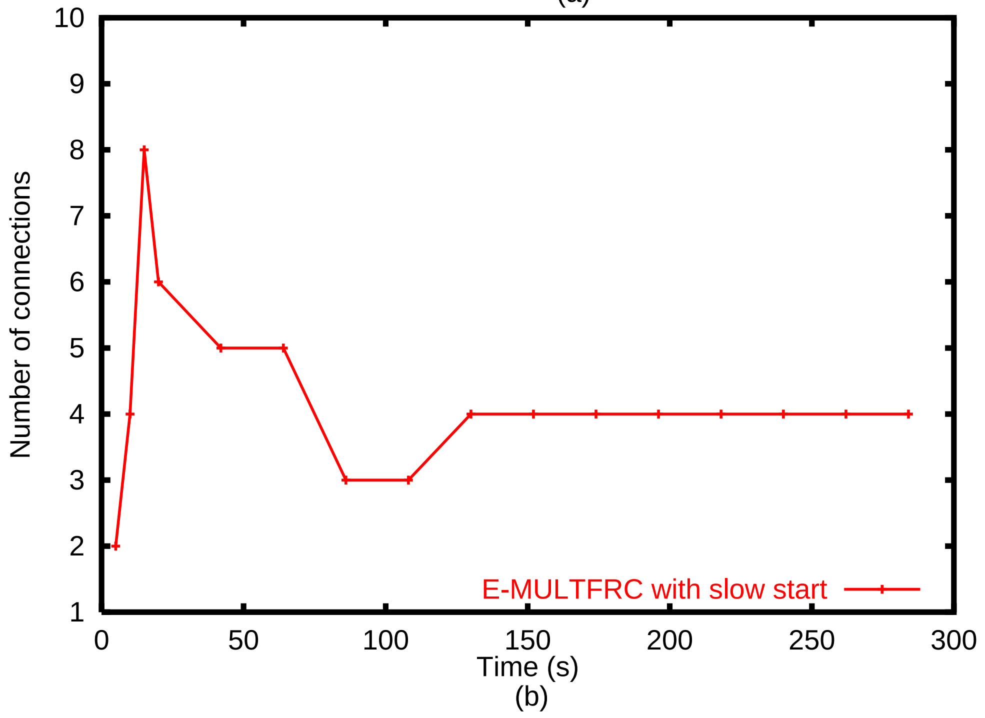


Figure 5.20: Performance comparison between E-MULTFRC and VTP, in the presence of burst loss: (a) throughput; (b) the number of connections opened by E-MULTFRC.

modification to existing transport layer protocol stack and underlying infrastructure, and is potentially easier to deploy.

5.3 MULTFRC: NS-2 Simulations and 1xRTT Wireless Experiments

In this section, we carry out NS-2 simulations and actual experiments over Verizon Wireless 1xRTT CDMA data network to evaluate the performance of MULTFRC system.

5.3.1 Setup

The topology used in simulations is the same as E-MULTFRC's simulations, as shown in Figure 5.1. The sender denoted by s, and the receiver denoted by r, both run MULTFRC at the application layer.

We examine two issues; first, how MULTFRC performs in terms of average throughput, average round trip time, and packet loss rate, as a function of p_w. Second, whether the number of connections is stable.

For the actual experiments over 1xRTT, we stream from a desktop connected to Internet via 100 Mbps Ethernet in EECS, Berkeley, to a notebook connected to Internet via Verizon Wireless 1xRTT CDMA data network. The packet size S is 1460 bytes, and the streaming takes 30 minutes.

5.3.2 Performance Characterization of MULTFRC

We have empirically found the following parameters to result in reasonable performance: $\alpha = \beta = 1$, $\gamma = 0.2$ and $m = 50$.

We simulate the MULTFRC system to stream for 9000 seconds, and compute the average throughput and packet loss rate for p_w =0.0, 0.02, 0.04, 0.06 and 0.08, and compare them to the optimal, i.e. $B_w(1 - p_w)$ for each p_w. The results for $B_w = 1\ Mbps$ and $RTT_{min} = 168\ ms$ are shown in Figure 5.21. As seen, the throughput is within 25% of the optimal. The round trip time is within 20% of RTT_{min}, the same as the expected range, i.e. γRTT_{min}. The packet loss rate is almost identical to the optimal, i.e. a line of slope one as a function of wireless channel error rate. As expected, the average number of connections increases with wireless channel error rate, p_w. To confirm MULTFRC's performance over a wider range of parameters, we carry out additional simulations using the same topology as in Figure 5.1, with $B_w = 100\ kbps$ and $RTT_{min} = 757\ ms$. The results, shown in Figure 5.22 are as expected, and validate our earlier observations. [9]

Considering the throughput plots in Figures 5.21 and 5.22, we notice that for some values of p_w, there is a significant difference between the actual and optimal throughput. This is due to the quantization effect in situations where the number of connections is small, i.e. 2 to 4. In these situations, a small oscillation around the optimal number of connections results in large variation in observed throughput. One way to alleviate this problem is to increase γ in order to tolerate larger queuing delay and hence absorb throughput fluctuations, at the expense of being less responsive. Another alternative is to use smaller packet size in order to reduce the "quantization effect" at the expense of (a) lower transmission efficiency and (b) the slower rate of convergence to the optimal number of connections. As discussed in Section 4.5, it is also possible to combine the multiple TFRC connection into one, in order to mitigate quantization effect.

The comparison between the performance of E-MULTFRC and MULTFRC in Figure 5.10 also shows the difference caused by applying different control laws in these two schemes. When p_w is small, e.g. 0.02, the optimal number of connections is small. In this case, E-MULTFRC outperforms MULTFRC since when decreasing the number of connections n, E-MULTFRC only decreases it proportionally whereas MULFRC decrements it. On the other hand, when p_w is large, e.g. 0.08, the optimal number of connections is large. In this case, the proportional decrease in n is more significant than decrementing it. Therefore, as the behavior of E-MULTFRC and MULTFRC are similar when increasing the number of connections, the one with more significant decrease in n will have lower average throughput. Hence, when channel condition is worse and the packet loss rate caused by channel error is high, MULTFRC will have a better utilization than E-MULTFRC, due to the advantage of IIAD control law on the number of connections. This confirms the utilization ratio comparisons in Sections 4.2.3 and 4.4 for E-MULTFRC and MULTFRC, respectively.

To examine the dynamics of MULTFRC system, we show throughput, packet loss

[9]Note the round trip times for $p_w = 0$ are shown neither in Figure 5.21 or 5.22 because they represent the channel error free case in which MULTFRC reduces to one TFRC connection.

rate, and the number of connections as a function of time for $p_w = 0.04$ in Figure 5.23. As seen, the throughput and the number of connections are quite stable; as expected, packet loss rate is around 0.04 and round trip time is low, and is in agreement with the results corresponding to $p_w = 0.04$ in Figure 5.21. Similar results are obtained for other values of p_w.

In order to examine MULTFRC's performance as a function of p_w, we use MULTFRC with p_w initially set at 0.02. Then at 3000^{th} second, p_w is switched to 0.08, and at 6000^{th} second switched back to 0.02. Here, we artificially change p_w to see how MULTFRC adapts to the change in p_w. The throughput, packet loss rate, round trip time and the number of connections opened are shown in Figure 5.24. As seen, the number of connections varies from around 3 to around 7 as p_w switches from 0.02 to 0.08.

Using NS-2 simulations, we have empirically found the fairness results between MULTFRC and TCP/TFRC to be similar to that of E-MULTFRC and TCP/TFRC. This is not surprising because from Corollaries 3.4.1 and 3.7.1, it is clear that both E-MULTFRC and MULTFRC use only the leftover bandwidth that TCP/TFRC can not utilize in the channel-error-limited region. Furthermore, in the capacity-limited region, we have shown that E-MULTFRC and MULTFRC open only one connection, and as such share bandwidth fairly with TCP/TFRC.

5.3.3 Experimental Results for MULTFRC

As for actual experiments, we compare the performance of MULTFRC, one TFRC, and E-MULTFRC in Table 5.11. As seen, MULTFRC on average opens up 1.8 connections, and results in 60% higher throughput than one TFRC connection system, at the expense of a larger round trip time, and higher packet loss rate. As seen, the performance of MULTFRC and E-MULTFRC are pretty close.

Table 5.11: Actual experimental results for a MULTFRC system over 1xRTT CDMA.

scheme	**throughput (kbps)**	**rtt (ms)**	**packet loss rate**	**ave. # of conn.**	**throughput improvement over one TFRC (%)**
one TFRC	54	1624	0.031	N/A	N/A
MULTFRC	86	2512	0.045	1.8	60
E-MULTFRC	89	2762	0.41	1.9	65

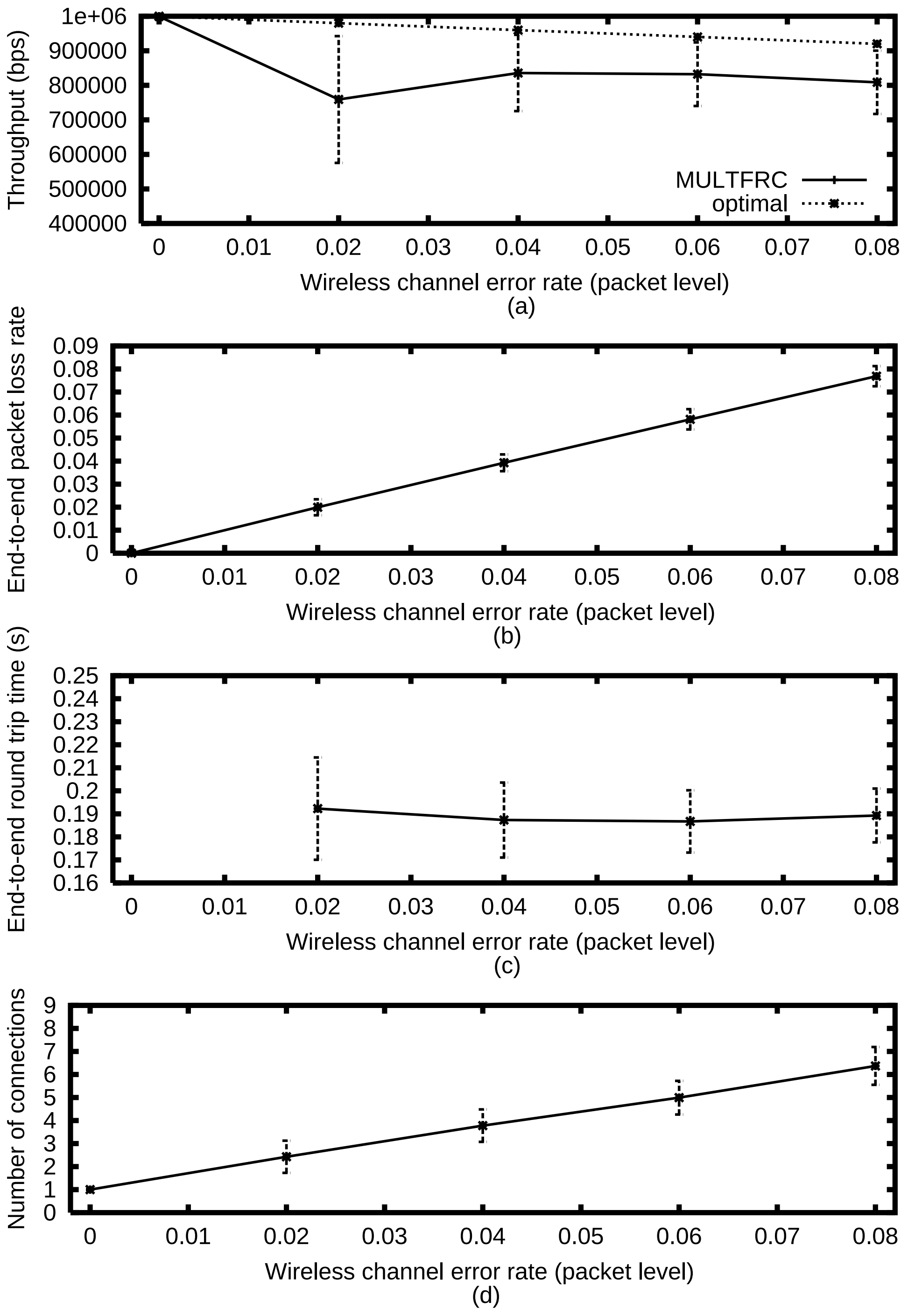


Figure 5.21: NS-2 simulations of MULTFRC for $B_w = 1\ Mbps$ and $RTT_{min} = 168\ ms$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

