A User-level, Reliable and Reconfigurable Transport Layer Protocol

by

Tan Wang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2009

AUTHOR'S DECLARATION

Abstract

Over the past 15 years, the Internet has proven itself to be one of the most influential inventions that humankind has ever conceived. The success of the Internet can be largely attributed to its stability and ease of access. Among the various pieces of technologies that constitute the Internet, TCP/IP can be regarded as the cornerstone to the Internet's impressive scalability and stability. Many researchers have been and are currently actively engaged in the studies on the optimization of TCP's performance in various network environments.

This thesis presents an alternative transport layer protocol called RRTP, which is designed to provide reliable transport layer services to software applications. The motivation for this work comes from the fact that the most commonly used versions of TCP perform unsatisfactorily when they are deployed over non-conventional network platforms such as cellular/wireless, satellite, and long fat pipe networks. These non-conventional networks usually have higher network latency and link failure rate as compared with the conventional wired networks and the classic versions of TCP are unable to adapt to these characteristics. This thesis attempts to address this problem by introducing a user-level, reliable, and reconfigurable transport layer protocol that runs on top of UDP and appropriately tends to the characteristics of non-conventional networks that TCP by default ignores. A novel aspect of RRTP lies in identifying three key characteristic parameters of a network to optimize its performance.

The single most important contribution of this work is its empirical demonstration of the fact that parameter-based, user-configurable, flow-control and congestion-control algorithms are highly effective at adapting to and fully utilizing various networks. This fact is demonstrated through experiments designed to benchmark the performance of RRTP against that of TCP on simulated as well as real-life networks. The experimental results indicate that the performance of RRTP consistently match and exceed TCP's performance on all major network platforms. This leads to the conclusion that a user-level, reliable, and reconfigurable transport-layer protocol, which possesses the essential characteristics of RRTP, would serve as a viable replacement for TCP over today's heterogeneous network platforms.

Acknowledgments

I would like to express my deep gratitude to my supervisor, Professor Ajit Singh, for his continual guidance, support, and encouragement throughout my undergraduate and graduate studies. I feel extremely fortunate to have met Professor Singh at a very early stage of my research career and benefited from his advices and wisdom for the past five years. I would also like to thank Professor Singh for the generous funding that he has provided for this piece of research.

As the readers for this Master’s thesis, Professor Naik and Professor Yang both offered their invaluable insights and advices. I would like to thank both of them for the important roles that they have played throughout my graduate studies.

Furthermore, I would like to thank the entire staff of the department of electrical and computer engineering at the University of Waterloo for their continual help and support.

Last but not least, I am deeply grateful for the kind support that I have received from my friends and families.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Over the past decade, non-conventional network platforms such as cellular/wireless, satellite, and high bandwidth high latency networks, have become increasingly commonplace. The driving force behind the proliferation of these non-conventional network platforms is the enormous growth of the Internet and the diversification of the needs of the Internet users. As pervasive computing technologies continue to be advocated by hardware and software vendors and embraced by corporate as well as individual users, software applications are expected to operate over increasingly heterogeneous network environments and their performance often critically depends on their adaptability to the varying network conditions.

Presently, TCP is the default protocol used by software applications for reliable data transfer over most types of computer networks. Over the past 35 years of its existence, TCP has been proven to work well over conventional wired networks. However, most versions of TCP, including the most widely deployed version – TCP Reno, fail to offer competitive performance over certain non-conventional network platforms. This thesis presents a user-level, reliable and reconfigurable transport layer protocol, named RRTP, which offers significant performance improvement over TCP on non-conventional network platforms while still providing competitive performance over traditional network platforms. The performance characteristics of RRTP are empirically studied through an extensive set of experiments conducted in both simulated and real-life network environments.

1.1 Problem Definition

TCP has been the dominant transport layer protocol used for reliable network computing for many years. However, being originally designed for steady bandwidth, low latency, and low bit error rate (BER) networks, traditional TCP (refers to TCP Tahoe or TCP Reno) adapts ineptly to networks with moderate to high BER, high delay latency, as well as fluctuating and scarce bandwidth. Over the past three decades, researchers have investigated various approaches for enhancing the performance of TCP. In particular, they have been interested in improving TCP's performance over wireless and long fat pipe network platforms. Traditional TCP faces a few critical challenges when deployed over these two types of networks. As it was pointed out in Balakrishnan et al. [1997], TCP treats all packet losses as signs of network congestion and reacts by invoking its effective yet inefficient congestion control mechanisms. Consequently, deploying TCP over wireless network, where losses due to wireless link failures and handoffs are more commonplace than congestion-related losses, inevitably results in poor performance. The cellular wireless network platform holds another major hurdle for TCP to overcome: since TCP's congestion control mechanism is highly sensitive to transmission timeouts, handoffs that are regularly encountered by mobile hosts on cellular networks have detrimental effects on TCP's performance. Packet loss is also detrimental to TCP's performance over long fat pipe networks. As it was mentioned in Jacobson et al. [1992], even with the help of the Fast Retransmit and Fast Recovery algorithms, multi-packets losses would still trigger TCP's slow-start mechanism, consequently draining the data pipeline and resulting in lower overall throughput.

Despite the fact that many researchers have previously attempted to address TCP's performance problems from various angles, there is still a lack of unified solutions that offer competitive, if not superior performance over all major network platforms. Moreover, most of the previously proposed TCP variants require some kernel-level software changes to be made to the existing protocol stack, thereby significantly reducing the adoptability and backward compatibility of these solutions. Drawing from the previous observations, it can be stated that one of the most difficult and desirable problems to be solved in the area of reliable transport protocol design is to create an unified solution that enhances TCP's performance over all major network platforms while leaving zero footprints (i.e. requiring no changes) on the existing software infrastructure. This thesis presents a potential solution to this problem and the necessary empirical support for the performance claims made in this study.

1.2 Main Contributions

Before diving into the main body of this thesis, an outline is provided below to highlight the two main contributions of this piece of research work:

- Identification of a set of user-configurable network parameters that can be pre-configured for tuning the protocol to quickly achieve close-to-optimal performance.
- Demonstration of a user-level transport layer protocol that outperforms TCP over a variety of network platforms, both in simulations and in real-life tests.

These two contributions will become more evident as the readers proceed through this thesis. It is important for the readers to keep in mind a key difference between this piece of research and most of the past related research work. In the field of network protocol research, most scholars take a mathematically rigorous and/or simulation-based approach to support the causal claims made in their studies. In contrast, this piece of research relies on an extensive amount of empirical tests carried out on real physical networks to support the performance claims made in this study. It is the author's belief that tests carried out using a physical implementation of the proposed protocol add realism to the research findings and contribute to the external validity of this research. More in-depth discussions on this topic will be provided in the later chapters of this thesis.

1.3 Thesis Organization

This thesis is organized in the following manner: the first chapter states the core problem to be solved and highlights the main contributions of this piece research; the second chapter provides the necessary background information for this study as well as an overview of previously proposed TCP enhancement solutions; the third chapter discusses the detailed design of RRTP; the fourth chapter presents a set of experiments designed to benchmark RRTP's performance against TCP's performance over a group of representative network platforms; the fifth chapter discusses a few key issues regarding the design of RRTP as well as several important limitations of this piece of research; the final chapter summarizes the key results and discusses a number of open areas for future research.

Chapter 2
Overview of Related Works

Over the past two decades, a large body of research works has emerged on the topic of improving TCP's performance over the various mainstream network platforms. These solutions are created for integration at different layers within the OSI protocol stack, ranging from the link layer to the transport layer. Basing on their design approaches, the proposed solutions can be classified into three categories: link-layer, split-connection, and end-to-end solutions. While link-layer and split-connection solutions primarily focus on the enhancement of TCP's performance over cellular wireless networks, end-to-end solutions are more general and apply to other types of non-conventional networks. This chapter first presents a brief overview of the characteristics of traditional TCP and the network platforms on which it performs poorly. Then, each category of the previously proposed TCP enhancement solutions is examined in considerable details. Towards the end of this chapter, a list of desirable characteristics of an ideal TCP replacement protocol is identified and a list of design criteria for RRTP is outlined.

2.1 Traditional TCP

TCP is a reliable, end-to-end, connection-oriented protocol. It was originally designed to provide reliable virtual circuits between endpoints over networks comprising of wired links and stationary computers. Early versions of TCP depended on a simple sliding window based flow control mechanism to guarantee reliability. The concept of TCP window refers to the maximum number of packets that can be transmitted onto the network by the TCP sender without receiving the corresponding acknowledgments (ACK's). Furthermore, one can obtain a rough estimate of the network throughput by dividing the round trip time (RTT, the time it takes for a packet to travel a round trip between the sender and the receiver) into the TCP window size. Consequently, the TCP sender is able to regulate the level of congestion within a given network through controlling the size of the TCP window.

Jacobson [1988] introduced several congestion control and avoidance mechanisms into TCP after he observes a series of large-scaled congestive network collapse in the 1980's. This new version of

TCP, named TCP Tahoe, uses an adaptive window-based flow/congestion control mechanism that follows the linear increase multiplicative decrease (LIMD) paradigm to regulate network traffic flows. The TCP sender adjusts the size of the sending window in accordance to the feedbacks (in the form of cumulative acknowledgments) that it gets from the TCP receiver. The most common problem that TCP encounters over wired links is packet loss due to path congestions. To address this problem, TCP employs a congestion control mechanism that treats every packet loss as a sign of congestion (Allman et al. [1999]) and takes corresponding measures to resolve the congestion. At the start of a connection, the TCP sender first enters the slow-start phase by setting its send window to be of size 1. Subsequently, for each properly received cumulative acknowledgment, the sender exponentially increases the send rate until it reaches the slow-start threshold (ssthresh). Then, the sender enters the congestion avoidance phase in which it additively increase the size of its congestion window (cwnd) until either the size of the send window matches the maximum window size advertised by the receiver or the sender detects signs of path congestion.

The TCP sender enters the packet retransmission phase upon detecting signs of congestion, which could be either a retransmit timeout or the arrival of a triple duplicated acknowledgment. In the first scenario, TCP maintains a running average of the RTT as well as an expected deviance from this running average. Using these two figures, the TCP sender can heuristically determine whether a sent packet was lost or not. If the current RTT value is more than four times the deviance larger than the running average, then the corresponding packet is assumed to be lost and the sender would retransmit the lost packet.

Furthermore, the TCP sender also treats duplicated acknowledgments as signs of packet losses. When multiple cumulative acknowledgments with the same sequence number is received by the sender, the fast retransmit mechanism is triggered to resend all packets in the send queue that have sequence numbers larger or equal to the one advertised in the cumulative acknowledgment.