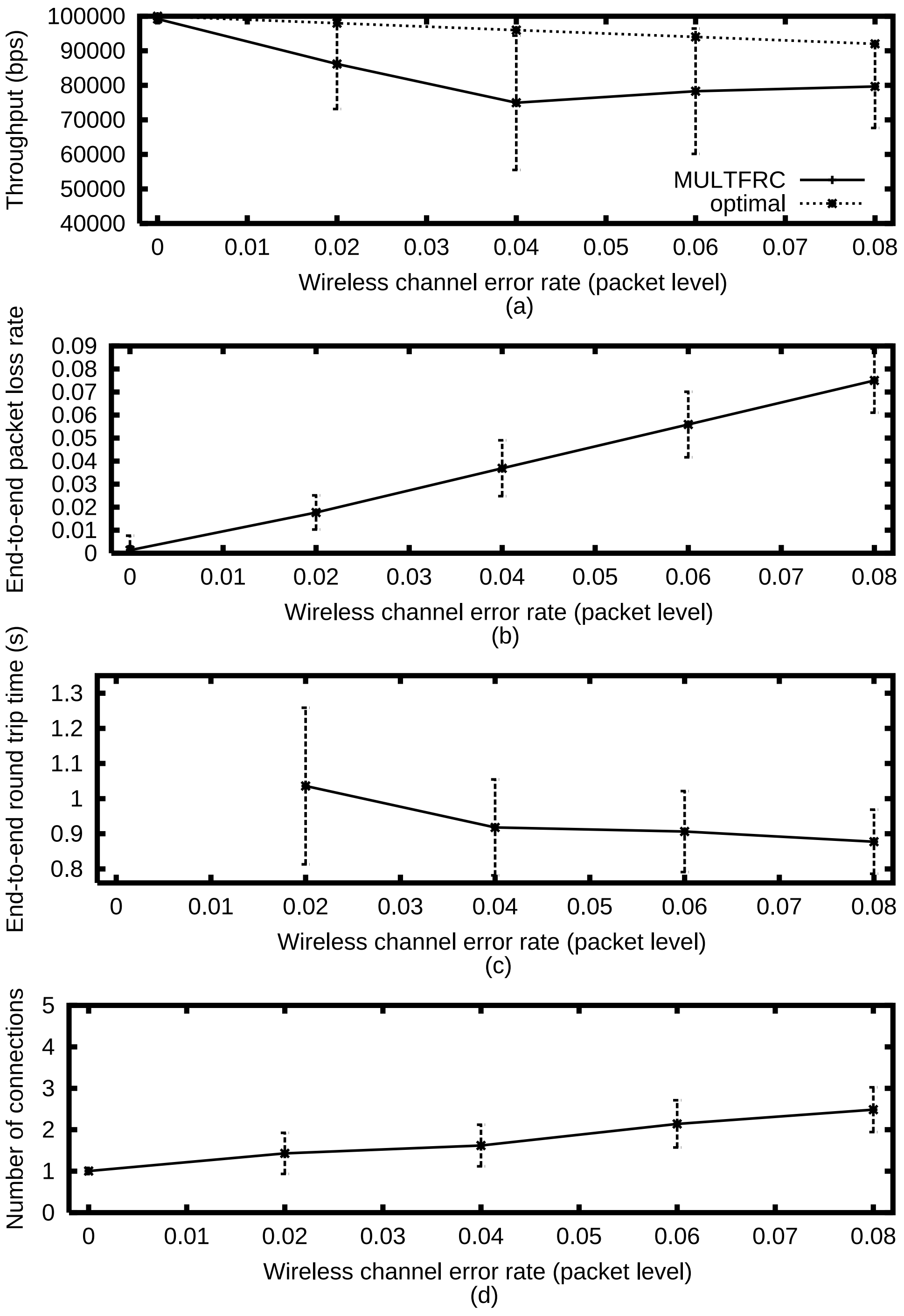


Figure 5.22: NS-2 simulations of MULTFRC for $B_w = 100\ kbps$ and $RTT_{min} = 757\ ms$; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end round trip time, (d) number of connections, all as a function of packet level wireless channel error rate.

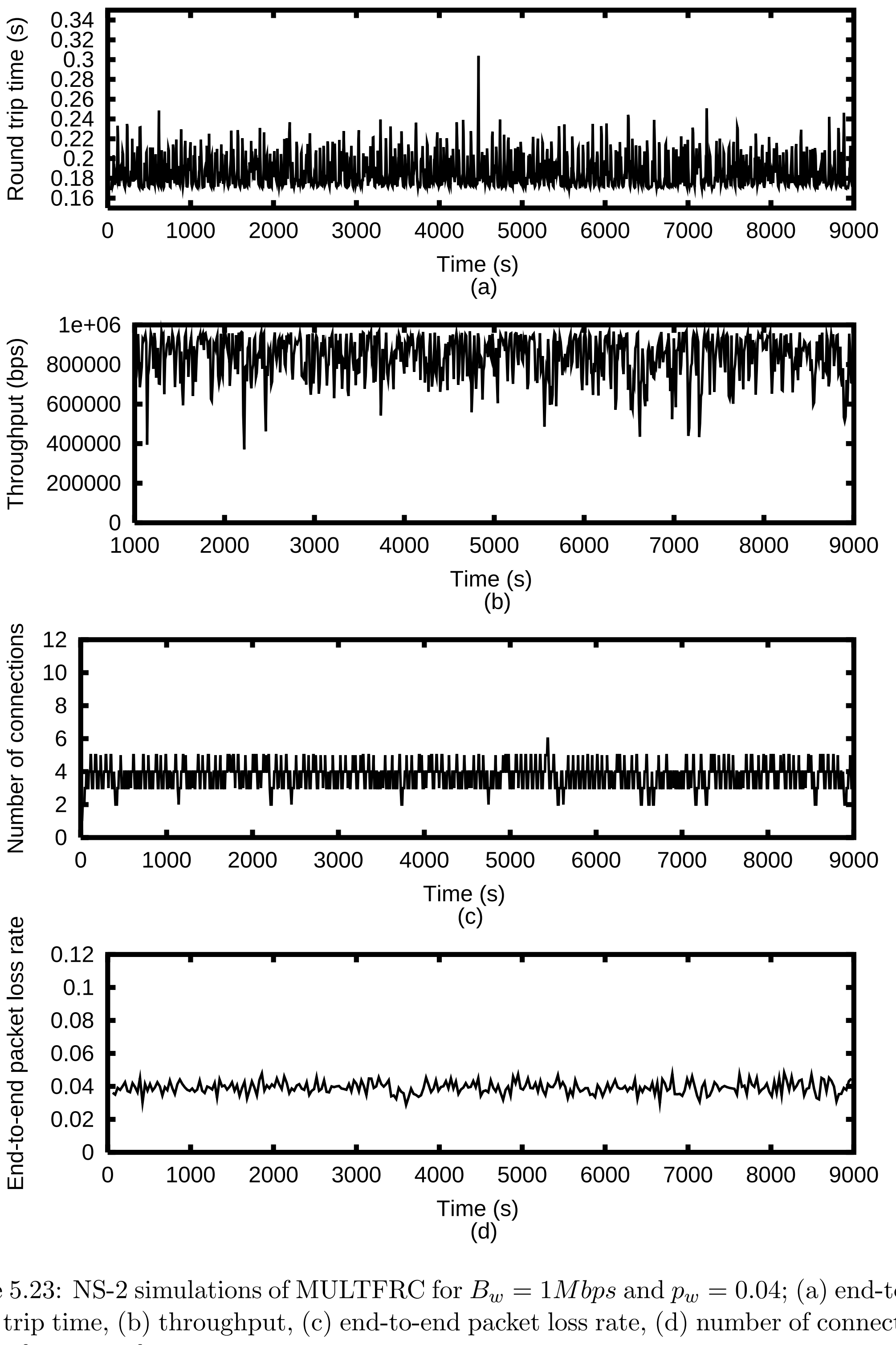


Figure 5.23: NS-2 simulations of MULTFRC for $B_w = 1Mbps$ and $p_w = 0.04$; (a) end-to-end round trip time, (b) throughput, (c) end-to-end packet loss rate, (d) number of connections, all as a function of time.

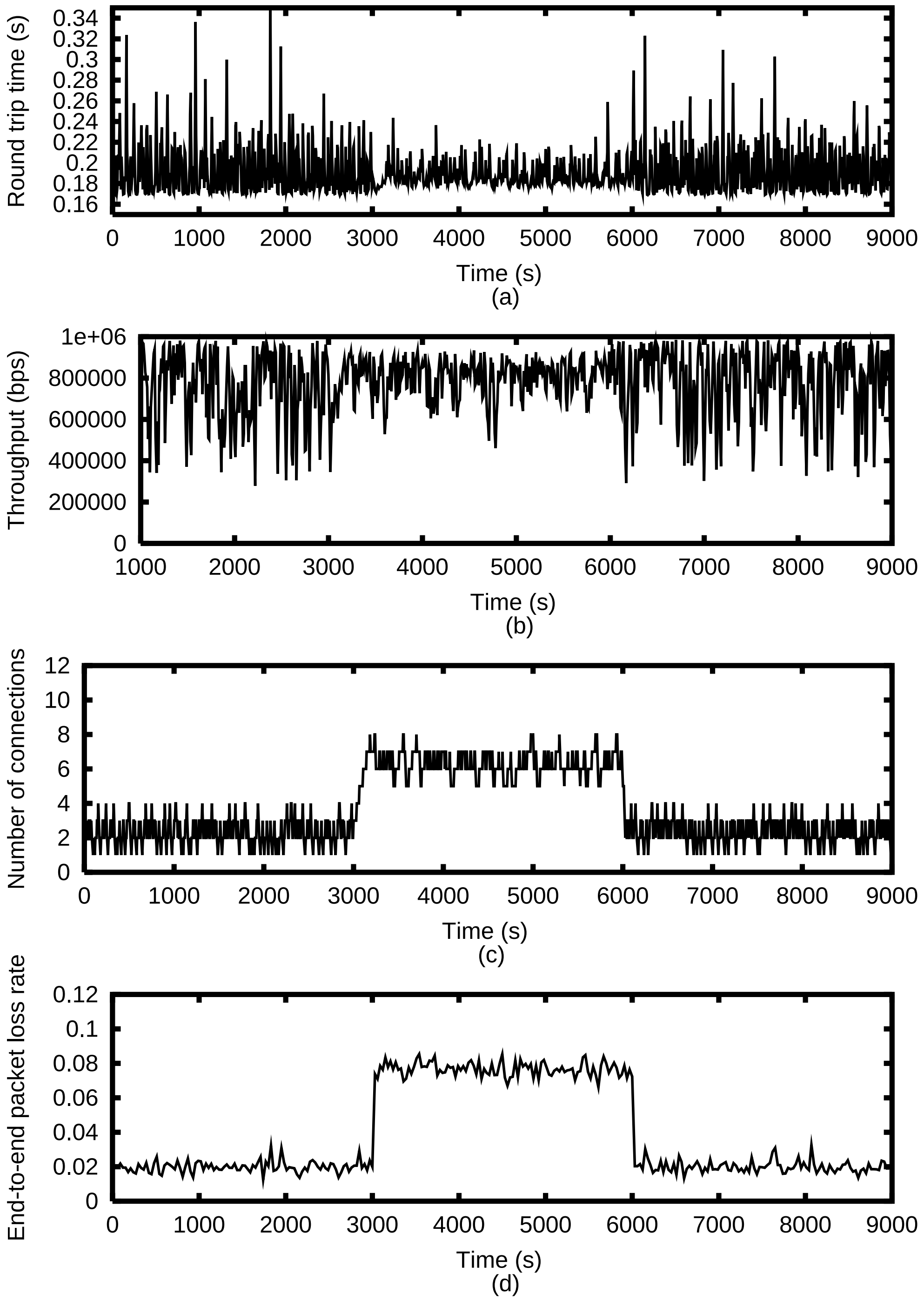


Figure 5.24: NS-2 simulation results of MULTFRC as p_w changes from 0.02 to 0.08 and back again; (a) end-to-end round trip time, (b) throughput, (c) numbers of connections, (d) end-to-end packet loss rate, all as a function of time.