At the end of the packet retransmission phase, the TCP sender reduces the value of ssthresh to half of the current cwnd and resets the size of its send window to 1. The congestion control algorithm re-enters the slow-start phase to facilitate the recovery process from network congestion. TCP Tahoe's congestion control algorithm is depicted in Figure 2.1 shown below:

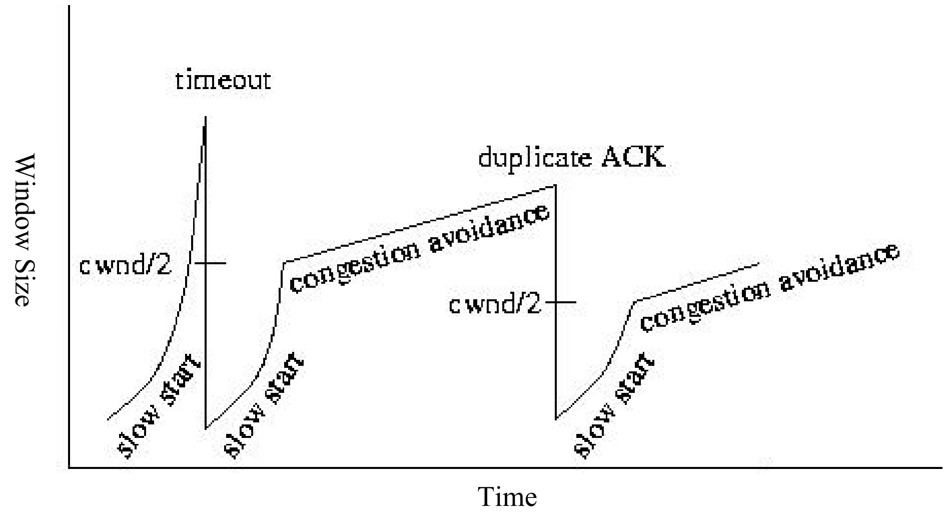


Figure 2.1: TCP's Congestion Control Mechanism

Over the past thirty years, several request for comments (RFC's) have been created to document the various aspects of TCP. The basic features of TCP are described in Postel [1981] [RFC793], Braden [1989] [RFC1122], Allman et al. [1999] [RFC2581], and Floyd [2000] [RFC2914], while the TCP extensions are recommended by Jacobson et al. [1992] [RFC1323], Mathis et al. [1996] [RFC2018], Ramakrishnan and Floyd [1999] [RFC2481], and Floyd et al. [2000] [RFC2883]. Additional background details of the TCP protocol can be found in Stevens and Wright [1994].

2.2 Characteristics of Wireless Network

As it was alluded in chapter one, the wireless network presents several challenges for traditional TCP to enjoy good performance. These barriers to good performance include: high link failure rate due to radio propagation impairments, frequent connection interruptions, scarce and fluctuating bandwidth, limited power supply, as well as dynamic network topology. Section 2.2.1-2.2.5 will explain these five barriers respectively.

2.2.1 High Link Failure Rate

The wireless medium is inherently less reliable than its wired counterpart. The bit error rate (BER) of a typical wireless network can be as high as 10^{-5}. Even minor degradations over the wireless interface such as fast fading can cause packet losses, which in turn are mistakenly interpreted as signs of congestion by TCP. Consequently, it is expected that medium failures, instead of network congestions, cause most of packet losses over wireless networks. However, traditional TCP is unable to distinguish between wireless and congestive packet losses. This shortcoming directly leads to TCP's poor performance over wireless network mediums.

2.2.2 Frequent Connection Interruption

For wireless networks, especially in the case of cellular networks, ongoing network connections could be interrupted or dropped in the following scenarios:

1. A cellular handoff occurs when a mobile device moves from cell A to cell B. The wireless connection between the mobile device and the base station of cell A is migrated to the base station of cell B. Although the duration of a handoff is usually quite short, the brief disconnection inevitably causes packet losses over the wireless medium.
2. Depending on the coverage of a cellular network, a mobile device operating in rural areas may move out of the reaches of all nearby base station transceivers. When this occurs, the mobile device will remain disconnected until it moves back into the range of one or more transceivers. Depending on the duration of the connection disruption, significant amount of packets may be lost.
3. Even when a mobile device is well within the reaches of multiple base station transceivers, signal fading could occur in the presence of large blockage objects such as office buildings. Brief periods of network disconnection could occur as the mobile device move through spots where the strength of wireless signals approaches zero.
4. Connection interruptions may also result from overcapacity of a cell. As a cell becomes jammed with users, the bandwidth per user may become so low that certain user applications perceive the network connection to have effectively been interrupted.

2.2.3 Fluctuating Bandwidth

As it was mentioned in the previous paragraph, the available network bandwidth per user fluctuates as the number of users within a certain cell changes over time. As a result, a transport layer protocol ideally should take this fact into account when trying to achieve optimal performance.

2.2.4 Limited Power Supply

The battery lives of mobile devices are usually around a few hours of continuous usage. Unnecessary retransmissions will inevitably shorten the amount of usage time that one can get out of a single charge-discharge cycle. Therefore, a transport layer protocol ideally should maximize the efficiency of packet transmission in order to keep its energy requirement at a minimum.

2.2.5 Dynamic Network Topology

As the mobile device moves among the vicinities of different wireless networks, the topology of the network between the mobile device and its corresponding fixed host may change quite rapidly. In order to achieve superior performance when operating over a mesh of dynamic and heterogeneous network platforms, the transport layer protocol should be able to intelligently adapt to the changing network topology and its corresponding characteristics.

2.3 Characteristics of Long Fat Pipe Networks

As it was previously mentioned in chapter one, traditional TCP is subjected to performance degradation when deployed over long fat pipe networks (LFP). Also known as high bandwidth high delay networks, LFP's are typically characterized by their large bandwidth delay product, possessing both high bandwidth and large end-to-end latency. Two examples of LFP's are terrestrial fiber-optical links and high capacity satellite data channels, both of which have bandwidth delay product on the order of 10^6 bits.

Traditional TCP performs poorly over LFP's because its flow-control and congestion-control mechanisms are incompatible with the large end-to-end latency. TCP's native response to a packet loss is the throttling of its send rate. This reaction along with the large delay between the sender and the receiver inevitably lead to the drainage of the corresponding data pipeline followed by the commencement of a new slow-start phase. Clearly, frequent slow-starts greatly undermine the proper utilization of the available bandwidth, making traditional TCP ill-suited for high-bandwidth, high-delay networks.

Thus far, an overview of TCP and its performance limitations over non-conventional networks has been presented. Section 2.4-2.7 will examine previously proposed solutions for addressing the various shortcomings of TCP.

2.4 Link Layer Solutions

In the past, several link layer solutions have been proposed to address the poor performance that TCP exhibits over wireless-last-hop networks. These solutions essentially follow the same design principle of shielding the transport layer from the packet losses that occur at the lower layers (network and MAC layers). Techniques such as forward error correction (FEC) and automatic repeat request (ARQ) are used to hide wireless link errors from TCP. The intuition behind this approach is that since the problem is local to the wireless link, it should be resolved locally. Providing loss recovery at the link layer seems like a natural way to address the high BER of wireless networks. However, link-layer solutions are known to cause adverse interactions with transport-layer protocols.

Among the proposed link-layer protocols, Snoop by Balakrishnan et al. [1995], Radio Link Protocol (RLP) by Nanda et al. [1994], AIRMAIL by Ayanoglu et al. [1995], and TULIP by Parsa and Garcia-Luna-Aceves [1999] are the most well known ones. Section 2.4.1 – 2.4.4 will provide an overview for each of these four solutions respectively.

2.4.1 Snoop

The snoop protocol attempts to resolve the high BER problem of wireless networks by caching unacknowledged packets at the base station and performing local retransmission over the wireless link for lost packets. More specifically, the snoop protocol introduces a snoop agent that resides at the base station and monitors every packet within the TCP connections that pass through the base station. For each connection, the snoop agent maintains state information on the yet-to-be acknowledged packets. Upon detections of packet loss, the snoop agent retransmits the locally cached copy and suppresses any duplicated acknowledgments.

Since the snoop protocol relies on the caching of unacknowledged packets at the base station, such cache content must be transferred to or built up at the new base station during a handoff process. Consequently, the snoop protocol could suffer from performance degradation if the mobile host moves completely out of the coverage of its old base station before its new base station has enough time to obtain a working set of the cache. Another drawback that the snoop protocol suffers from is that for encrypted packet transmissions, snooping inside the TCP connection is not feasible, thereby rendering the snoop protocol useless in such scenarios.

2.4.2 Radio Link Protocol

The radio link protocol proposed by Nanda et al. [1994] employed a point-to-point automatic repeat request mechanism over radio channels in order to bring the perceived packet loss rate at the transport layer down to 0.1%. RLP maximizes bandwidth utilization by preemptively transmitting unacknowledged packets over the channel during idle periods. Most versions of RLP is negative acknowledgment (NAK) based, meaning that retransmissions are usually triggered when the receiver notifies the sender of packet loss event using NAK packets. Cellular networks such as GSM and CDMA networks employ RLP as their link layer protocol.

2.4.3 AIRMAIL

AIRMAIL, also known as asymmetric reliable mobile access in link-layer, was proposed by Ayanoglu et al. [1995]. AIRMAIL is asymmetric in the sense that all decision makings regarding packet transmission are made by the base station. This design choice was made under the consideration that the bulk of the processing work should take place at the base station in order to maximize the battery life of the mobile host. The base station periodically sends status messages to the mobile host. In contrast, acknowledgment from the mobile host is sent to the base station in an event driven manner in order to conserve battery power. The potential drawback of this scheme is that the additional delay caused by the event driven acknowledgments may trigger unnecessary invocations of TCP's congestion control mechanism.

AIRMAIL uses a combination of ARQ and FEC to ensure the reliability of the wireless link. FEC can be implemented at three different levels: bit-level (physical layer), byte-level (per packet cyclic redundancy code), and packet-level (extra packets carrying redundancy). The designers of AIRMAIL showed that different level of FEC is needed depending on the characteristics of the mobile network. Consequently, AIRMAIL adaptively utilizes all three levels of FEC to achieve reliable wireless link with minimum coding overhead.

2.4.4 Transport Unaware Link Improvement Protocol (TULIP)

Another notable link-layer solution called TULIP was proposed by Parsa and Garcia-Luna-Aceves [1999] to improve the performance of TCP over lossy wireless links without modifying the transport or network-layer protocols. TULIP is transport unaware in the sense that its operation does not depend on any knowledge about the transport layer protocol session status. However, TULIP is designed to be service aware in the sense that it provides reliable link-layer services for TCP-based applications and non-reliable link-layer services for UDP-based applications. This QoS-based service segmentation distinguishes TULIP favorably from the other available link-layer solutions.

TULIP provides a MAC acceleration mechanism for the collision-avoidance MAC protocols to improve overall throughput. TULIP achieves link-layer reliability through local retransmission of erroneous packets and maintains the flow control via a sliding window mechanism. To avoid TCP's spurious triple dup-acks and congestion control triggers, TULIP ensures in-order delivery of link-layer data packets to the upper layer. TULIP's timers rely on the measured maximum propagation delay over the network, rather than the conventional round-trip time estimate of the channel delay. Furthermore, TULIP is very simple to implement since its operation does not require the implementation of a base station agent or any safe keeping of TCP state as in the case of TCP SNOOP. A direct consequence of this simplicity is that TULIP can work with any transport or network protocol even if there is upper-layer encryption involved as in the case of IPSec. The main drawback of TULIP is that it suffers from spurious retransmit timeouts due to the incompatibility that exists among the cross-layer timers.

2.5 Split-connection Solutions

An alternative way to improve TCP's performance over wireless-last-hop networks is the split-connection approach. For this type of solutions, the TCP connection between the server and the mobile client is split up into two separate connections at the base station – one runs on the fixed part of the network (between the server and the base station) and the other one runs on the mobile part (between the base station and the mobile client). Figure 2.2 shown below demonstrates this process.

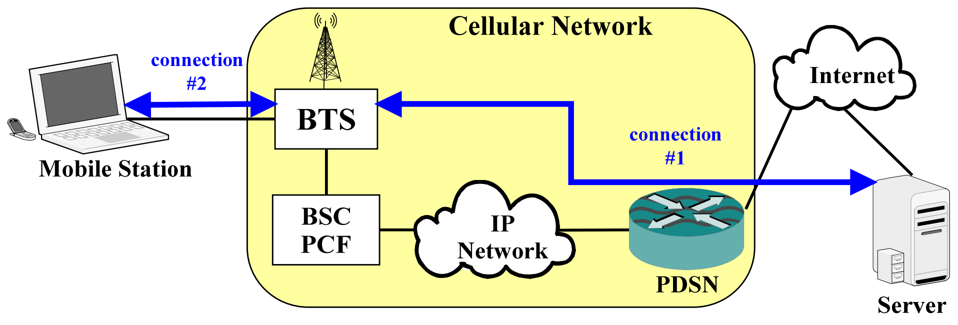


Figure 2.2: Split-connection Protocol

The primary goal of partitioning the network is to shield the TCP sender (the server) from the lossy wireless network, and thereby avoiding unnecessary triggering of TCP's congestion control mechanism. The major drawback of this approach is that it violates the end-to-end semantics of TCP since packet acknowledgments can now arrive at the packet source before the actual packet arrives at the mobile host. Another disadvantage is that a large amount of state information must be maintained at the base station, leading to rather costly handoff operations.

In the past, a number of split-connection protocols have been proposed by researchers. The well known ones include: MTCP by Yavatkar et al. [1995], I-TCP by Bakre et al. [1997], M-TCP by Brown et al. [1997], and METP by Wang and Tripathi [1998]. Sections 2.5.1-2.5.4 will provide an overview for each of these four solutions respectively.

2.5.1 MTCP

In MTCP, Yavatkar et al. [1995] introduced a session layer protocol called the Mobile Host Protocol (MHP), which resides on both the base station and the mobile host. The principle functionality of the MHP is to segregate the part of the connection that runs over the wired network from the part that runs over the wireless link, thereby shielding the fixed host from the more lossy and unpredictable mobile environment. In addition, the MHP adaptively compensates for the wireless link unreliability based on the characteristics of the mobile environment as well as the migration pattern of the mobile host. The MHP uses a selective repeat technique over the wireless link to quickly recover from bursty packet losses.

2.5.2 I-TCP

I-TCP is very similar to MTCP in term of design. When a mobile host attempts to communicate with a fixed host, the base station to which the mobile host is connected to creates two separate connections. One of the connections runs between the base station and the fixed host and operates on top of regular TCP. The other connection, operating between the base station and the mobile host, uses a version of TCP that is adapted to the characteristics of the mobile environment.

I-TCP differs from MTCP in the sense that these two protocols concentrate on different aspects of the same problem. While MTCP addresses the issue involving lossy wireless links in the absence of motion, I-TCP concentrates on the effects that mobility and handoff have on TCP performance. When a movement of the mobile host or a temporary disconnection due to handoff is detected, I-TCP resets the retransmission timers belonging to the wireless part of the connection and immediately enters the slow-start phase. Consequently, I-TCP is able to resume normal mode of operation quicker than traditional TCP, resulting in smaller performance hit in the presence of temporary disconnections.

2.5.3 M-TCP

In designing M-TCP, Brown and Singh [1997] addressed several key issues regarding wireless TCP traffic: fluctuating bandwidth, frequent disconnections, and limited power supply on the mobile devices. The result of this work is a protocol that performs local retransmission to compensate the lossy wireless link, dynamically allocates bandwidth to mobile hosts based on their changing needs, and minimizes duplicate packets to reduce energy consumption.

The architecture of M-TCP adopts a three-layer hierarchical approach: the base layer includes the mobile hosts and the base station belonging to the same cell; the middle layer consists of a supervisor host, which controls several base stations from different cells; and at the top layer, the supervisor hosts are connected to the wired network. Similar to MTCP and I-TCP, M-TCP is a split-connection protocol where the connection is split at the supervisor host. An adapted version of TCP is used over the wireless link whereas the regular version of TCP is used on the wired portion of the network. The supervisor host is responsible for relaying packets transmitted between the fixed host and the mobile host. In the case when the mobile host is temporarily disconnected from the base station, the supervisor host forces the TCP sender on the fixed host to go into persist mode by setting the window size to zero. This prevents the TCP sender from downsizing its congestion window or entering retransmission timeout loops. Correspondingly, the mobile host freezes all of its internal states upon the temporary disconnection and later resumes its normal operation when it is reconnected to the base station.

The main advantage of M-TCP's layered structure is that no state-transfer or other handoff processes need to take place when the mobile host moves from one cell to a neighboring cell so long as the base stations of the two cells are associated with the same supervisor host node. Consequently, M-TCP enjoys much more efficient handoffs than both I-TCP and MTCP.

2.5.4 METP

Even after the splitting of the TCP connection at the base station, it can still be observed that TCP is poorly suited for the wireless link between the base station and the mobile host. Based on this fact, Wang and Tripathi [1998] have created an enhanced version of the split-connection protocol called Mobile End Transport Protocol (METP). METP eliminates the TCP and IP layers over the wireless link and directly operate over the link layer. Consequently, the base station acts as a translation proxy between the fixed and the mobile part of the connection by providing a conversion of the packets received from the fixed network into METP packets.

The authors have reported that TCP METP enjoys a throughput enhancement of 37 percent over TCP Reno. However, this performance gain does not come for free. METP suffers from two notable drawbacks. Firstly, it violates the end-to-end semantics of TCP as in the cases of other split-connection solutions. Secondly, this scheme greatly increases the complexity of the base station due to the added overhead of packet conversion and processing. During handoffs, large amount of state information must be transferred from the old base station to the new one in order to maintain the integrity of the connection.

2.6 End-to-End Protocols

For this category of solutions, the end-to-end semantics of the original TCP design is maintained. This means that the TCP sender and receiver are ultimately responsible for the reliability of packet transmission. Various reliable transport mechanisms have been devised in previous research to improve the throughput of TCP connections without sacrificing other quality measures. This section examines the most representative solutions in the end-to-end paradigm including Reno, New-Reno, selective acknowledgment (SACK), Vegas, Westwood, Peach, Veno, Freeze, Santa Cruz and WTCP. Furthermore, a multi-homed TCP solution, called pTCP, is also examined in this section to offer the reader a different perspective on TCP performance enhancing techniques. Sections 2.6.1-2.6.11 will provide an overview for each of these eleven solutions respectively.

2.6.1 TCP Reno

Although time-outs and three duplicate ACK's (triple-dup-ack's) are both considered signs of congestion, triple-dup-ack's correspond to the less severe form of congestion. TCP Reno exploits this fact by introducing the concept of fast recovery (as depicted in Figure 2.3), which optimizes TCP Tahoe's congestion control mechanism through differential treatments of timeouts and triple-dup-ack's.

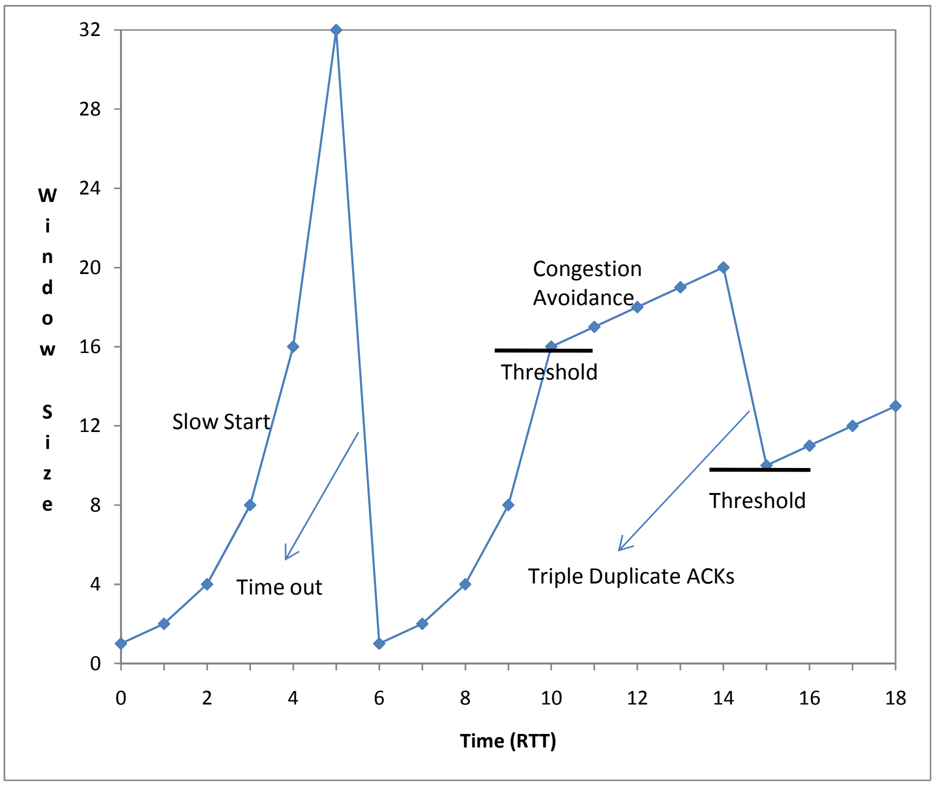


Figure 2.3: Fast Recovery Mechanism

As it is illustrated in Figure 2.3, the sender in TCP Reno initiates the fast recovery phase whenever it receives a triple-dup-ack's. In the fast recovery phase, the TCP Reno sender retransmits the lost packet to the receiver while simultaneously reduces the value of the slow-start threshold (*ssthresh*) by half. The key difference between TCP Tahoe and TCP Reno is that unlike in TCP Tahoe, the Reno sender does not enter the slow-start phase after receiving a triple-dup-ack's. Instead, it first halves the congestion window (*cwnd*) and subsequently increments *cwnd* by one upon each arrival of a duplicate ACK. The arrival of the first non-duplicate ACK signals the sender that the network is no longer in a congested state. The Reno sender responds by setting *cwnd* to *ssthresh* and then enters the congestion avoidance phase.