5.4 E-AIOTFRC: Simulation Results

In this section, we carry out NS-2 simulations to evaluate the performance of E-AIOTFRC. Specifically, we examine three issues in these simulations: (a) how E-AIOTFRC performs in terms of average throughput, average RTT, and packet loss rate, as a function of p_w, and how it compares with E-MULTFRC; (b) whether n is stable; (c) whether or not E-AIOTFRC is fair to TCP. In all the simulations, throughput is measured every 10 seconds, packet loss rate is measured every 30 seconds, the average RTT is measured every 100 packets, and the number of connections is sampled whenever there is a change.

The topology used in simulations for utilization evaluation is again the same as EMULTCP's, shown in Figure 5.1. The sender is denoted by s, and the receiver is denoted by r. They both run E-AIOTFRC at the application layer.

We simulate the E-AIOTFRC system to stream for 9000 seconds. The average throughput, end-to-end packet loss rate, average RTT, and average n for p_w =0.0, 0.02, 0.04, 0.06 and 0.08 are shown in Figure 5.25, where $RTT_{min} = 168\ ms$. As seen, the throughput is within 15% of the optimal, and the packet loss rate is almost identical to the optimal, i.e. a line of slope one as a function of wireless channel error rate. As expected, the average n increases with wireless channel error rate, p_w. [10]

The throughput of E-MULTFRC is also shown in Figure 5.25(a) for comparison. As seen, E-AIOTFRC has almost the same throughput as E-MULTFRC when p_w is high, while it significantly outperforms E-MULTFRC when p_w is low. For example, when $p_w =$ 0.02, E-AIOTFRC achieves 95% utilization of the wireless bandwidth, while E-MULTFRC's utilization is only 77%. Therefore, by avoiding the "quantization effect", E-AIOTFRC achieves better throughput performance than E-MULTFRC.

To examine the dynamics of E-AIOTFRC systems, we show throughput, the number of virtual connections n, packet loss rate, and average RTT as a function of time for $p_w = 0.04$ in Figure 5.26. As seen, the throughput and the number of connections are quite stable; as expected, packet loss rate is around 0.04 and RTT is properly controlled to be around γRTT_min. Similar results are obtained for other values of p_w.

To investigate the fairness of E-AIOTFRC, we carry out NS-2 simulations based on the "dumbbell" topology shown in Figure 5.6. Senders are denoted by $si, i = 1, \ldots, 16$, and

[10]Note the RTT for $p_w = 0$ is not shown in Figure 5.25 because it represents the channel error free case in which E-MULTFRC reduces to one TFRC connection.

receivers are denoted by $di, i = 1 \ldots, 16$. We investigate two types of fairness: the inter-protocol fairness between E-AIOTFRC and TCP, and the intra-protocol fairness within E-AIOTFRC.

The intra-protocol fairness is defined as the fairness between E-AIOTFRC flows. In our simulations, we run E-AIOTFRC on all 16 sender-receiver pairs shown in Figure 5.6 for 5000 seconds, and compare their throughput. E-AIOTFRC is said to be intra-protocol fair if all receivers get the same throughput. The fairness ratios for $p_w = 0.01$ and $p_w = 0.04$ are shown in Table 5.12. The fairness ratio is defined as receivers' throughput divided by the average throughput; the closer to one, the more fair the E-AIOTFRC system is. As seen, the fairness ratio is fairly close to one, indicating E-AIOTFRC flows are fair to each other, at least in this simulation setting. The bandwidth utilization ratios are 95% for $p_w = 0.01$ and 98% for $p_w = 0.04$.

Table 5.12: Simulation results for intra-protocol fairness of E-AIOTFRC.

receiver	fairness ratio p_w=0.01	fairness ratio p_w=0.04	receiver	fairness ratio p_w=0.01	fairness ratio p_w=0.04
d1	1.00	0.99	d9	1.02	1.03
d2	0.99	0.99	d10	0.98	0.98
d3	0.96	1.00	d11	0.97	1.01
d4	0.99	0.97	d12	0.99	0.99
d5	1.05	0.95	d13	0.99	1.01
d6	1.04	1.01	d14	1.01	0.97
d7	1.03	1.02	d15	1.01	0.98
d8	1.02	1.03	d16	0.96	1.02

The inter-protocol fairness is defined as the fairness between E-AIOTFRC and TCP[11]. In our simulations, we run E-AIOTFRC on the first 8 sender-receiver pairs, i.e. $(si, di), i = 1, \ldots, 8$, and TCP on the remaining 8 sender-receiver pairs shown in Figure 5.6; each session lasts 5000 seconds, and we compare their throughput for $p_w = 0.01$ and $p_w = 0.02$. Under the simulation settings, each E-AIOTFRC consumes more bandwidth than one TCP under full utilization. This is because in this case, the wireless channel error rate is large enough to make the number of virtual connections of each E-AIOTFRC to be larger than one. Hence, it is meaningless to define the fairness between E-AIOTFRC and

[11]We use TCP SACK implementation in simulations.

TCP as having the same throughput.[12] As such, in our simulations, we define E-AIOTFRC to be fair to TCP if it does not result in a decrease in TCP's throughput. Specifically for our simulations, it implies TCP retains the same throughput whether or not it coexists with E-AIOTFRC under the same network setting. The throughput of E-AIOTFRC and TCP, as well as the total bandwidth utilization ratios for the setup shown in Figure 5.6, are shown in Table 5.13 for two scenarios: (a) 8 E-AIOTFRC coexisting with 8 TCP connections, (b) 16 TCP connections.

Figures 5.27 and 5.28 also show the dynamics of throughput, packet loss rate, RTT, and the number of virtual connections n. Comparing E-AIOTFRC+TCP with TCP-alone, we see the former has a much higher utilization of the wireless bandwidth at the expense of lower TCP throughput. A careful examination of Figure 5.27 reveals that this throughput drop is caused by the higher RTT for E-AIOTFRC+TCP as compared with TCP-alone. For example, for $p_w = 0.01$ and $\gamma = 0.5$, E-AIOTFRC+TCP experiences around 0.6 seconds RTT, while TCP-alone only experiences 0.5 seconds RTT, i.e. the propagation delay. As TCP's throughput is known to be inversely proportional to RTT, the 20% increase in the RTT explains the 16% decrease in the TCP's throughput shown in first row in Table 5.13.

Table 5.13: Simulation results for fairness between E-AIOTFRC and TCP.

settings	8 E-AIOTFRC + 8 TCP			16 TCP	
	ave. thput. (E-AIOTFRC) (kbps)	ave. thput. (TCP) (kbps)	utili-zation (%)	ave. thput. (TCP) (kbps)	utili-zation (%)
p_w=0.01 γ=0.5	436.501	168.048	98	200.168	65
p_w=0.01 γ=0.1	379.656	185.286	91	200.168	65
p_w=0.02 γ=0.5	486.821	120.313	99	139.674	46
p_w=0.02 γ=0.1	449.226	130.953	95	139.674	46

This increase in the RTT is, by design, a consequence of E-AIOTFRC controlling the number of virtual connections n according to Equation (4.2). As n is only decreased

[12]Obviously, there are situations in which E-AIOTFRC ends up with performing similar to one TFRC. An example would be E-AIOTFRC competing for bandwidth with TCP on wired networks. In that case, however, the fairness between E-AIOTFRC and TCP is reduced to the fairness between TFRC and TCP, and has been well explored in [2].

after the queuing delay exceeds the threshold γrtt_min, round trip time is increased when E-AIOTFRC increases n to achieve full utilization. One way to address this problem is to use a smaller value for γ, in order to reduce the increase in the RTT, and hence minimize the TCP's throughput drop. However, smaller values of γ also results in lower bandwidth utilization due to increased sensitivity of E-AIOTFRC to fluctuations in measured RTT. As shown in Table 5.13, $\gamma = 0.1$ results in a smaller drop in the TCP's throughput than $\gamma = 0.5$.

5.5 Video Related Simulations

To evaluate the performance of E-MULTFRC in video streaming applications, we simulate streaming of a 60 second long video clip through a channel, with throughput trace corresponding to one of the traces obtained from actual experiments over 1xRTT CDMA as described in Section 5.2.2. Our goal is to compare the quality of video streaming achievable using one TFRC connection with that of E-MULTFRC.

We encode 300 frames of *news.cif* sequence using MPEG-4 at bit rates varying from 50kps to 100 kbps [13] as controlled by TMN-5 [59]. The frame rate is 10 frame per second; the I-frame refresh rate is once every fifteen frames. The coded video bit stream is packetized with fixed packet size of 760 bytes. The packets are then protected using Reed-Solomon (RS) codes with different protection levels for one TFRC and E-MULTFRC. This is because packet loss statistics are different in the two cases. Specifically, the statistics of 30 minutes long trace indicates the longest burst loss to be 6 packets long for one TFRC and 11 packets long for E-MULTFRC. Thus, we apply RS(56,50) to one TFRC case, and RS(61,50) to E-MULTFRC case in order to sufficiently protect packets in both cases. Under ideal conditions where all packets in both schemes get through, the decoded video quality is identical between the two schemes. This is because before adding RS code, the source video bit rate is chosen to be the same for both schemes.

The RS-coded packets are then passed through channels simulated using one TFRC, and E-MULTFRC packet level traces each lasting 70 seconds, selected from the 30 minutes long actual experiments described in Section 5.2. The throughput and packet loss details for a 70 second long segment of one TFRC and E-MULTFRC connections are

[13] Our choices of video bit rates are related to the available bandwidth in today's commercial wireless networks.

shown in Figure 5.29. As Seen, both throughput and packet loss rate are higher for E-MULTFRC than for one TFRC case. The large throughput fluctuations in E-MULTFRC due to changing number of connections can potentially be argued not to be suitable for video applications in general; however, proper buffering can absorb these fluctuations in non-delay sensitive streaming applications.

The receiver decodes the received RS-coded packets and stores the MPEG-4 bit streams into a playback buffer. In this simulation, we fill the buffer with 10 seconds worth of data before starting the MPEG-4 decode and display process. The playback rate is fixed at 10 frames per second, and hence decoding process is stopped and the display is frozen whenever the playback buffer is empty.

To show the efficiency of E-MULTFRC, we compare the playback buffer occupancies of E-MULTFRC and one TFRC for several bit rates in Figure 5.30. At each bit rate the source rate is the same for both E-MULTFRC and TFRC, but the total streaming bit rate after FEC is higher for E-MULTFRC than TFRC. Thus, if both streams were received successfully at the receiver, we would expect the video quality to be identical. As seen in Figure 5.30, compared to one TFRC case, E-MULTFRC can sustain video streaming at higher bit rates and hence higher visual quality, despite the fact that it needs stronger FEC to combat the higher packet loss rate. Specifically, E-MULTFRC can sustain source bit rate of 90kbps and total streaming rate, including FEC, of 110kbps; while TFRC can only sustain some bit rate of 50kbps and total streaming rate, including FEC, of 56 kbps.

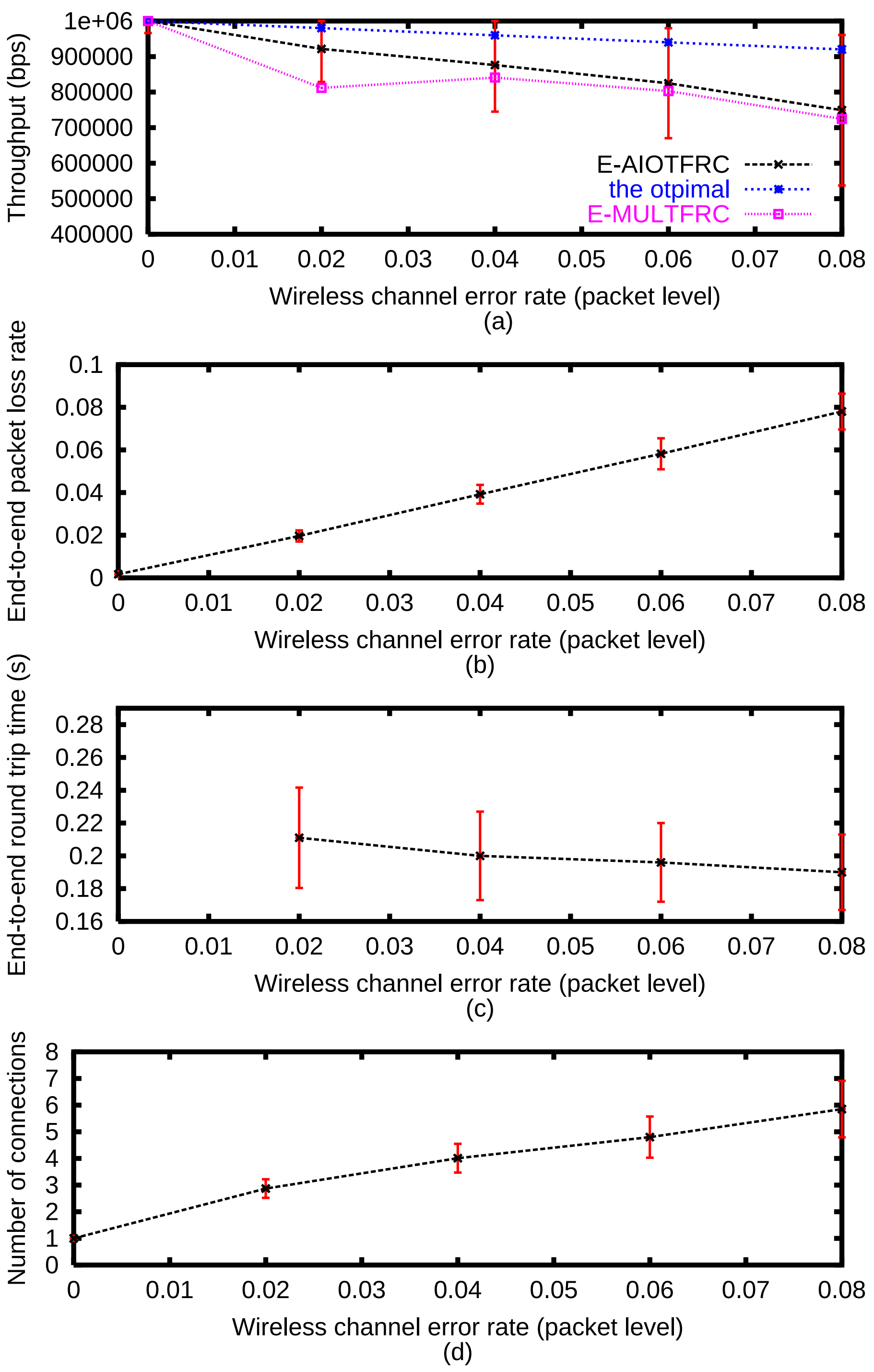


Figure 5.25: NS-2 simulations of E-AIOTFRC for Bw = 1 Mbps and RTTmin = 168 ms; (a) throughput, (b) end-to-end packet loss rate, (c) end-to-end RTT, (d) number of connections, all as a function of packet error rate on the wireless channel.

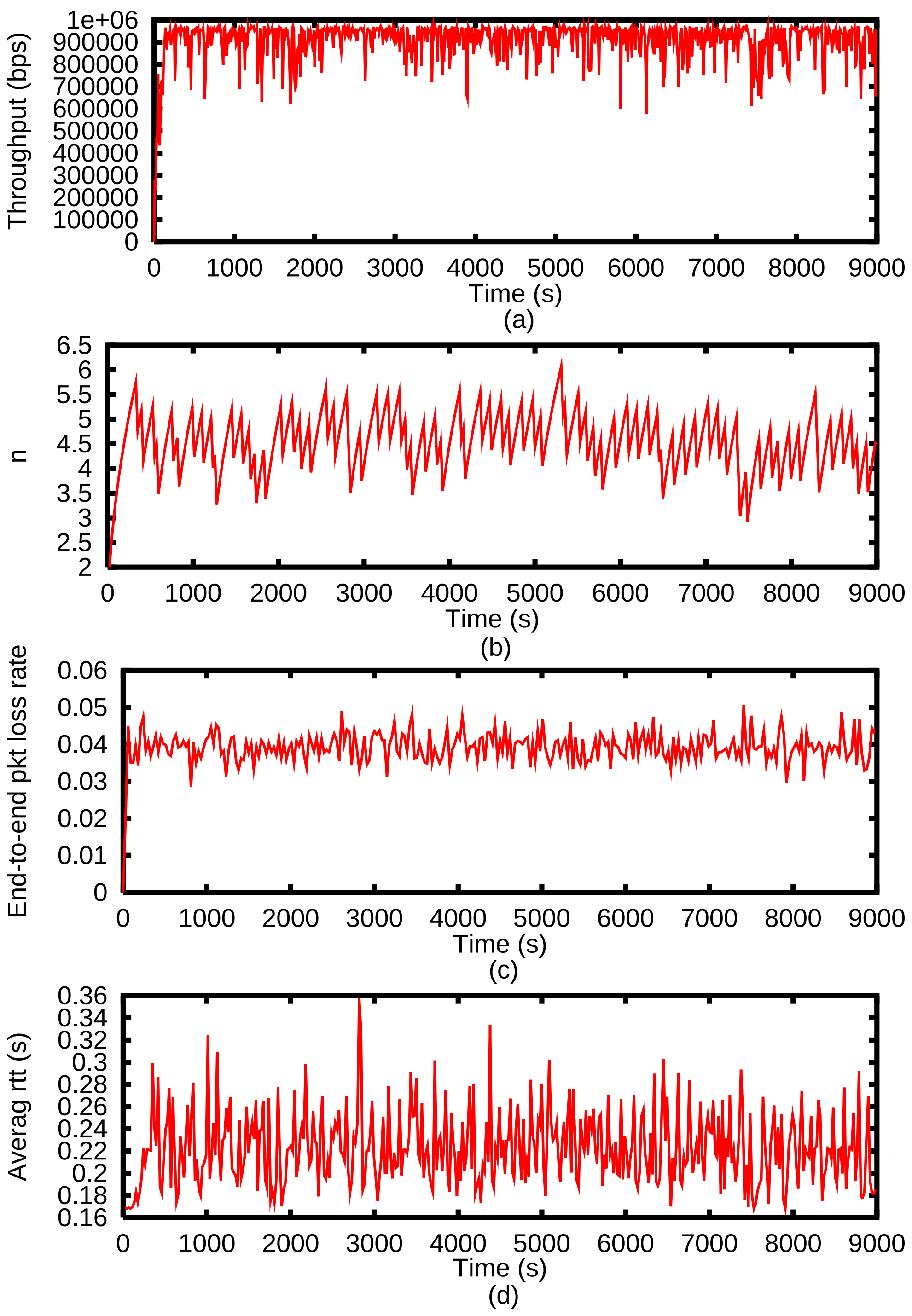


Figure 5.26: NS-2 simulations of E-AIOTFRC for $B_w = 1Mbps$ and $p_w = 0.04$; (a)throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

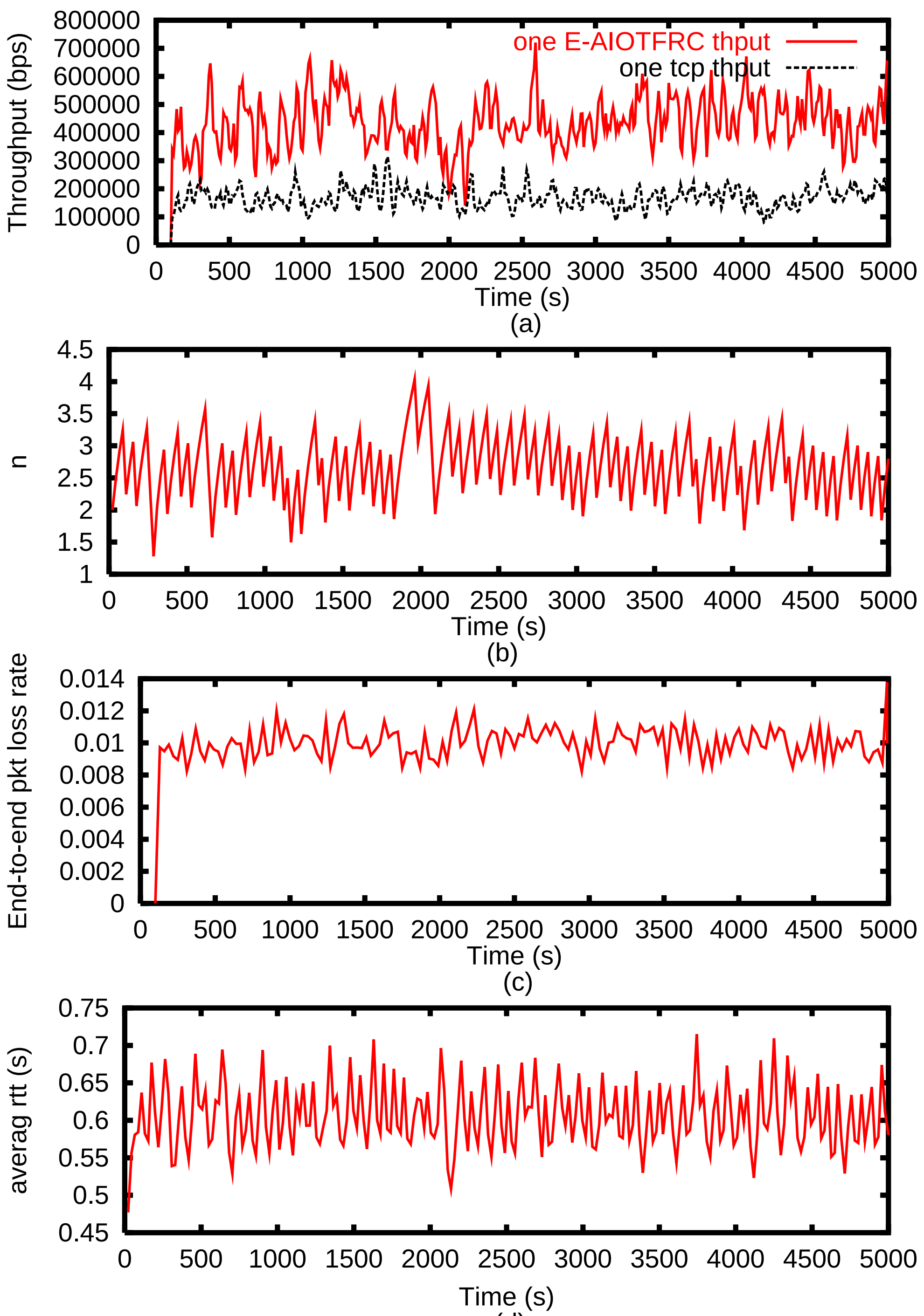


Figure 5.27: NS-2 simulation results of E-AIOTFRC coexisting with TCP, for the case $p_w = 0.01, \gamma = 0.5$: the dynamics of (a)throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