2.6.2 TCP New-Reno

TCP New-Reno, originally proposed by Hoe [1996], offers the capability of recovering from multiple packet losses within the same sender window, a feature not found in either TCP Reno or TCP Tahoe. Multiple packet losses within a single window occur quite often during network congestions owing to the fact that most Internet routers use droptail queues, which drop all late-arriving packets when buffer overflows occur. TCP Reno fails to properly handle multi-packet losses since it exits the fast-recovery phase as soon as its sender receives the first non-duplicate ACK. As a result, TCP Reno recovers from a multi-packet loss event by waiting through a series of retransmission timeouts. TCP New-Reno, on the other hand, exits the fast recovery phase only when all data transmitted prior to the start of the fast recovery phase have been acknowledged. TCP New-Reno achieves this by taking note of the highest sequence numbered packet (denoted as HSeqnoPak) it has sent prior to entering the fast recovery phase. When a non-duplicate ACK, with lower sequence number than HSeqnoPack, is received by the New-Reno sender during the fast recovery phase, the sender treats it as a partial ACK and assumes that the packet (denoted as packet X) with sequence number immediately higher than the one in the partial ACK is lost as well. Consequently, the sender then re-transmits packet X. This process continues until the sender receives the non-duplicate ACK with HSeqnoPack's sequence number. In essence, New-Reno improves TCP's throughput in the presence of multi-packet losses by decreasing the number of timeouts.

2.6.3 TCP SACK

An alternative approach to recover from multi-packet losses is to use the selective acknowledgment (SACK) scheme on top of traditional TCP. SACK was defined by Mathis et al. [1996] and Floyd et al. [2000], in RFC 2018 and 2883, respectively. In traditional TCP, the receiver is only capable of informing the sender of the last in-order packet that it has properly received, leading to bulk retransmission upon packet losses. But with TCP SACK, the receiver can inform the sender of the actual lost packets using selective ACK, thereby minimizing the bandwidth wastage that would have otherwise occurred due to the retransmission of already received packets. In addition, upon receiving a selective ACK, the TCP SACK sender can retransmit the lost packets as reported by the selective ACK in the next RTT, leading to a significant throughput improvement. Compared with TCP Reno and New-Reno, TCP SACK suffers from the drawback that modifications are required not only on the sender side, but also on the receiver side as well. However, for long fat pipe networks, the SACK scheme is the key for handling packet losses without incurring the expensive data pipeline drainage. This point is further emphasized and explored in section 3.5.

2.6.4 TCP Vegas

As it was originally proposed in Brakmo et al. [1995], TCP Vegas takes a bandwidth estimation approach to congestion control. By accurately estimating the available channel bandwidth, TCP Vegas can avoid potential congestions and packet drops. In essence, TCP Vegas pays more attention to packet delays rather than packet losses in the process of determining the proper send rate. TCP Vegas closely monitors the variation in packet delays by keeping track of the RTT values. At the beginning of a connection, TCP Vegas records down the first RTT measurement as the base RTT value, representing the minimum RTT under a non-congested network state. At any given time, the expected flow rate and the actual flow rate can be calculated using the expressions: cwnd/baseRTT and cwnd/actualRTT, respectively. By comparing these two values, TCP Vegas can get a good estimation of the congestion level of the network and thereby fine-tuning the flow rate by either linearly increasing or decreasing the congestion window (cwnd). This type of congestion avoidance strategy enables TCP Vegas to keep the flow rate around the maximum non-congestive level.

The downside of the congestion avoidance scheme is that it makes TCP Vegas non-competitive in term of bandwidth acquisition as compared with TCP Reno. Consequently, it would be difficult to widely deploy TCP Vegas on today's networks where TCP Reno already enjoys a predominate presence.

2.6.5 TCP Westwood

TCP Westwood is a sender-side-only modification to TCP New-Reno, originally proposed by Casetti et al. [2001] to address TCP's performance issues over network platforms with high packet loss rate as well as high bandwidth delay product. TCP Westwood takes a rate-based approach to flow/congestion control instead of the traditional sliding-window approach taken by the other TCP variants. The TCP Westwood estimates the available bandwidth base on the interval times between successively received ACK's.

In order to properly deal with heavy loss environment such as wireless-last-hop networks, TCP Westwood utilizes a loss discrimination algorithm (LDA) to distinguish between losses due to congestions and losses due to wireless medium failures. The LDA uses a combination of cumulative RTT estimation and the deviance between observed and expected flow rate to judge whether a certain packet loss event is due to congestion or medium failure. When a non-congestive packet loss event occurs, the TCP Westwood sender uses a bulk retransmit mechanism which retransmits all unacknowledged packets instead of one at a time to enable prompt recovery from multiple bursty packet losses.

2.6.6 TCP Peach

To address TCP's performance problems in satellite networks, Akyildiz et al. [2002] proposed an end-to-end solution called TCP Peach. In their paper, the authors introduced two new algorithms for enhancing TCP's congestion control mechanism: sudden start and rapid recovery. The core idea is to utilize low-priority packets called dummy segments to probe the availability of the sender. Because these dummy packets are marked with low-priority, they are the first one to be dropped by intermediate routers when congestions occur. Consequently, the dummy packets do not negatively impact the congestion level of the network.

During the sudden start phase of a new TCP Peach connection, dummy packets are used to probe the available network resource and adaptively improve the slow-start growth of bandwidth acquisition.

When congestions occur, the Peach sender enters into the rapid-recovery phase and uses the dummy packets to detect the nature of the packet losses. Congestive losses are handled differently from wireless link losses. Finally, during the congestion avoidance phase, dummy packets are used to monitor the congestion level of the network and thereby preventing the occurrences of congestion.

2.6.7 TCP Veno

According to Fu and Liew [2003], TCP Veno is a TCP-friendly, sender-side-only modification to TCP Reno and Vegas. It attempts to improve the performance of traditional TCP over lossy wireless network by distinguishing between congestive and random wireless packet losses. Based on the severity of congestion, TCP Veno dynamically adjusts the slow-start threshold to achieve the best performance. Veno estimates the degree of congestion of a network by estimating the amount of backlogged data on the network. In the event of packet losses, Veno determines the nature of the losses by using a threshold value β. If the number of backlogged packets on the network is less than β, then the losses are considered random. Veno responds to random packet losses by decreasing the congestion window by 1/5. However, when the number of backlogged packets exceeds β, Veno deems the packet losses to be due to congestion and reacts in the same manner as TCP Reno would. The effectiveness of this scheme relies critically on the selection of the β value, which is experimentally determined to be 3.

2.6.8 Freeze

In order to overcome TCP's performance problems in the presence of cellular signal fading and handoffs, Goff et al. [2000] proposed a modified version of TCP called Freeze TCP. The intuition of this solution is as follows: since it is often observed that temporary disconnections occur during deep signal fading and lengthy handoffs, it is more efficient to stop the transport layer transmission all together upon the detection of these events and resume only when the transient network disturbances disappear. In Freeze TCP, the TCP receiver is responsible for predicting the occurrences of temporary wireless link disconnection based on the observed wireless conditions. When a disconnection is predicted, the receiver notifies the sender by setting the advertised window size to zero and transmit this value to the sender via an acknowledgment packet. The sender then goes into a persist mode in which it suspends all packet transmissions to the receiver but leaves the congestion window size and the retransmit timer value unchanged. Upon reconnection, the receiver transmits a packet to the sender to acknowledge the last received packet before disconnection. This process alleviates the slow-start inefficiency and enables the resumption of packet transmission at the pre-freeze rate.

Freeze TCP does not require any software changes to be made to the sender side or any intermediate routers. The necessary changes are restricted to mobile client side, making Freeze TCP fully compliant with the existing TCP infrastructure. However, it suffers from a minor drawback associated with its high timer sensitivities: The performance of Freeze TCP is critically affected by the timing of sending out the zero window-size acknowledgment. Sending out the acknowledgment too early would cause earlier than necessary termination of packet transmission. On the other hand, sending out the acknowledgment too late could cause the unnecessary triggering of TCP's congestion-control mechanism.

2.6.9 Santa Cruz

Parsa and Garcia-Luna-Aceves [1999] created TCP Santa Cruz to accommodate the following five problems seen on today's Internet: path asymmetries, out-of-order packet delivery, lossy links, dynamic fluctuations in path delay, and limited bandwidth. The protocol improves over TCP Reno in two major areas: congestion control and error recovery. These two mechanisms use estimates of delay along the forward path rather than the RTT in converging the packet transmission rate to the target operating point without congesting the network. TCP Santa Cruz's error recovery method allows the sender to efficiently retransmit any lost packet in a timely fashion, thereby eliminating any unnecessary retransmission for properly received packets when multiple losses occur within a single transmission window. The congestion control algorithm is similar to the one used by TCP Vegas although the former relies on the forward path delay whereas the later relies on the RTT. TCP Santa Cruz utilizes the measured forward path delay to detect incipient stages of congestion in the forward path and thereby allowing the sender to react to early congestive warning signs through a proactive adjustment scheme of the congestion window.

Through simulation, the authors showed that TCP Santa Cruz on average achieves higher throughput than TCP Reno and is capable of preventing congestion window size swings that typically occur in TCP Reno and TCP Tahoe.