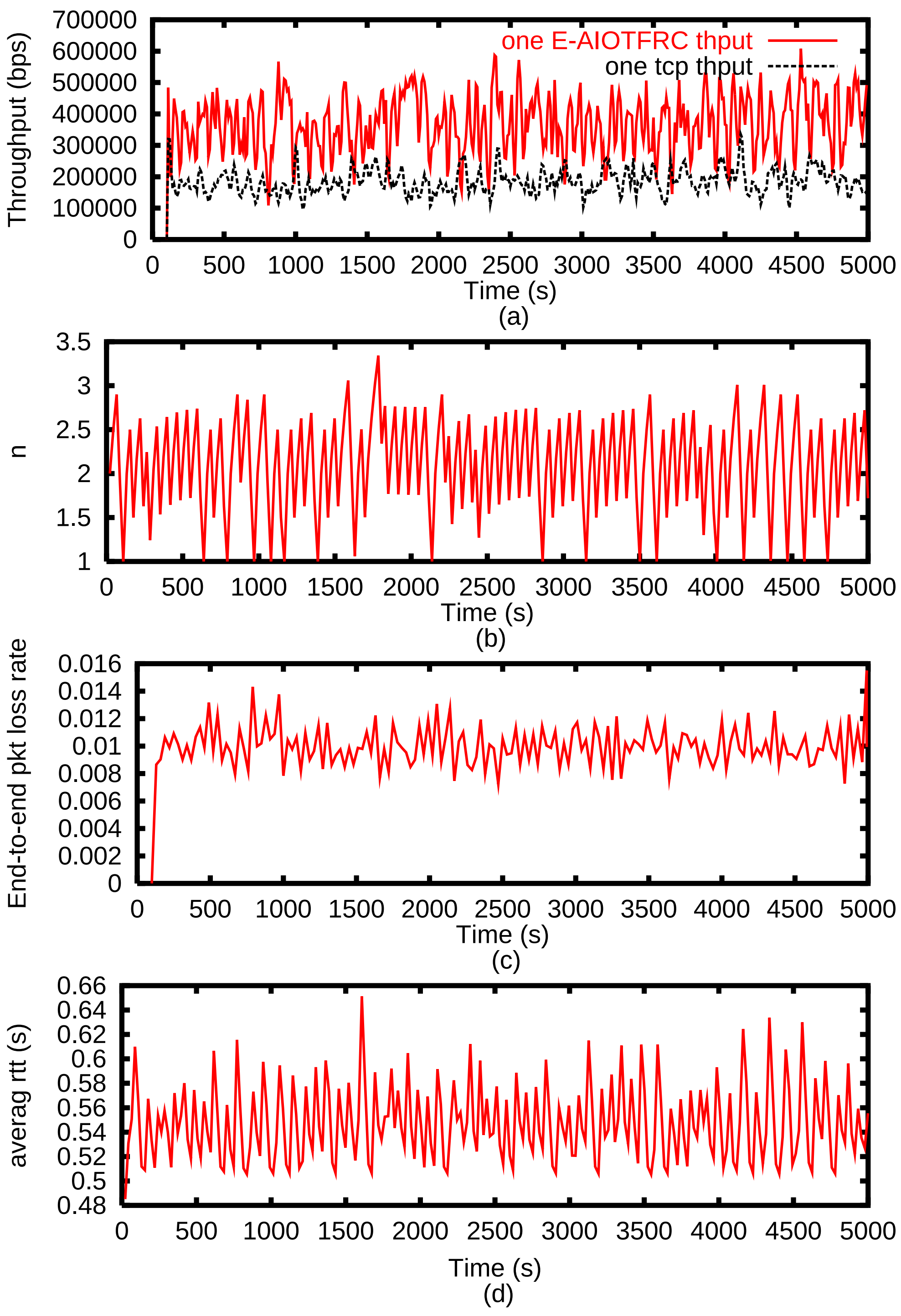
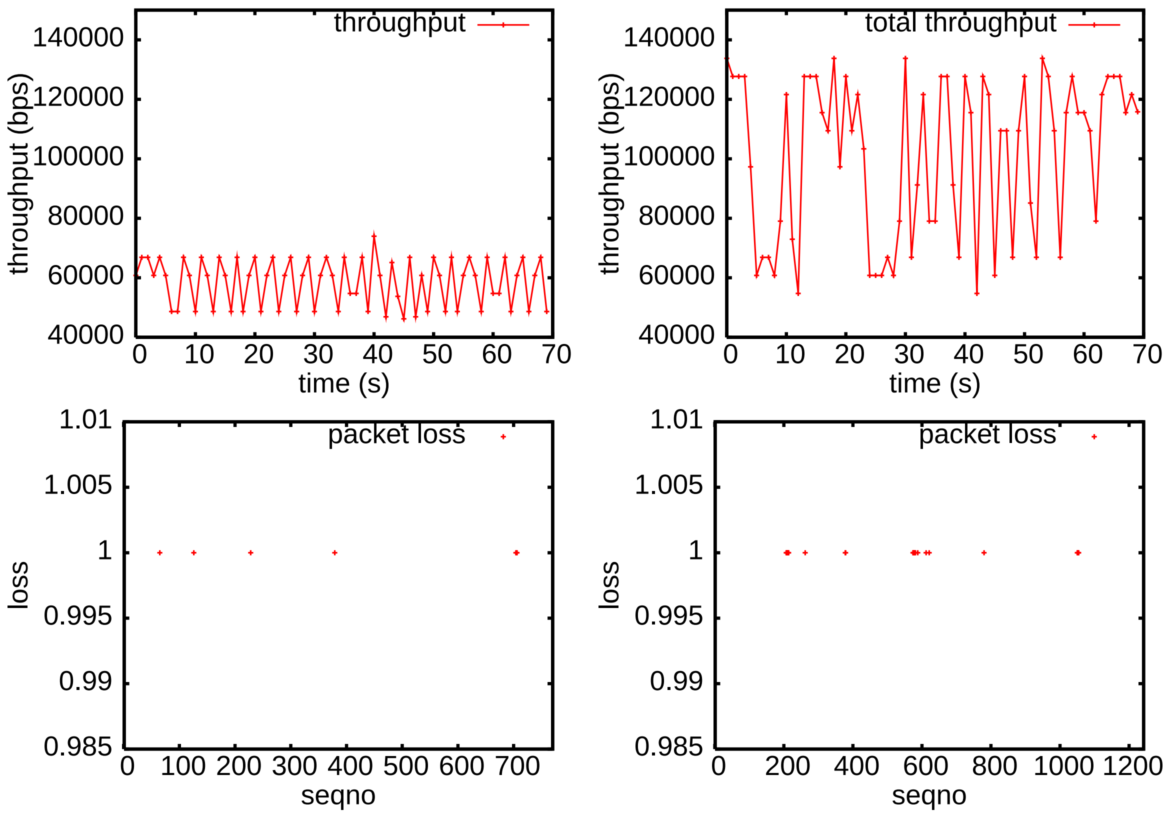
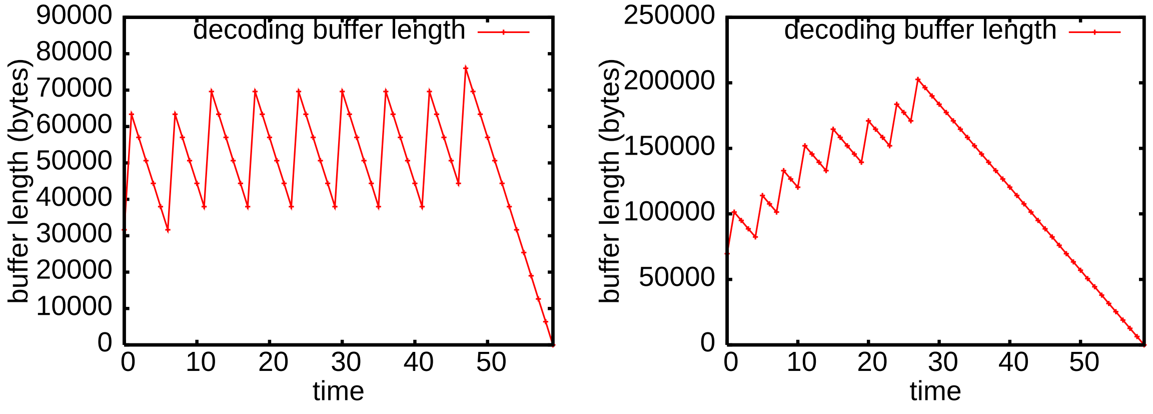


Figure 5.28: NS-2 simulation results of E-AIOTFRC coexisting with TCP, for the case $p_w = 0.01, \gamma = 0.1$: (a)throughput, (b) number of connections , (c) end-to-end packet loss rate, (d) end-to-end RTT, all as a function of time.

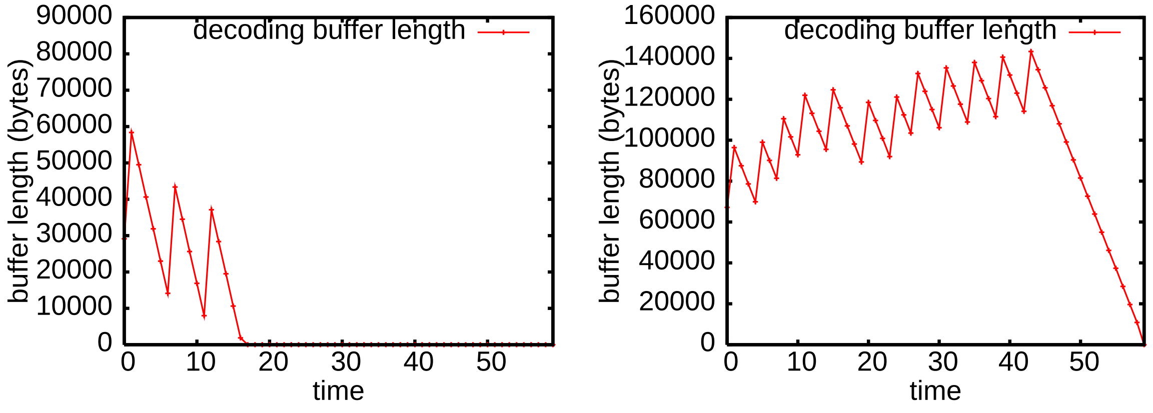


(a) (b)

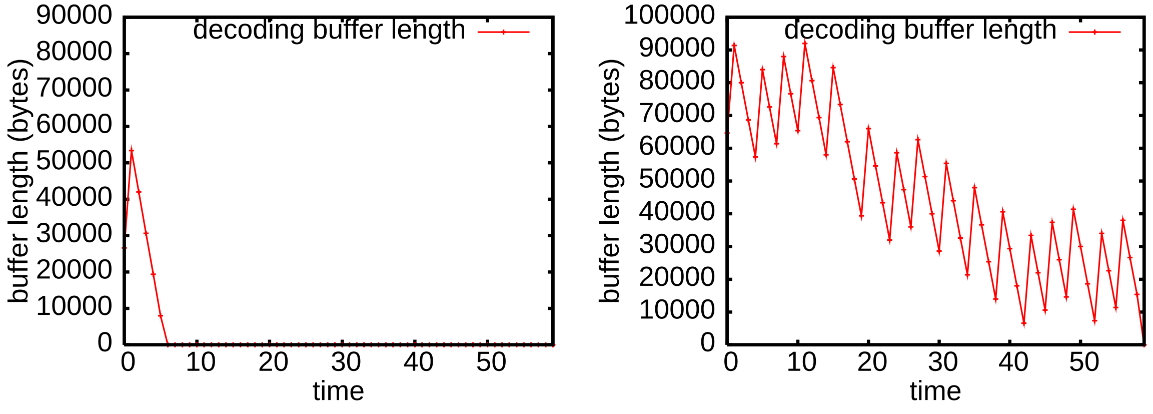
Figure 5.29: Throughput and packet loss details for (a) one TFRC; (b) E-MULTFRC.



(a)



(b)



(c)

Figure 5.30: Throughput and packet loss details for one TFRC (left) and E-MULTFRC (right): the source bit rate is at (a) 50kbps; (b) 70kbps; (c) 90kbps.

Chapter 6

Conclusion and Future Work

This chapter includes concluding remarks and future work.

6.1 Discussion

E-MULTCP, E-MULTFRC, MULTFRC, and E-AIOTFRC are different from existing schemes such as TCP-Vegas [28] and VTP [58] which use delay or round trip time variation to infer congestions in several ways: first both TCP-Vegas and VTP are transport layer solutions, and could solve the wireless problem only if it were deployed universally by every single computer, cell phone, PDA, and handheld device. But the fact is that there are many flavors of TCP deployed today on a heterogeneous set of devices, and trying to make everyone use a new version of TCP is impractical, if not impossible. As such, the strength of our proposed application layer scheme is that it requires absolutely no modification to the transport layer protocol stack, and hence no modifications to the operating systems of the end user devices. It also does not require any changes to the network infrastructure such as routers.

Second, TCP-Vegas needs to measure precise queueing delay every RTT, e.g. typically on the order of tens of milliseconds. It is difficult to measure this accurately in such a short interval, because the large variance in the packet processing time at routers and end-hosts may overwhelm the actual value of queuing delay. In contrast, our proposed scheme is based on only one bit of information to detect existence of queuing delay once in a while, e.g. every 20 seconds in current implementation of E-MULTCP. Such one bit of information is relatively easy to measure accurately.

6.2 Conclusions

In this thesis, we have formulated flow control problem in wireless networks as a general concave optimization problem, of which Kelly's optimization problem can be shown to be a special case. This reformulation results in a new class of end-to-end based solutions, in which an appropriate number of connections are opened at the application layer by the sender. The solutions require only one bit of information on whether or not the route is congested, making it easy to estimate accurately at the application layer. Hence no modification to either existing transport protocols stack, for example TCP, or infrastructure elements, for example routers, is needed. We have shown that our proposed scheme has a unique stable equilibrium that solves the concave optimization problem, implying the stability, scalability and optimality of the solution. Specifically, the unique optimal equilibrium is semi-globally exponentially stable. Stability, optimality and scalability of proposed solution coexisting with TCP can also be inferred in a straightforward fashion.