2.6.10 WTCP

Wireless TCP (WTCP) proposed by Sinha et al. [2002] is designed to address the most common causes of throughput degradations observed in wireless wide area networks: limited and fluctuating bandwidth, highly variable RTT, and bursty packet errors. The main design criteria of WTCP are robustness, fairness, efficiency, reliability and deployability. WTCP uses two key schemes to realize its design goals. First, a rate-based rather than a window-based flow control mechanism is employed to achieve adaptive transmission rate adjustment. The WTCP receiver uses the ratio of interpacket durations recorded at both the sender and the receiver to estimate the optimal transmission rate. The sender embeds the interpacket duration value in every data packet that it sends to the receiver and receives an update of the current transmission rate estimate from the receiver upon the arrival of each ACK packet. This mechanism makes WTCP insensitive to non-congestive packet losses, large RTT fluctuations, and bursty ACK’s. Consequently, WTCP is able to ensure fairness among competing network flows having different RTTs.

The second main contribution of WTCP is its rate-based rapid recovery scheme, designed to overcome the slow-start inefficiencies suffered by traditional TCP. WTCP accomplishes this by sending out two back-to-back probe packets and use the interpacket delay to estimate the initial transmission rate upon rapid recoveries. The creators of WTCP have reported an improvement of 20%-200% over other TCP algorithms such as New Reno, Vegas, and Snoop in simulations.

2.6.11 pTCP

Hsieh and Sivakumar [2002] presented an alternative approach from the conventional thinking on improving TCP’s performance over wireless networks. Instead of directly addressing the shortcomings of TCP, the authors take the view point of achieving greater aggregate bandwidths through multi-homing techniques. In their paper, the authors argued that in the presence of the wide variety of available wireless access technologies, a mobile host could potentially gain access to multiple wireless networks at a given time and thereby enjoying a higher overall bandwidth. Prior to pTCP, other researchers have investigated both the link-layer and application-layer approaches to multi-homing. Hsieh and Sivakumar believe that neither of these two approaches performs satisfactorily: link-layer striping schemes fail to work if the multiple interfaces belong to different wireless network domains; application-layer techniques do not scale well in the presence of varying and fluctuating link characteristics. Following this path of reasoning, the authors developed a transport layer striping scheme to aggregate the bandwidth from multiple wireless network interface using a combination of mechanisms including: decoupled congestion control and reliability, congestion window based striping, dynamic windows reassignment, redundant striping to handle blackouts, and support for different congestion control schemes to co-exist within a single transport-layer framework. Through simulation, pTCP is shown to be efficient at aggregating bandwidths from multiple wireless interfaces under a variety of network conditions.

2.7 Comparison of Protocol Categories

This section provides a comparative analysis of the three categories of protocols discussed in the previous sections. The major disadvantage of split-connection protocols is that they break the end-to-end semantics that the regular TCP follows. As it was mentioned before, an important consequence of not maintaining the end-to-end semantics is that packets may not have reached mobile host before the corresponding ACK's get delivered to the fixed host. This poses a serious problem of state inconsistencies if the mobile host gets frequently disconnected from the base station. Consequently, split-connection protocols are not well-suited for applications that rely on the end-to-end semantics of TCP. The major advantage of split-connection protocols is that they provide backward compatibility with the existing wired network protocols, thus do not require any modifications at the fixed host for accommodating mobile hosts. They shield the mobility and wireless problems from the fixed host.

In comparison, link layer protocols maintain the end-to-end semantics of TCP while significantly improves the reliability of wireless links through local retransmissions. Lower BER over the wireless links can translate into improved TCP performance. However, Balakrishnan et al. [1996] has demonstrated that link-layer protocols may negatively affect the performance of TCP due to cross-layer timer conflicts that leads to spurious retransmissions. Furthermore, the FEC scheme introduces a significant amount of redundancy which inevitably results in inefficient bandwidth utilization.

End-to end protocols closely adhere to the design principles of TCP by addressing the network reliability issue at the transport layer. However, this category of protocols generally suffers from the drawback that sender-side, and sometimes receiver-side modifications need to be made to the original TCP software. It would be desirable to have an alternative end-to-end protocol that sits at the user-level, thereby eliminating the need for recompilation and re-linking of the OS kernel-level TCP software.

2.8 Desired Adaptations over Non-conventional Networks

A number of desirable traits that TCP should possess in order to better adapt to the various non-conventional networks can be concluded from the previous discussions in this chapter. Although some of these adaptations may not be realizable at the transport layer, they are nevertheless included in the following list for the sake of completeness.

1. Distinguish between congestive and non-congestive packet losses, and thereby preventing unnecessary triggering of the congestion control mechanism.
2. Avoid the serial timeout problem by properly handling multi-packet losses.
3. Resolve the data pipeline drainage problem on long fat pipes by using selective acknowledgment.
4. Improve the reliability of lossy wireless link by reducing the perceived BER and properly handling handoffs.
5. Obey the end-to-end semantics of TCP.
6. Achieve maximum performance gain with minimal footprint (i.e. avoid unnecessary changes to existing OS kernel-level software).
7. Realize greater aggregate bandwidth through all available network interfaces, a.k.a. multi-homing.

Out of the adaptations listed above, RRTP attempts to address all the transport layer issues except for the multi-homing approach. The decision to exclude multi-homing from this piece of research is made mainly due to time and scope constraints. An important quality that RRTP is designed to exhibit is the ability to serve as a viable replacement for TCP over all types of networks predominated by TCP today. In other words, RRTP not only aims to outperform TCP on non-conventional networks for which TCP is ill-suited for, it is also adept for providing competitive performance over conventional networks. Chapter 3 will present the detailed design of RRTP that enables it to serve as a versatile replacement for TCP.

Chapter 3
Design Description

In this chapter, the key aspects of RRTP, including its user-level design, connection setup procedure, reconfigurable options, flow/congestion control mechanism, reliability control mechanism, and loss differentiation algorithms are examined in detail.

3.1 User Level Design

While all of the previously discussed TCP variants provide performance improvement over TCP, most of them require kernel-level software changes to be made to the existing OS protocol stacks. In contrast, RRTP offers a user-level solution to address the various inefficiencies that TCP exhibits over non-conventional networks, leaving zero footprints on the existing software infrastructure. From an implementation point of view, software programs situated at or below the transport layer are considered to be at the kernel-level whereas those ones situated above the transport layer are considered to be at the user-level. Consequently, RRTP must be constructed on top of the transport layer in order to be considered a user-level protocol.

Among all of the existing transport layer protocol, UDP is the best candidate to serve as a base protocol for RRTP. This is due to the fact that UDP is a bare-bone transport layer protocol, lacking the support for connection-orientation, data reliability control, as well as flow and congestion control mechanisms. Consequently, having UDP as its base protocol, RRTP enjoys complete design freedom in complementing the advanced features that UDP lacks. Furthermore, in constructing RRTP on top of UDP, efficiency is achieved by ensuring that there are no feature overlaps between the two protocols.

Since RRTP relies on UDP for the basic transport layer services, RRTP packets are essentially UDP packets with control header add-ons as shown below in Figure 3.1. Additional control packet headers are illustrated in Appendix A.

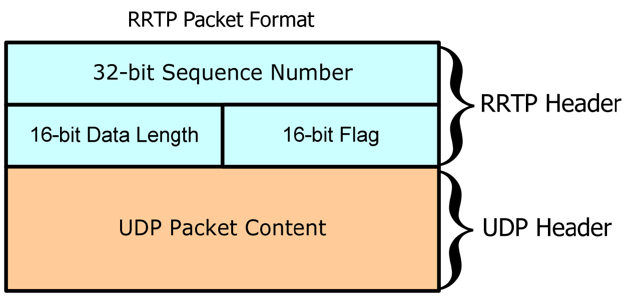


Figure 3.1: Generic RRTP Packet Format

Being a user-level protocol, RRTP offers performance improvement over TCP without incurring any effort in changing existing software stack to accommodate RRTP. This represents a significant improvement over and paradigm shift from the previous work done on TCP performance tuning since most of the past solutions require substantial changes to the kernel software stack. For an open-source operating system (OS), kernel-level changes entail partial recompilation of the OS. For a commercial OS, the vendor must adopt the kernel-level solution before the end-user can enjoy its superior performance. RRTP eliminates these inconveniences altogether as it works well in a plug-and-play manner over any platform that supports UDP.

3.2 Connection Setup

UDP is a connectionless protocol that lacks a set of connection establishment procedures. RRTP adds the concept of connection-orientation on top of UDP through a 4-way handshake process that is similar to the one described by Ong and Yoakum [2002]. The 4-way handshake is designed to eliminate the security risk posed by the denial of service (DoS) attack, which is often launched against servers operating over traditional TCP. In a denial of service attack, the hacker floods the target server with a large number of SYN requests without responding to any of the SYN ACK's sent by the server. This leads to a significant resource leak at the target server in the form of ghost transmission control blocks (TCB), and thereby effectively starves the legitimate connection requests.

The 4-way handshake solves the DoS problem by employing a server-generated cookie for connection request authentication. Upon receiving the connection request (SYN packet) from the client, instead of immediately allocating a TCB for the connection request, the server sends a SYN-ECHO packet with an auto-generated cookie and waits for the client's authentication. The client sends the original cookie back to the server in a START packet to authenticate itself. Upon receiving the START packet and successful verification of the cookie, the server allocates a TCB for the connection and sends a START-ECHO packet to the client to formally initiate the connection. The entire process is illustrated in Figure 3.2 shown below.

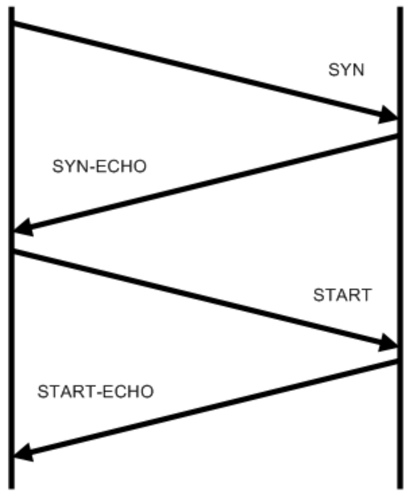


Figure 3.2: RRTP's Four-way Connection Establishment Process

During the handshaking process, the initial value of the connection's RTT is determined. This value will serve as the initial RTT estimate unless a user pre-configured seed value is provided by the application that runs on top of RRTP.

Once the network connection is fully established, the RRTP sender will send out two successive packets, denoted as P1 and P2, for the purpose of probing the network capacity and determining the initial send rate. Here, it is assumed that the two machines involved in the connection are free of CPU intensive tasks so that the RRTP process on each machine can get the necessary CPU cycles to complete the probe. Several other variants of TCP also depend on similar assumptions. Assume that the RRTP sender records the inter-packet duration between P1 and P2 to be X milliseconds. Once the receiver gets both packets, it will advertise to the sender the observed receive interval (Y milliseconds) for packets P1 and P2. The sender then calculates the initial send rate based on max(X, Y). Assuming that the network is not in a congested state for the duration of the probe, the resulting send rate value should be a good estimate of the characteristic value of the network.