We have also shown that our proposed solution follows a general framework for flow control. In this framework, solving a flow control problem is interpreted as pursuing a particular equilibrium of users' rates. It is sufficient to achieve this goal by using one control law for the number of connections in a fast timescale, and another control law for sending rate of individual connection in a slow timescale. If both control laws can guarantee exponential convergence to selected equilibria, then in general users' rates will converge to the desired equilibrium exponentially. A significant advantage of this framework is that it is possible to modify a control law in one timescale without affecting the one in the other timescale, while still maintaining convergence properties. Following this framework, we have easily derived a variant of proposed solution with similar convergence performance guarantees.

In practice, these results guarantee all network bottlenecks to be fully utilized, yet the network is free of congestion collapse, for *arbitrary* topology, *arbitrary* initial sending rates, and *arbitrary* number of users applying either proposed solution or TCP. Furthermore, users' rates globally exponentially converge to a unique and optimal equilibrium, resulting in no rate fluctuation in stationary state.

To demonstrate the generality of our proposed solution, we have developed practical scheme E-MULTCP for data transmission over wireless network, and E-MULTFRC, MULTFRC, and E-AIOTFRC for streaming over wireless networks. Among them, E-MULTCP, E-MULTFRC, and E-AIOTFRC use IIMD control law, and are designed based on the proposed solution; MULTFRC uses IIAD control law, and is designed based on the variant of proposed solution. Their efficient performance, and fairness to both themselves and TCP are characterized and evaluated using NS-2 simulations, and experiments over commercial 1xRTT and EVDO wireless networks. Analysis and simulation results indicate these schemes in fact work in both wired and wireless scenarios, and result in anywhere between 25% and 65% throughput improvement over TCP or TFRC in commercial cellular wireless networks.

In addition, Lemma 3.3.4 states that for equilibrium of continuous systems, local exponential stability and global asymptotical stability indicate the semi-global exponential stability. We believe that this lemma is general and as such can be applied to other problems in control for example. We also believe that our functions for applying the La Salle principle, in the proof of Lemma 3.3.3, may provide new insight to searching Lyapunov functions, or functions for the La Salle principle when exploring the stability of general nonlinear systems.

In summary, we have provided promising answers to the two questions we set out to answer in the Chapter 1.1 of this thesis:

Question: *Is it possible to solve the problem of flow control in wireless networks, without changing today's network infrastructure, operating systems, protocol stack, or violating the end-to-end principle?*

It is indeed possible to use the application layer to solve this problem. Several schemes proposed in this thesis, such as E-MULTCP, E-MULTFRC, MULTFRC and E-AIOTFRC, are concrete examples of such possibility, with theoretical and experimental verifications.

Question: *What is the framework for flow control problem in wired or wireless networks, without modifying today's network infrastructure, operating systems, or the protocol stacks?*

In the general framework proposed in Section 3.6, rate of individual connections and the number of connections opened by a user are controlled *independently* in two separate timescales. Hence, it is possible, but not necessary, to use an existing control law such as

TCP for rate of individual connections, and to design new control laws for the number of connections; this results in no modifications to transport layer protocol stacks and network infrastructure elements.

6.3 Future Work

Even though C_j and ϵ_j are assumed to be constant in our analysis in Chapter 3, in some networks such as wireless Local Area Networks (WLAN) and CDMA networks, C_j and ϵ_j might be time varying or even change in a correlated fashion. E-MULTCP, E-MULTFRC, MULTFRC, and E-AIOTFRC adjust the number of connections in a timescale much slower than that of each connection's sending rate, in order to satisfy the two timescale assumption, as well as to reduce the complexity and overhead of opening/closing connections. As a result, in a scenario where C_j and ϵ_j are time varying, by design, they could not react instantaneously. Hence, it is interesting to quantify how fast they can adapt to the changes in C_j and ϵ_j. In practice, our experimental results in this thesis have verified that our proposed schemes work in a cemmercial CDMA network, where the C_j and ϵ_j are typically not constant.

Currently, E-MULTCP, E-MULTFRC, MULTFRC, and E-AIOTFRC rely on accurate detection of queuing delay along the route, which in turn requires knowledge of the propagation delay. In situation where propagation delay is highly volatile, such as in 3G wireless networks that deploy opportunistic scheduling, their performance could potentially degrade due to inaccurate estimation of queuing delay. As discussed in the beginning of Section 5.1.2, even though it is possible to choose a large γ to tolerate inaccuracy in estimating queuing delay, it would be desirable to develop a more reliable way to estimate the congestion status of route, i.e. the one bit of information needed by these practical schemes.

There are many other interesting and important directions for future research. Stability in the presence of delay and noisy disturbances are of great interest, from both control and networking points of view. The fairness properties of the equilibrium rates are also of interest.

Our framework captures the impact of wireless channel packet loss on TCP and TFRC performance over wireless. However, it does not model the timeout effect of TCP and TFRC over wireless possibly caused by link layer local retransmissions or heavy wireless loss, potentially degrading the performance. It is desirable to extend our framework in order to take into account the effect of timeout in wireless networks.

In our current framework, we assume one data transmission consists of two end-hosts and *only* one path between them. In some scenarios, using multiple paths to transmit data between two end-hosts has pronounced advantages in terms of throughput and scalability, shown in many practical peer-to-peer applications and wireless multipath communications. Multipath communication allows the sending rate of a user to be a sum of the sending rates along two or more paths. It is not clear how to define proper fairness among users and hence is not clear how the proposed solution E-MULTCP and the framework can be applied to this scenario.

Last but not the least, it is interesting to examine whether it is possible to use a different utility maximization problem that leads to another fundamentally different solution for the TCP and TFRC over wireless network problem.

Bibliography

[1] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1988, pp. 314–329.

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

[3] M. Chen and A. Zakhor, "Rate control for streaming video over wireless," in *Proc. IEEE INFOCOM*, Hongkong, China, Mar. 2004.

[4] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving tcp performance over wireless links," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 756–769, 1997.

[5] *Draft ITU-T recommendation and final draft international standard of joint video specification*, JVT of ISO/IEC MPEG and ITU-T VCEG Std. ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC, JVTG050, 2003.

[6] H. Balakrishnan and R. Katz, "Explicit loss notification and wireless web performance," in *Proc. of IEEE Globecom Internet Mini-Conference*, Nov. 1998.

[7] D. Barman and I. Matta, "Effectiveness of loss labeling in improving tcp performance in wired/wireless networks," in *Proc. of the 10th ICNP*, Washington, DC, USA, 2002, pp. 2–11.

[8] S. Biaz and N. H. Vaidya, "Discriminating congestion loss from wireless losses using inter-arrival times at the receiver," in *Proc. of IEEE Symposium on Application-specific System and Software Engr. and Techn.*, Richardson,TX, USA, Mar. 1999, pp. 10–17.

[9] N. Samaraweera, "Non-congestion packet loss detection for tcp error recovery using wireless links," *IEE Proceedings of Communications*, vol. 146, no. 4, p. 222C230, Aug. 1999.

[10] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for udp flows by pathstatus classification," in *Proc. of 25th Annual IEEE Conf. on Local Computer Networks*, Tampa,FL, USA, Nov. 2000, p. 252C261.

[11] C. JA and A. P., “Congestion or corruption? a strategy for efficient wireless tcp sessions,” in *Proc. of IEEE Symposium on Computers and Communications*, Los Alamitos, CA, USA, 1995, pp. 262–268.

[12] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, “A wireless transmission control protocol for cdpd,” in *Proc. of IEEE Wireless Communications and Networking Conference*, Piscataway, NJ, USA, Jan. 1999, pp. 953–957.

[13] ——, “Wtcp: a reliable transport protocol for wireless wide-area networks,” *Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.

[14] W. Ding and J. A, “A new explicit loss notification with acknowledgment for wireless tcp,” in *Proc. of 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Piscataway, NJ, USA, 2001, pp. B–65–9.

[15] C. CF and M. M., “Improving tcp over wireless through adaptive link layer setting,” in *Proc. of IEEE Global Telecommunications Conference*, Piscataway, NJ, USA, 2001, pp. 1766–1770.

[16] J.-H. Choi, S.-H. Yoo, and C. Yoo, “A flow control scheme based on buffer state for wireless tcp,” in *Proc. of the 4th International Workshop on Mobile and Wireless Communications Network*, Piscataway, NJ, USA, 2002, pp. 592–596.

[17] K. Ratnam and I. Matta, “Wtcp: an efficient mechanism for improving wireless access to tcp services,” *International Journal of Communication Systems*, vol. 16, no. 1, pp. 47–62, Feb. 2003.

[18] J. Rendon, F. Casadevall, and J. Carrasco, “Wireless tcp proposals with proxy servers in the gprs network,” in *Proc. of 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Piscataway, NJ, USA, 2002, pp. 1156–1160.

[19] Y. Yang, H. Zhang, and K. R., “Channel quality based adaptation of tcp with loss discrimination,” in *Proc. of IEEE Global Telecommunications Conference*, Piscataway, NJ, USA, 2001, pp. 2026–2030.

[20] J.-J. Lee, F. Liu, and K. C-CJ, “End-to-end wireless tcp with non-congestion packet loss detection and handling,” in *Proc. of the SPIE*, San Jose, USA, Jan. 2003, pp. 104–113.

[21] S. Cen, P. Cosman, and G. Voelker, “End-to-end differentiation of congestion and wireless losses,” *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 703–717, 2003.

[22] J. Liu, I. Matta, and M. Crovella, “End-to-end inference of loss nature in a hybrid wired/wireless environment,” in *Proc. WiOpt*, 2003.

[23] T. eun Kim, S. Lu, and V. Bharghavan, “Improving congestion control performance through loss differentiation,” in *Proc. ICPP Workshop*, 1999, pp. 140–145.

[24] C. Parsa and J. Garcia-Luna-Aceves, “Improving tcp congestoin control over internet with heterogeneous media,” in *Proc. ICNP*, 1999, pp. 213–221.

[25] J. Tang, G. Morabito, I. F. Akyildiz, and M. Johnson, "Rcs: A rate control scheme for real-time traffic in networks with high bandwidth-delay products an high bit error rates," in *Proc. IEEE INFOCOM*, Alaska, USA, Apr. 2001, pp. 114–122.

[26] G. Yang, M. Gerla, and M. Y. Sanadidi, "Adaptive video streaming in presence of wireless errors," in *Proc. ACM MMNS*, San Diego, USA, Jan. 2004.

[27] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "End-to-end TCP-Friendly streaming protocol and bit allocation for scalable video over mobile wireless internet," in *Proc. IEEE INFOCOM*, Hongkong, China, Mar. 2004.

[28] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[29] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10–23, Oct. 1994.

[30] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness, and stability," *Journal of the Operationl Research Society*, vol. 49, pp. 237–252, 1998.

[31] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ecn marks," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[32] T. Alpcan and T. Basar, "A game-theoretic framework for congestion control in general topology networks," in *Proc. IEEE CDC*, Las Vegas, NV, Dec. 2002, pp. 1218–1224.

[33] G. Vinnicombe, "On the stability of end-to-end congestion control for the internet," University of Cambridge, Cambridge, UK, Tech. Rep. CUED/F-INFENG/TR.398, 2001.

[34] T. Alpcan and T. Basar, "Global stability analysis of end-to-end congestion control schemes for general topology networks with delay," in *Proc. IEEE CDC*, Maui, Hawaii, Dec. 2003.

[35] R. Johari and D. Tan, "End-to-end congestion control for the internet: delays and stability," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 818–832, Dec. 2001.

[36] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–875, Dec. 1999.

[37] F. Paganini, Z. Wang, J. Doyle, and S. Low, "A new TCP/AQM for stable operation in fast networks," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 2003.

[38] H. Yaiche, R. R. Mazumdar, and C. Rosenberg, "A game-theoretic framework for bandwidth allocation and pricing in broadband network," *IEEE/ACM Trans. Netw.*, pp. 667–678, Oct. 2000.

[39] F. P. Kelly, "Fairness and stability of end-to-end congestion control," *European Journal of Control*, vol. 9, pp. 159–176, 2003.

[40] S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Trans. Netw.*, Aug. 2003.

[41] S. Kunniyur and R. Srikant, "A time scale decomposition approach to adaptive ECN marking," *IEEE Trans. Autom. Control*, vol. 47, no. 6, pp. 882–894, Jun. 2002.

[42] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 556 – 567, Oct. 2001.

[43] M. Chen, A. Abate, and S. Sastry, "New congestion control schemes over wireless networks: Stability analysis," in *Proceedings of the* 16^{th} *IFAC World Congress*, Prague, 2005.

[44] A. Abate, M. Chen, and S. Sastry, "New congestion control schemes over wireless networks: Delay sensitivity analysis and simulations," in *Proceedings of the* 16^{th} *IFAC World Congress*, Prague, 2005.

[45] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 458 – 472, Aug. 1999.

[46] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM*, 1998, pp. 303 – 314.

[47] Network simulation version 2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[48] J. Mahdavi and S. Floyd. (1997, Jan.) TCP-Friendly unicast rate-based flow control. Technical note sent to end2end-interest mailing list. [Online]. Available: http://www.psc.edu/networking/papers/tcp_friendly.html

[49] S. Sastry, *Nonlinear Systems, Analysis, Stability and Control.* New York, NY: Springer Verlag, 1999.

[50] H. Khalil, *Nonlinear Systems (3rd edition).* Prentice Hall, 2001.

[51] Traceroute. [Online]. Available: http://www.traceroute.org/

[52] Kazza. [Online]. Available: http://www.kazza.com

[53] Pplive - internet peer-to-peer video streaming. [Online]. Available: http://www.pplive.com

[54] Brew - binary runtime environment for wireless. [Online]. Available: http://brew.qualcomm.com/brew/en/

[55] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for live media streaming," in *Proc. IEEE INFOCOM*, Miami, FL, USA, Mar. 2005.

[56] D. E. Ott, T. Sparks, and K. Mayer-Patel, "Aggregate congestion control for distributed multimedia applications," in *Proc. IEEE INFOCOM*, Hongkong, China, Mar. 2004.

[57] X. Tong and Q. Huang, "MULTFRC-LERD: An improved rate control scheme for video streaming over wireless," in *Proc. 5th Pacific Rim Conference on Multimedia*, Tokyo, Japan, Nov. 2004, pp. 282–289.

[58] G. Yang, L.-J. Chen, T. Sun, M. Gerla, and M. Y. Sanadidi, "Smooth and efficient real-time video transport in presence of wireless errors," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 2, no. 2, pp. 109–126, May 2006.

[59] *TMN (H.263) encoder/decoder, version 2.0, tmn (h.263) codec*, T. Research Std., 1996.

Appendix A

Proof of Theorem 3.3.1

Theorem 3.3.1: *For arbitrary $\beta > 0$, the approximate system in Equation (3.14) has a unique equilibrium, denoted by (x^*, n^*), given by*

$$\begin{aligned} n_r^* &= \frac{1}{\sqrt{f(\sum_{j\in r} g_j(y_j^*))}}, \qquad r \in R; \\ x_r^* &= \frac{\sqrt{2}S}{\sqrt{f(\sum_{j\in r} g_j(y_j^*))}T_r\sqrt{\sum_{j\in r}[g_j(y_j^*)+\epsilon_j]}}, \quad r \in R. \end{aligned} \tag{A.1}$$

Further, this unique equilibrium solves the following concave optimization problem

$$\max_{x\geq 0} \sum_{r\in R} U_r(x_r) - \sum_{j\in J} \int_0^{y_j} g_j(z)\, dz, \tag{A.2}$$

with U_r being concave function:

$$U_r(x_r) = \int_0^{x_r} h_r^{-1}\left(\frac{2S^2}{T_r^2\nu^2}\right) d\nu, \quad r \in R,$$

where h_r^{-1} is the inverse of a monotonically increasing function h_r:

$$h_r(z) \triangleq \left(\sum_{j\in r} \epsilon_j + z\right) f(z) = \left(\sum_{j\in r} \epsilon_j + z\right)\frac{e^{\beta z}-1}{e^{\beta z}+1}, \quad r \in R.$$

Proof. First note (x^*, n^*) is the equilibrium of the system in Equation (3.14), and x^* is also the solution for the optimization problem in Equation (3.20), which can be seen by setting the derivative to zero. By the definition of $h_r(z)$ and $g_j(z)$, it is not difficult to see the objective function in Equation (3.20) is a concave function. Note the constraint set of the optimization problem shown in Equation (3.20) is convex, and that the optimization problem is in fact a concave optimization problem. Hence it has a unique solution, which is in fact x^*. Since the equilibrium must lie on the equilibrium manifold in Equation (3.17), by Lemma 3.3.1 we know the unique x^* leads to a unique n^*. Therefore, the equilibrium (x^*, n^*) exists and is unique. □

Appendix B

Proof of theorem 3.3.2

Theorem 3.3.2: *For arbitrary $\beta > 0$, under the two timescale assumption, the unique equilibrium of the reduced system in Equation (3.16) is locally exponentially stable. Also, the unique equilibrium of the approximate system in Equation (3.14), (x^*, n^*), is locally exponentially stable.*

Proof. Let $G(x) = diag(\frac{(x_r)^2 T_r^2}{2S^2})$ and

$$D(x) = diag\{\frac{1}{x_r}\sum_{j\in r}[\epsilon_j + g_j(y_j)]\} + \frac{1}{2}A^T diag\{g_j'(y_j)\}A,$$

where A is the connectivity matrix defined in Section 3.1. We can have the relation between $\dot{x}$ and $\dot{n}$ on the equilibrium manifold as:

$$G(x)D(x)\dot{x} = diag\{n_r\}\dot{n},$$

Combining the dynamics of n in the reduced system, we can rewrite the above equation as:

$$\dot{x} = cD^{-1}(x)G^{-1}(x)\{1 - \frac{T_r^2}{2S^2}x_r^2\sum_{j\in r}[\epsilon_j + g_j(y_j)]f(\sum_{j\in r} g_j(y_j))\}.$$

Around the equilibrium of the reduced system, let $x_r(t) = x_r^* + z_r(t)$, denote $D(x^*)$ as $\tilde{D}$ and $G(x^*)$ as $\tilde{G}$; after linearization, we have that, $\forall r \in R$,

$$\begin{aligned}\dot{z}(t) \;\; = \;\; & -c\tilde{D}^{-1}\tilde{G}^{-1}\left[2diag\left(f(\sum_{j\in r} g_j(y_j^*))\right)\tilde{G}\tilde{D} + \right.\\ & \tilde{G}\cdot diag\big(\sum_{j\in r}\big(\epsilon_j + g_j(y_j^*)\big)\big)\cdot \\ & \left. diag\left(f'(\sum_{j\in r} g_j(y_j^*))\right)A^T diag(g_j'(y_j^*))A\right] z(t), \qquad \text{(B.1)}\end{aligned}$$

where $g_j'(y_j^*) = \frac{C_j}{(y_j^*)^2} \frac{e^{\beta \frac{y_j^*-C_j}{y_j^*}}}{1+e^{\beta \frac{y_j^*-C_j}{y_j^*}}} > 0, \quad j \in J.$

Denote

$$E = 2diag\left(f(\sum_{j\in r} g_j(y_j^*)) \right) \tilde{G}\tilde{D} + \tilde{G} \cdot diag\big(\sum_{j\in r} \left(\epsilon_j + g_j(y_j^*)\right) \big) diag\left(f'(g_j(y_j^*)) \right) A^T diag(g_j'(y_j^*))A.$$

Then by simple linear algebra arguments, the system in Equation (B.1) is stable if and only if $\tilde{D}^{-1}\tilde{G}^{-1}E$ has all positive eigenvalues. We now show that this requirement is verified.

First note that this is equivalent to showing $E\tilde{D}^{-1}\tilde{G}^{-1}$ has all eigenvalues be positive since $E\tilde{D}^{-1}\tilde{G}^{-1}$ is similar to $\tilde{D}^{-1}\tilde{G}^{-1}E$. But

$$\begin{aligned} E\tilde{D}^{-1}\tilde{G}^{-1} = 2\tilde{G} \cdot diag &\left(f'(\sum_{j\in r} g_j(y_j^*)) \frac{[\sum_{j\in r}(\epsilon_j + g_j(y_j^*))]^2}{x_r^*} \right) \\ \cdot \Bigg\{ \cdot diag &\left(\frac{x_r * f(\sum_{j\in r} g_j(y_j^*))}{f'(\sum_{j\in r} g_j(y_j^*))[\sum_{j\in r}(\epsilon_j + g_j(y_j^*))]^2} \right) \\ + \frac{1}{2} diag &\{ \frac{x_r^*}{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]} \} A^T diag(g_j'(y_j^*)) A\tilde{D}^{-1} \Bigg\} \tilde{G}^{-1}, \end{aligned}$$

Define the terms inside the curly brackets as B for later usage. $A^T diag(g_j'(y_j^*))A\tilde{D}^{-1}$ is a product of a positive definite matrix and a (semi)-positive definite matrix, having all non-negative eigenvalues. On the other hand, $A^T diag(g_j'(y_j^*))A = 2(\tilde{D} - diag\{\frac{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]}{x_r^*}\})$; hence

$$\begin{aligned} A^T diag(g_j'(y_j^*))A\tilde{D}^{-1} &= 2 \cdot diag\{ \frac{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]}{x_r^*} \} \cdot \\ &\quad \left[diag\{ \frac{x_r^*}{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]} \} - \tilde{D}^{-1} \right]. \end{aligned}$$

Now $diag\{\frac{x_r^*}{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]}\} - \tilde{D}^{-1} \succeq 0$. This can be shown by contradiction. Suppose the contrary; left multiple the above equation by $\left(diag\{\frac{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]}{x_r^*}\}\right)^{-1/2}$, and right multiple the above equation by $\left(diag\{\frac{\sum_{j\in r}[\epsilon_j + g_j(y_j^*)]}{x_r^*}\}\right)^{1/2}$; then the left hand side has all nonnegative eigenvalues, while the right hand side is a non-semi-positive definite matrix with at least one eigenvalue being negative, resulting in contradiction.

We claim $E\tilde{D}^{-1}\tilde{G}^{-1}$ has all positive eigenvalues, due to the following three facts:

- $B \succ 0$ since it is a sum of two positive definite matrices.
- $diag\left(f'(\sum_{j\in r} g_j(y_j^*)) \frac{[\sum_{j\in r}(\epsilon_j + g_j(y_j^*))]^2}{x_r^*} \right) B$ has all positive eigenvalues, because it is the product of two positive definite matrices;

- $E\tilde{D}^{-1}\tilde{G}^{-1}$ has all positive eigenvalues, because it is similar to $diag\left(f'(\sum_{j\in r} g_j(y_j^*))\frac{[\sum_{j\in r}(\epsilon_j+g_j(y_j^*)]^2}{x_r^*}\right)B$.

Eventually, $\tilde{D}^{-1}\tilde{G}^{-1}E$ has all positive eigenvalues and hence the reduced system in Equation (B.1) is locally exponentially stable for arbitrary $\beta > 0$.

Hence combining the fact that the boundary system is locally exponentially stable, we conclude the entire system is locally exponentially stable, according to Theorem 11.4 in [50]. □

Appendix C

Proof of Lemma 3.3.2

Lemma 3.3.2: *There exists a compact set, denoted by Ω_1, for $n(t)$ in the reduced system in Equation (3.16) with arbitrary $\beta > 0$, such that any compact set containing it is a positively invariant one. A positive invariant set is a set with all trajectories on its boundary pointing inwards; as such, no trajectories inside the set will ever move out. The same observation is also true for $x(t)$ in the boundary layer system Equation (3.15), and the corresponding compact set is defined as $\Omega_2(n)$, a function of n.*

Proof. To construct the compact set Ω_1 for $w_r(t)$ in the reduced system, first note

a) if route r is not congested, then $g_j(y_j(t)) \leq \frac{1}{\beta} \ln 2$; so

$$x_r(t) \geq \frac{n_r(t)S}{T_r\sqrt{\sum_{j\in r}(\epsilon_j + \frac{1}{\beta}\ln 2)}}.$$

As $n_r(t)$ increases, $x_r(t)$ will eventually hit $\min_{j\in r} C_j$ and route r becomes congested (cross traffic only helps to cause congestion). Hence, if $n_r(t)$ is sufficiently large, route r will be congested.

b) if route r is congested, then we must have $\sum_{j\in r} g_j(y_j(t)) > \frac{1}{\beta}\ln 2$. Together with the fact $f(\sum_{j\in r} g_j(y_j(t)))$ is a nondecreasing function of $\sum_{j\in r} g_j(y_j(t))$, we have

$$f(\sum_{j\in r} g_j(y_j(t))) \geq 2\frac{e^{\ln 2}}{1+e^{\ln 2}} - 1 = \frac{1}{3}.$$

Hence as long as route r is congested, we have

$$\dot{n}_r(t) = c\left(\frac{1}{n_r(t)} - n_r(t)f(\sum_{j\in r} g_j(y_j(t)))\right) < c\left(\frac{1}{n_r(t)} - \frac{1}{3}n_r(t)\right)$$

There must exist one n_r^{max} such that $\dot{n}_r^{max}(t) < 0$.