Just as in the case of determining the initial RTT value, the application programmer has the option of seeding RRTP with an initial send rate value to achieve faster convergence and better bandwidth utilization. The positive effect of send rate pre-configuration is most pronounced on wireless-last-hop networks for reasons that will be discussed in section 3.4.2. In general, proper utilization of the application programmer's knowledge of the network characteristics can translate into improvement in both connection throughput and stability. RRTP's ability of achieving better performance through parameter-based user configuration is further explored in section 3.3.

3.3 Reconfigurability

As it was mentioned in the previous chapter, TCP was originally designed for wired networks and consequently adapts poorly to various non-conventional networks. In fact, it is intuitive that a one-size-fit-all protocol like TCP is unlikely to perform well over all types of networks. One way to rectify this design level deficiency is to add configurable components to the key algorithms of a one-size-fit-all protocol, thereby enhancing its adaptability to the different types of networks that exist today. Although the various types of networks could differ quite drastically in terms of their underlying hardware and software technologies, a set of key parameters, including network throughput, end-to-end delay and packet loss rate, is usually sufficient to uniquely identify each type of network platforms.

The ability to reconfigure in order to adapt to different network platforms is the key feature that sets RRTP apart from other TCP variants. As an application programmer who designs software that communicates via the transport layer protocols, he or she is likely to be intimately familiar with the characteristics of the network platform on which the software is to be deployed. For instance, the designer of a download manager application for a 3G cellular network usually knows the ballpark values of the important parameters such as network throughput and delay under normal operating conditions. Such information can be utilized to enhance the efficiency of flow control and congestion control processes and thereby improving the connection throughput.

RRTP offers two levels of configuration. At the higher level, appropriate algorithms are chosen based on the targeted network: a more aggressive version of the congestion control mechanism is used for the wireless-last-hop networks to deal with the high BER issue. Additional details regarding the higher level configuration will be presented in section 3.4. The lower level configuration takes the form of pre-seeding the key algorithmic parameters with characteristic values to better adapt to the targeted network platform. Table 3.1 shown below summarizes the set of configurable parameters that RRTP exposes to its users.

Table 3.1: User-configurable Network Parameters

Parameter	Meaning
$\text{SendRate}_{\text{char}}$	Characteristic send rate
$\text{RoundTripTime}_{\text{char}}$	Characteristic end-to-end network latency
$\text{LossRate}_{\text{char}}$	Characteristic data loss rate

Here, the phrase "characteristic value" should be interpreted as the steady-state value for a specific network platform. Steady-state refers to the condition in which the dynamics of a network is not rapidly fluctuating and additional bandwidth can be acquired without causing network congestions. For network platforms such as the wireless-last-hop networks, the characteristic send rate is likely to be quite stable as the bandwidth is effectively allocated on a per-user basis. However, for more complex or heterogeneous network platforms, steady-state conditions may be disrupted by unforeseeable events that last for lengthy durations. As a result, the meaning of "characteristic value" is further restricted to the most commonly observed value for a given network.

Intuitively, the effectiveness of parameter-based configuration is directly correlated to the stability of the network dynamics. This in turn means that the positive impact of parameter-based configuration is likely to be most pronounced for short-lived connections, whose window of opportunity for the occurrences of significant network perturbations is relatively limited.

Assuming that the application programmer has an accurate knowledge of the characteristic throughput of a network, he or she can preset the $\text{SendRate}_{\text{char}}$ parameter prior to the start of a connection, enabling the bandwidth utilization of RRTP to quickly reach the near-optimal level. For relatively short-lived connections, TCP, due to its slow-start mechanism, spends a significant amount of time in converging to the optimal level of throughput of the network. Assuming that the time spent in the slow-start phase is around 50% of the total connection time, the overall throughput of the TCP connection is expected to be quite low as the TCP sender only operated at close to full capacity for about half of the connection duration. RRTP, on the other hand, is capable to operate at full capacity for nearly the entire duration of the connection. Consequently, for applications that communicate via frequent, short-lived connections, RRTP is the protocol of choice.

Similar to the SendRate$_{char}$ parameter, the RoundTripTime$_{char}$ and the LossRate$_{char}$ parameters can be pre-configured by the application programmer to optimize RRTP`s flow-control and congestion-control algorithms. All three parameters are continually refined over the course of a RRTP connection. Even in the case where the original estimates for the parameters are not accurate, the algorithms within RRTP are robust enough to quickly converge to the actual network values.

3.4 Flow Control and Congestion Control Mechanisms

According to Chiu and Jain [1989], the linear increase multiplicative decrease (LIMD) approach to congestion control is the only paradigm that will settle down to a state of fairness with an arbitrary starting send rate. However, achieving fairness is not always a necessary requirement for congestion control algorithms. For networks on which there is effectively only one traffic flow between the sender and the receiver, the concept of fairness is no longer relevant. As a result, depending on the underlying network platform, RRTP chooses the appropriate approach to flow control and congestion control. For wireless-last-hop networks, the fairness restriction can be completely relaxed since there is only one effective traffic flow between the mobile host and the fixed host, as demonstrated in Figure 3.3 shown below:

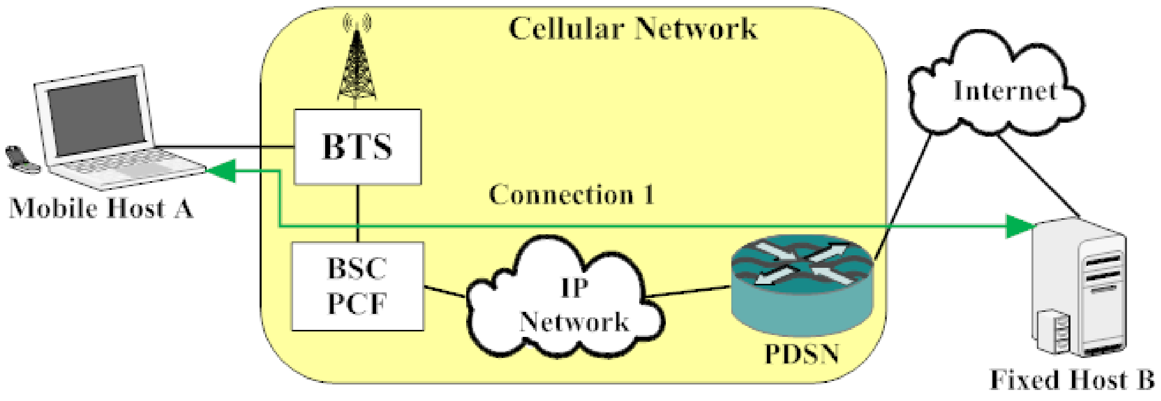


Figure 3.3: Wireless-last-hop Network

The wireless-last-hop network is assumed to be a single-flow environment (here after known as the dedicated channel assumption) for two reasons. The first reason is that for the connection (denoted as Connection 1 in Figure 3.3) running between Mobile Host A and Fixed Host B, the wireless link is the bottleneck of the connection due to the significant difference in capacity between the wired and wireless links. In other words, fluctuations of the available bandwidth on the wired part of the connection have no effect on the overall throughput of Connection 1. Consequently, Connection 1 can be viewed as an independent traffic flow between Mobile Host A and Fixed Host B, not affected by any other flows on the wired part of the network. The second reason is that the entire bandwidth of the wireless channel established between Mobile Host A and the BTS is dedicated for the usage of Mobile Host A. In other words, there are no other traffic flows competing for the network bandwidth allocated to Mobile Host A in the same wireless channel. As a result, congestion can be avoided as long as the sender does not exceed the bandwidth limit of the wireless channel. Consequently, RRTP can take a more aggressive approach to flow control and congestion control than the traditional LIMD paradigm that TCP follows.

For networks other than the wireless-last-hop platform, RRTP follows the basic framework of the LIMD paradigm to fulfill the fairness requirement in the presence of multiple traffic flows through the same pipe. However, instead of taking TCP's approach of congestion window based bandwidth acquisition and adjustment, RRTP uses a rate-based algorithm that reacts to incipient congestion and attempts to limit the rate of traffic flow below the estimated maximum available bandwidth. This is a better approach than TCP's native flow control and congestion control mechanisms because RRTP can avoid serious network congestions by reacting to early signs of path saturation. The details of RRTP's flow control and congestion control mechanisms are presented in section 3.4.1 and 3.4.2.

3.4.1 Flow Control and Congestion Control for Non-wireless-last-hop Networks

For network platforms with multiple traffic flows, RRTP's approach to flow control and congestion control builds on TCP Reno's solution. The principal problem with TCP Reno's approach is that the congestion avoidance phase often ends when the send rate exceeds or overshoots the available bandwidth, leading to congestive packet losses. Such overshoots significantly decreases the overall throughput of the connection since congestion control kicks in whenever signs of path saturation are detected by the sender. RRTP rectifies this shortcoming by reacting to early signs of network congestion and thereby avoiding the triggering of the congestion control mechanism. In addition, RRTP imposes upper and lower bounds on the send rate, denoted as $\text{SendRate}_{\max}$ and $\text{SendRate}_{\min}$ respectively, to avoid unnecessary overshoots during the congestion avoidance phase. $\text{SendRate}_{\max}$ serves the purpose of preventing the newly computed send rate from exceeding the maximum available bandwidth. $\text{SendRate}_{\min}$ prevents the underutilization of the network that sometimes occurs due to the downward fluctuations of the newly computed send rate. In other words, RRTP takes a pre-emptive, bounded LIMD approach to flow and congestion control. Additional details regarding RRTP's congestion avoidance mechanism will be presented later in this section.

In protocol design terminology, an epoch refers to a certain interval of packet interchange. In RRTP, an epoch is defined to be the interval in which 10 packets are sent or received. The value 10 was experimentally determined to offer a good compromise between the average send buffer size and the timeliness of the communication between the RRTP sender and receiver. The single most important parameter in RRTP's flow control and congestion control mechanisms is the packet interval time, which is measured between every 2 consecutive packets sent or received. The adjustment of send rate critically depends on the ratio between the sending intervals and the receiving intervals among consecutive packets. RRTP keeps track of a pair of running averages, a long-term average and a short-term average, for both the sending and receiving packet intervals. The long-term running average is calculated using the packet interval measurements obtained during the most recently observed 50 packet intervals. The short-term running average is computed at the end of each epoch, using the previous 10 packet interval values. The two values chosen here, 50 and 10, were also experimentally determined to offer good compromises between the relevant space and time considerations. The actual formulas for the calculations are presented in Figure 3.4 shown below:

$$SendInterval_{LongAveInit} = \frac{SendInterval_1 + SendInterval_2 + \ldots + SendInterval_{50}}{50}$$

$$SendInterval_{LongAve} = SendInterval_{current} \times 0.02 + SendInterval_{LongAve} * 0.98$$

$$SendInterval_{ShortAveInit} = \frac{SendInterval_1 + SendInterval_2 + \ldots + SendInterval_{10}}{10}$$

$$SendInterval_{ShortAve} = SendInterval_{current} \times 0.1 + SendInterval_{ShortAve} * 0.9$$

Figure 3.4: Formulas for Computing Packet Interval Running Averages

Once the long term running averages are computed, they are used for calculating the send/receive rate ratio, which is critical for fine-tuning the current send rate. A significantly discrepancy between the current short-term and long-term running average most likely indicates that the dynamics of the network has been perturbed. During such times, the short-term average is used for the purpose of send rate adjustment in place of the long-term average in order to accurately reflect the latest network conditions. If major discrepancies between the long-term and short-term averages are observed for 5 or more consecutive epochs, RRTP assumes that there has been a permanent change in network dynamics and the old long-term average value is replaced with the current short-term average value. The value 5 here was chosen in an intuitive yet conservative manner. Further optimizations and justifications can be carried out via simulations in the future.

Because of the fact that the send rate is only adjusted at the end of each epoch, constant fluctuation of network traffic is minimized, resulting in a more stable network. At the time of send rate adjustment, the newly adjusted rate is subjected to comparison with two parameters mentioned previously: $SendRate_{max}$ and $SendRate_{min}$. In other words, the new rate must fall within the range of $SendRate_{min}$ to $SendRate_{max}$. This is to minimize the chance that an overshoot will result when RRTP linearly ramps up the send rate during its congestion avoidance phases, and no unnecessary reduction in the send rate will result during multiplicative decrease phases. Just as in the case of packet interval running averages, $SendRate_{max}$ and $SendRate_{min}$ are continuously refined to reflect the changing dynamics of the network. In the event of severe path congestion, $SendRate_{min}$ is adjusted downwards in an exponential fashion in order to avoid further burdening the network.

The send rate adjustment is carried out using the following algorithm: first, RRTP defines an additive increase factor α with different initial values based on the type of network RRTP is operating on as well as a multiplicative decrease factor β with an initial value of 0.05. If send/receive rate ratio is greater than 1.05, that means RRTP is operating at a level above maximum allocated bandwidth. RRTP treats such situations as signs of incipient congestion and will do the following adjustment: $SendRate_{new} = SendRate_{prev} * \max((1-\beta), 0.5)$. The value of β will be doubled for every consecutive multiplicative decrease phase with the upper bound of $1-\beta >= 0.5$. The expression $\max((1-\beta), 0.5)$ ensures that the rate reduction factor will never drop below 0.5. In other words, when RRTP first detects signs of incipient congestion, it gently scales down the send rate with a small β value. If the incipient congestion persists, the value of β will be doubled every epoch to more effectively suppress incipient congestions. When β reaches 0.5, RRTP essentially assumes that a serious congestion has occurred and will continue to downward adjust the send rate by half at a time until the network is not longer in a congested state. On the other hand, if send/receive rate ratio is less than 0.95, then RRTP is operating well below the maximum network capacity. This translates into a linear increase (or congestion avoidance) phase in which $\text{SendRate}_{new} = \text{SendRate}_{prev} + \alpha$. In addition, β is reset to its initial value of 0.05 at the start of every congestion avoidance phase. Please note that the suitability of the set of initial and boundary values used in this algorithm was experimentally verified in simulation. Future research efforts can be spent on the tuning of this set of values over real-life network platforms.

Since the purpose of send rate adjustment is to probe the current network capacity and try to achieve full bandwidth utilization, a characteristic send rate can be chosen to realize this objective. RRTP provides an estimate of this characteristic value by averaging the send rate during epochs in which send rate stays constant ($0.95 < \text{send/receive ratio} < 1.05$). This estimate is put into use on the second consecutive epoch in which the send rate stays constant.

With the rate-based congestion avoidance mechanism described above, RRTP should be able to avoid congestion that would otherwise be encountered by TCP. However, major perturbations to the network dynamics will still cause congestions that RRTP is unable to avoid. Under ill-fated network conditions such as link failures or major router overflows, RRTP responds in a similar manner as TCP Reno would by aggressively reducing the send rate. Such efforts are necessary to clear out the congestions and re-establish equilibrium on the network. When the signs of congestion disappear, RRTP undergoes either a slow-start or a fast-recovery phase, depending on the severity of the original congestion.

3.4.2 Flow Control and Congestion Control for Wireless-last-hop Networks

For wireless-last-hop networks, RRTP takes a more aggressive approach to flow control and congestion control due to the dedicated channel assumption: since there will be at most one effective traffic flow between the sender and the receiver over a wireless-last-hop platform, the best flow control strategy is to maintain the send rate at the level of full wireless channel bandwidth utilization.

The bandwidth allocation for a mobile device may change from time to time, depending on the number of users in the corresponding cell and the frequency of handoffs incurred by the mobile device. Consequently, send rate needs to be adjusted whenever events leading to bandwidth reallocation occur. The adjustment process is simpler for wireless-last-hop networks. Instead of taking the standard LIMD approach, the new send rate is obtained by scaling the most recent send rate with the send/receive rate ratio.

From a usage pattern point of view, traffics on wireless-last-hop networks tend to be dominated by short-lived flows. As it was mentioned before, the shorter the connection lasts, the less likely that significant perturbations would occur during the connection. As a result, it is safe to assume that for short-lived flows, the available bandwidth usually remains unchanged for the duration of the connection. Under such conditions, the send rate does not need to be adjusted provided that it is set to the characteristic value at the start of the connection.

The common cause for packet losses over wireless-last-hop networks is link failure due to the high BER of the wireless mediums. However, packet loss due to congestion may occasionally occur as the wired part of the network becomes heavily congested. Under such circumstances, RRTP resorts back to its regular LIMD approach to congestion control. In order to properly identify the causes of packet losses, loss differentiation algorithms (LDA) are employed to make the judgment calls. The detail of LDA is presented in section 3.6.

3.5 Reliability

RRTP achieves end-to-end transmission reliability by using a combination of ACK's and timeouts. Three types of ACK's are used in RRTP: cumulative acknowledgment (CACK), negative acknowledgment (NACK), and selective acknowledgment (SACK).

CACK's serve as confirmations for correctly received packets. The receiver sends out a CACK for every 32 consecutive packets received. When the sender gets a CACK, it can safely flush out the corresponding acknowledged packets from its send buffer. Suppose that since the last CACK sent, the receiver has properly received 31 consecutive packets and the 32nd packet is lost, the receiver will not send out any additional CACK's until the lost packet is retrieved. This situation can be generalized to multi-packet loss events.

NACK's are designed for quickly recovering a single lost packet. When the receiver detects a lost packet event, it sends out a NACK requesting the sender to resend the lost packet. For each unique NACK sent out, the receiver sets up a corresponding timer. After a timer expires (hereafter denoted as a timeout), the receiver will retransmit the corresponding NACK to the sender. This process will continue until the lost packet is properly received. RRTP's timeout mechanism uses the RTT value as the base unit for the timer's waiting period. In order to ensure that the RTT value used in the timeout interval calculation does not come from a transient, non-representative measurement, running averages for RTT are calculated at the end of every epoch and used in the timeout interval calculation.

NACK's are very useful for recovering from single packet loss events. However, when multiple packets are lost in close successions, using NACK to recover them individually is inefficient due to high timer overhead. SACK solves this problem by offering the capability of handling multiple packet losses in aggregation. SACK enables the receiver to advertise the lost packets to the sender and recover these lost packets without frequent triggering of the congestion control mechanism. For long fat pipe networks, SACK is essential for maintaining a healthy throughput in the presence of packet losses. In order to avoid frequent triggering of the slow start mechanism, the sender should have an accurate assessment of the condition of the channel and the receiver. SACK enables the sender to paint a complete picture of the receiving queue at the receiver and thereby inferring the channel conditions with greater accuracy.

3.6 Loss Differentiation Algorithm

Several papers published on the TCP performance enhancement over wireless networks have considered sender-based loss differentiation. RRTP, on the other hand, is based on the notion that the receiver usually has more accurate and timely knowledge of packet losses. Consequently, the RRTP receiver is responsible for determining the cause of a particular loss and informing the sender to take the appropriate action.

For wireless-last-hop networks, a variant of the LDA proposed by Biaz and Vaidya [1998] is adopted. Two crucial assumptions need to be made here. The first one is that the wireless link has the lowest bandwidth and thus is the bottleneck of the network. The second one is that the wireless base station serves strictly as a routing agent between the wired and the wireless network. As one can see quite easily, with the big difference in bandwidth between wired LAN (100 Mbps – 1 Gbps) and wireless WAN (20 - 500 Kbps), packets traveling on the wired network will get congested at the base station while adapting to the lower send rate imposed by the wireless network. As a result, the packets transmitted over the wireless link tend to be clustered together. If a packet loss occurs due to random wireless transmission errors, the receiver should be able to observe a certain time interval in which the packet is expected but not received. Such an event can be interpreted to be the sign of wireless loss due to link failures. Following this line of reasoning, RRTP can distinguish wireless losses from congestive losses using the following heuristics: Let T_{min} be the minimum interpacket duration observed by the receiver and let $T_{separation}$ be the interval between the time when the last correct packet is received and the time when the lost packet is detected by the receiver. Now suppose n consecutive packets are lost at some point during the communication, the loss is characterized as a wireless loss if the following relation holds: $(n + \sigma)*T_{min} < T_{separation} < (n + \varphi)*T_{min}$. Intuitively, σ should assume a value around 1 while φ should assume a value around 2. In order to determine the set of values for σ and φ that results in the least number of misclassifications, a set of simulations are performed using the topology illustrated in Figure 3.5 shown below:

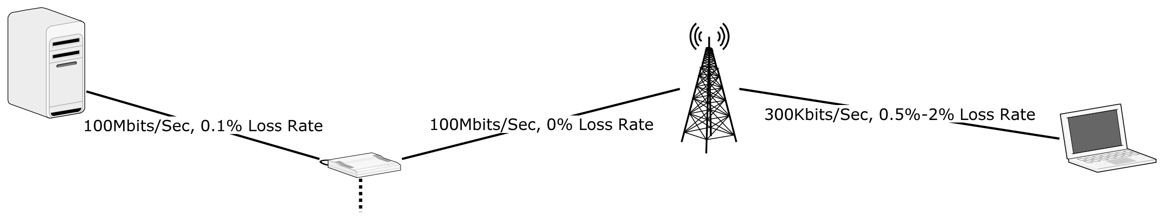

100Mbits/Sec, 0.1% Loss Rate
100Mbits/Sec, 0% Loss Rate
300Kbits/Sec, 0.5%-2% Loss Rate

Figure 3.5: Wireless-last-hop Topology with Both Congestive and Wireless Losses

In each simulation, a certain number of congestive and wireless losses are generated. On the receiver side, the number of packet loss misclassifications for each simulation trial is recorded in Table 3.2 shown below:

Table 3.2: Misclassification Rate With Respect to Parameter Value Adjustments

σ	φ	C-to-W Misclassifications	W-to-C Misclassifications	Misclassification Rate
0.5	2	2	8	20%
0.75	2	1	2	6%
1	2	0	0	0%
1.25	2	0	4	8%

1	1.75	0	3	6%
1	2.25	2	3	10%
1	2.5	3	7	20%

The simulation results clearly demonstrate that the rate of misclassifications converges to 0 as the values of σ and φ converges to 1 and 2 respectively. Consequently, this pair of values for σ and φ is used for determining the nature of a packet loss in the wireless-last-hop environments.