Therefore, there exists large enough n_r^{max} such that route r is congested regardless of cross traffic, i.e. even only user r is active and others are inactive, and $\dot{n}_r^{max}(t) < 0$. Eventually, Ω_1 can be defined as

$$\Omega_1 = [0, n_1^{max}] \times [0, n_2^{max}] \cdots [0, n_N^{max}].$$

It is easy to check the flow of the vector field satisfies:

- if $n_r(t) \leq 0$, $\dot{n_r}(t) > 0$ according to Equation (3.16);
- if $n_r(t) \geq n_r^{max}$, then by n_r^{max}'s definition, $\dot{n_r}(t) \leq 0$.

Therefore, on the boundary of any compact set containing Ω_1, vector field defined in Equation (3.16) points inward. Hence by definition, the compact set is positive invariant.

Similarly, a compact set $\Omega_2(n)$ can be defined for $x(t)$ in boundary layer system in Equation (3.15), as follows:

$$\Omega_2(n) = [0, x_1^{max}(n_1)] \times [0, x_2^{max}(n_2)] \cdots [0, x_N^{max}(n_N)],$$

where $x_r^{max}(n_r) = \frac{n_r \sqrt{2S}}{T_r \sqrt{\sum_{j \in r}(\epsilon_j + g_j(x_r^{max}))}}$ satisfies that given n_r and regardless of cross traffics along route r. If $x_r(t) \geq x_r^{max}(n_r)$, then $\dot{x}_r(t) \leq 0$.

From networking point of view, containing Ω_1 implies the number of connections of each user should be allowed to take large enough values to fully utilize his bottleneck, even if only he is active. Similarly, containing $\Omega_2(n)$ means sending rate of each user can be sufficiently large to achieve its equilibrium. These requirements are trivial to meet in practice. $\square$

Appendix D

Proof of Lemma 3.3.3

Lemma 3.3.3: *The unique equilibrium of reduced layer system in Equation (3.16), with arbitrary $\beta > 0$, is a globally asymptotically stable one.*

Proof. Define $z_r = \frac{4S^2 n_r^2}{x_r^2 T_r^2}$, and $\rho_r = \frac{4S^2}{T_r^2}$, for $r \in R$. We have

$$\begin{aligned}\dot{Z} &= diag(\frac{2\rho_r}{x_r^2})diag(n_r)\dot{n} - diag(\frac{2\rho_r n_r^2}{x_r^3}) \cdot \\ &\quad D^{-1}diag(\frac{2\rho_r}{x_r^2})diag(n_r)\dot{(n)} \\ &= diag(n_r\sqrt{z_r})diag(\frac{2\rho_r}{x_r^2})\Lambda diag(\frac{2\rho_r}{x_r^2})diag(n_r)\dot{n} \qquad \text{(D.1)}\end{aligned}$$

where $\Lambda \triangleq diag(x_r^3/2n_r^2\rho_r) - D^{-1}$ has been shown in Appendix B to be a semi-positive definite matrix.

Define nondecreasing functions $\phi_r(y)$ and $\varphi_r(y), r \in R$ as follows:

$$\phi_r(y) = \int_{\epsilon_r}^{f^{-1}(y)+\epsilon_r} \frac{1}{\sqrt{y}}dy, \quad r \in R \qquad \text{(D.2)}$$

$$\varphi_r(y) = \int_{\epsilon_r}^{f^{-1}(y)+\epsilon_r} \frac{f(y-\epsilon_r)}{\sqrt{y}}dy, \quad r \in R \qquad \text{(D.3)}$$

The following is a La Salle function for the reduced system:

$$\begin{aligned}V(n,z) &= -\sum_{r\in R} c_r \left[\int_{\epsilon_r}^{z_r} \frac{1}{\sqrt{y}n_r}dy - \int_{\epsilon_r}^{z_r} \frac{n_r}{\sqrt{y}} f(y-\epsilon_r)dy \right. \\ &\quad \left. + \int_0^{n_r} \frac{1}{y^2}\phi_r(\frac{1}{y^2})dy + \int_0^{n_r} \varphi_r(\frac{1}{y^2})dy\right].\end{aligned}$$

Its Lee derivative is

$$\begin{aligned}
\dot{V}(n,z) &= -\sum_{r\in R} c_r \left(\frac{1}{\sqrt{z_r} n_r} - \frac{n_r^2}{\sqrt{z_r} n_r} f(z_r - \epsilon_r) \right) \dot{z}_r - \\
&\quad \sum_{r\in R} c_r \frac{\dot{n}_r}{n_r^2} \left[\phi_r(\frac{1}{n_r^2}) - \phi_r(f(z_r - \epsilon_r)) \right] - \\
&\quad \sum_{r\in R} c_r \dot{n}_r \left[\varphi_r(\frac{1}{n_r^2}) - \varphi_r(f(z_r - \epsilon_r)) \right] \\
&= -\dot{n} diag(n_r) diag(\frac{2\rho_r}{x_r^2}) \Lambda \, diag(\frac{2\rho_r}{x_r^2}) diag(n_r) \dot{n} - \\
&\quad \sum_{r\in R} c_r \frac{\dot{n}_r}{n_r^2} \left[\phi_r(\frac{1}{n_r^2}) - \phi_r(f(z_r - \epsilon_r)) \right] - \\
&\quad \sum_{r\in R} c_r \dot{n}_r \left[\varphi_r(\frac{1}{n_r^2}) - \varphi_r(f(z_r - \epsilon_r)) \right] \qquad \text{(D.4)} \\
&\leq 0.
\end{aligned}$$

The equality is taken only when $\dot{n} = 0$, i.e. at the equilibrium. Therefore, the system is globally asymptotically stable.

□

Appendix E

Proof of Lemma 3.3.4

Lemma 3.3.4: *Consider a system $\dot{\xi} = \varphi(\xi, t, \xi_0)$ satisfying the following assumptions:*

- *it has a unique equilibrium at* 0 *that is locally exponentially stable and globally asymptotically stable;*
- $\varphi(\xi, t, \xi_0)$ *is continuous.*

Then the equilibrium of the system is semi-globally exponentially stable.

Proof. By definition of local exponential stability, there exists a $r > 0$ s.t.

$$\|\xi(t, \xi_0)\| \leq K\|\xi_0\|e^{-\gamma t}, \quad \forall \|\xi_0\| \leq r,$$

where K, r and $\gamma > 0$ are constant. Without loss of generality, we assume $K > 1$ and $r < 1$.

Define $T_r(\xi_0) = \inf\{t \geq 0 : \|\xi(t, \xi_0)\| \leq r\}$. By definition of global asymptotical stability, $T_r(\xi_0) < \infty$.

Since $\xi(t, \xi_0)$ is continuous with respect to t, let $M_r(\xi_0) = \max\{K, \xi(t, \xi_0) : 0 \leq t \leq T_r\} < \infty$.

Define $L = M_r(\xi_0)e^{\gamma T_r(\xi_0)}/r$, we claim

$$\|\xi(t, \xi_0)\| \leq L\|\xi_0\|e^{-\gamma t}. \tag{E.1}$$

To see this, we study two cases:

- if $\|\xi_0\| \leq r$, then $T_r(\xi_0) = 0$, by setting, we already have $\|\xi(t, \xi_0)\| \leq K\|\xi_0\|e^{-\gamma t} \leq L\|\xi_0\|e^{-\gamma t}$;
- if $\|\xi_0\| > r$, the following is true for $t \in [0, T_r(\xi_0)]$:

$$\begin{aligned} L\|\xi_0\|e^{-\gamma t} &= M_r(\xi_0)\frac{\|\xi_0\|}{r}e^{\gamma T_r(\xi_0)}e^{-\gamma t} \\ &\geq M_r(\xi_0) \geq \|\xi(t, \xi_0)\|. \end{aligned}$$

  For $t \geq T_r(\xi_0)$, let $t' = t - T_r$, we have $\|\xi(t' = 0)\| \leq r$. Applying the first case's result concludes our claim.

Therefore, by Definition 5.10 in [49], Equation (E.1) implies semi-global exponential stability of the equilibrium. The semi-global term is due to L's dependency on initial condition ξ_0. $\square$

Appendix F

Proof of Theorem 3.3.3

Theorem 3.3.3: *The unique equilibrium of singularly perturbed system in Equation (3.14) with arbitrary $\beta > 0$ is semi-globally exponentially stable.*

Proof. We have shown that the reduced system has a unique equilibrium (Theorem 3.3.1) that is globally asymptotically stable (Lemma 3.3.3) and locally exponentially stable (Theorem 3.3.2). Hence by Lemma 3.3.4, we conclude the unique equilibrium of reduced system in Equation (3.16) is semi-globally exponentially stable.

Similar arguments are also true for boundary layer system. It has a unique equilibrium that is globally asymptotically stable and locally exponentially stable [30]. Hence, again by Lemma 3.3.4, we conclude the unique equilibrium of boundary layer system in Equation (3.16) is semi-globally exponentially stable.

If we constrain $n(t)$ to a compact set containing Ω_1, denoted by Σ_1 and let $n_r^{max} = \max(n_r : n_r \in \Sigma_1)$, then any compact set containing $\Omega_2(n^{max})$ is positive invariant, following the arguments in Appendix C. Constrain $x(t)$ to a compact set containing $\Omega_2(n^{max})$, denoted by Σ_2.

On $\Sigma_1 \times \Sigma_2$, the equilibrium of the reduced system in Equation (3.16) is exponentially stable, and the one of the boundary layer system in Equation (3.15) is exponential stable uniformly in n (verification is the same as in Appendix II.F, [41]).

Together with the fact $\Sigma_1 \times \Sigma_2$ is a positive invariant, by Theorem 11.4 in [50], we conclude that the singularly perturbed system in Equation (3.14) has a unique equilibrium that is exponentially stable in $\Sigma_1 \times \Sigma_2$. Hence, the equilibrium of singularly perturbed system in Equation (3.14) is semi-globally exponentially stable. $\square$

Appendix G

Proof of Theorem 3.7.1

Theorem 3.7.1: *For arbitrary $\beta > 0$, the approximate system in Equation (3.24) has a unique equilibrium, denoted by (x^*, n^*) as*

$$\begin{aligned} n_r^* &= \tfrac{1}{f(\sum_{j\in r} g_j(y_j^*))}, \qquad r \in R; \\ x_r^* &= \tfrac{\sqrt{2}S}{f(\sum_{j\in r} g_j(y_j^*))T_r\sqrt{\sum_{j\in r}\left[g_j\left(y_j^*\right)+\epsilon_j\right]}}, \quad r \in R. \end{aligned} \tag{G.1}$$

Further, this unique equilibrium solves the following concave optimization problem

$$\max_{x\geq 0} \sum_{r\in R} U_r(x_r) - \sum_{j\in J} \int_0^{y_j} g_j(z)\, dz, \tag{G.2}$$

with U_r being concave function:

$$U_r(x_r) = \int_0^{x_r} h_r^{-1}\left(\frac{2S^2}{T_r^2\nu^2}\right) d\nu, \quad r \in R, \tag{G.3}$$

where h_r^{-1} is the inverse of a monotonically increasing function h_r:

$$h_r(z) \triangleq \left(\sum_{j\in r} \epsilon_j + z\right) f^2(z) = \left(\sum_{j\in r} \epsilon_j + z\right)\left(\frac{e^{\beta z}-1}{e^{\beta z}+1}\right)^2, \quad r \in R.$$

Proof. First note (x^*, n^*) is the equilibrium of the system in Equation (3.24), and x^* is also the solution for the optimization problem in Equation (3.28), which can be seen by setting the derivative to zero. By the definition of $h_r(z)$ and $g_j(z)$, it is not difficult to see the objective function in Equation (3.28) is a concave function. Note the constraint set of the optimization problem shown in Equation (3.28) is convex, and that the optimization problem is in fact a concave optimization problem. Hence it has a unique solution, which is in fact x^*. Since the equilibrium must lie on the equilibrium manifold in Equation (3.17), by Lemma 3.3.1 we know the unique x^* leads to a unique n^*. Therefore, the equilibrium (x^*, n^*) exists and is unique. □