Figure 3.6 shown below illustrates the inner-workings of the loss differentiation algorithm under three different scenarios.

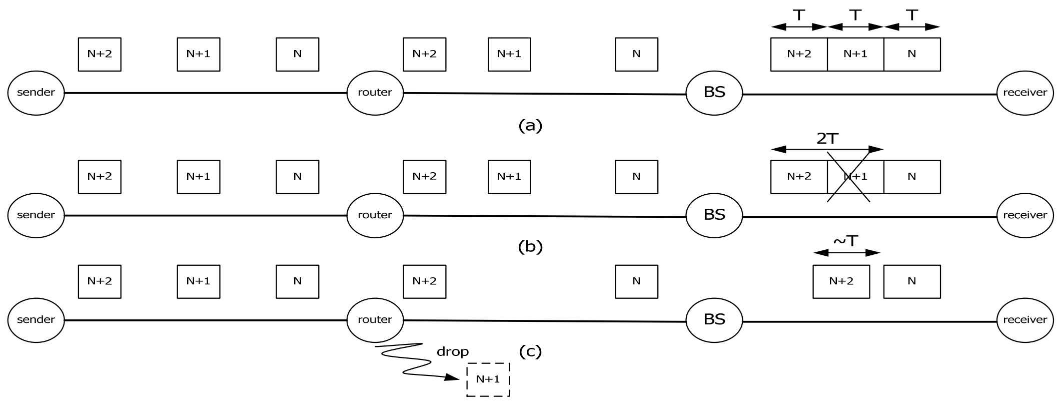


Figure 3.6: Packet interval based loss differentiation algorithm. (a) Normal network condition. (b) Packet loss on wireless link. (c) Packet loss on wired link.

For the wireless LAN topology, the assumptions made in the previous situation are usually not true as the throughput difference between wired and wireless LANs is not significant. As a result, packets do not necessarily travel in close successions on wireless LAN connections and the Biaz [1998] LDA will not perform as well as in the case of the wireless-last-hop topology. An alternative approach needs to be used here to distinguish between wireless loss and congestion losses. The Spike scheme proposed by Tobe et al. [2000] can be useful in this situation.

When used in the wireless backbone/LAN topology, the Spike scheme performs significantly better than the Biaz scheme in terms of loss differentiation accuracy. The reason for this performance advantage is mainly due to the fact that the Spike scheme uses the relative one-way trip time (ROTT) measurements as congestion indicators. ROTT, measured at the receiver, is the time between the moment when the packet is sent and the moment when the packet is received. During periods of smooth traffic flow, ROTT measurements will remain relatively stable. When the network starts to become congested, the receiver will detect rising ROTT values. The default behavior of RRTP in this situation is that the receiver will issue an explicit incipient congestion notification to the sender to throttle the send rate. In the event that the rise in ROTT values is coupled with packet losses, the receiver can be confident that the packet losses are caused by congestion. However, if the packet losses are not accompanied by a rise in ROTT value, the receiver will categorize these losses to be due to wireless errors, and consequently will not trigger the congestion control mechanism.

As it is discussed above, the Biaz scheme and Spike scheme work well on different wireless network platforms. Depending on the actual wireless network in use, RRTP selects the appropriate LDA to achieve the desired result.

3.7 Theoretical Performance Comparison

The send rate of a transport layer protocol can be approximated by dividing the data packet size by the expected value of network latency time. For non-wireless-last-hop networks, both TCP and RRTP undergo three phases of operation: slow-start, congestion avoidance, and error recovery. The latency time can be approximately broken down according to these three phases of operation, denoted respectively as: T_{ss} (latency due to slow start), T_{ca} (latency due to congestion avoidance), and T_{er} (latency due to error recovery). The mathematical expression of the send rate is shown in Figure 3.7:

$$SendRate = \frac{DataPacketSize}{E(T_{ss}) + E(T_{ca}) + E(T_{er})}$$

Figure 3.7: Send Rate Calculation

3.7.1 Short-lived Connections under the LIMD Constraints

For short-lived connections subjected to the LIMD constraints, TCP spends a significant portion of the connection duration in the slow-start phase. In contrast, RRTP can avoid the slow-start phase altogether in the best case (user pre-seeded parameters). Even in the worst case, RRTP enjoys a more efficient bandwidth probing process than TCP due to its enhanced rate-based send rate adjustment method. It is well known that the pattern of TCP send rate plotted against time is roughly of a saw tooth shape. RRTP's rate-based congestion avoidance has a smoothing effect on this saw tooth shape as it allows RRTP to avoid many unnecessary slow-start's that TCP encounters by default. Concluding from the previous three lines of reasoning, it is safe to state that generally speaking, the $E(T_{ss})$ value is smaller for RRTP than TCP. This implies the following relationship: $SendRate_{RRTP} \geq SendRate_{TCP}$.

3.7.2 Short-lived Connections under the Relaxed LIMD Constraints

For short-lived connections running over wireless-last-hop networks, the LIMD constraints can be significantly relaxed for reasons mentioned in previous sections. As a result, the term $E(T_{ss})$ can be eliminated for RRTP's send rate calculation. Since all terms in the send rate formula are positive, it is easily seen from the expression in Figure 3.8 that RRTP will consistently outperform TCP under the assumption that all terms within the expression are approximately equal in value for RRTP and TCP.

$$SendRate_{RRTP} = \frac{DataPacketSize}{E(T_{ca}) + E(T_{er})} > \frac{DataPacketSize}{E(T_{ss}) + E(T_{ca}) + E(T_{er})} = SendRate_{TCP}$$

Figure 3.8: Send Rate Comparison between RRTP and TCP

Thus far, this chapter has provided a thorough description of RRTP's inner workings. The next chapter will present a set of experimental results that compare the performance of RRTP and several TCP variants on both simulated and real-life network platforms.

Chapter 4
Experimental Setups and Results

This chapter examines the two sets of experiments that were carried out to provide an empirical understanding of the performance characteristics of RRTP. The first set of experiments comparatively studies the performance of RRTP and TCP Reno in term of throughput. The second set of experiments examines the impact that the three configurable parameters have on the performance of RRTP. For the purpose of these experiments, two versions of RRTP were implemented: one runs on top of the ns-2 simulator and the other one operates directly over Linux/Windows sockets.

Judging from the existing literatures in TCP algorithm research, few scholars have gone beyond the stage of testing their proposed solutions in simulation. There may be a number of reasons why network protocol researchers have adhered to a simulation-based theory validation paradigm: high cost of physical implementation and testing, normative pressure from the research community, time pressure in the race to be the first to publish novel results, etc. Perhaps a more fundamental reason could be that a large portion of researchers in the fields of applied, natural and social sciences are trained to be quantitative positivist methodologists. The quantitative positivist research paradigm places very stringent requirements and standards on hypothesis testing as well as on empirical validations of designs, systems and theories. For this reason, researchers that subscribe to this paradigm tend to favor more controllable forms of testing and experimentation in the process of establishing the desired levels of internal validity and causality. This is particularly true in the area of protocol research as a large body of research work dedicated to address TCP's performance issues has been relying solely on simulations to verify their claims. Although simulations are effective at establishing the internal validity of a study, this type of methodologies suffers from an inherent lack of realism. By demonstrating that in simulation, protocol X outperforms TCP Reno, the researchers only manage to establish their causal claims (e.g. enhancement A caused the superior performance of protocol X) in simulated environments. Empirical tests in real settings are necessary for establishing externally valid claims and the practical significance of the research. In this piece of research, a large amount of effort has been dedicated to realize a platform-independent (both Windows and Linux compatible) physical implementation of RRTP for the purpose of empirically demonstrating the performance merits of RRTP over real-life network platforms.

The rest of this chapter presents the experimental setups as well as the corresponding analyses of the results.

4.1 Simulation Experimental Setups

The ns-2 simulator is the de facto software tool that researchers use for protocol performance benchmarking and other types of network analysis. As a result, it is used to create the simulated test-bed for studying the performance of RRTP in simulation. The detailed experimental setup is summarized in Table 4.1 shown below:

Table 4.1: ns-2 Simulation Setup

Setup Parameters	**Descriptions**
Simulation Software Components	ns-2.32, Tcl/Tk-8.4.14, otcl-1.13, tclcl-1.19, nam-1.13, xgraph-12.1, perl-5.6.2, tcl-debug-1.7, dmalloc-4.8.0, zlib-1.2.3
PC Processor & RAM	Pentium D945 3.4 GHz, 2 GB
Operating System	Fedora 7

4.2 Simulation Results

To evaluate the performance of RRTP, several representative simulation scenarios were created using the ns-2 simulator. Tests were conducted in these simulated environments to comparatively study the performance of RRTP, TCP Reno, TCP New Reno and TCP Vegas. The specifications used for each of the testing environments are chosen to closely reflect the observed values in real-life networks. For the high latency high bandwidth environment, a dedicated fat pipe with 1 Gbps bandwidth and 200 ms one-way latency is assumed. Such network infrastructures are often used for inter-continental corporate VPNs. For the CDMA environment, only the 1xRTT mode is considered here since it is the de facto standard in both metropolitan and non-metropolitan areas. The theoretical maximum physical layer downlink bandwidth of 144 Kbps is assumed here. For the satellite environment, a shared download carrier typically has a downlink bandwidth from 1 to 40 Mbps and is shared up to 100 to 4000 concurrent end users. An average bandwidth of 20 Kbps per end user is assumed here.

The one-way latency is assumed to be 300 ms since the typical distance between a satellite and a ground antenna is around 70,000 kilometers. Among all the environments, LAN is assumed to have the lowest packet loss rate of 0.1% and the lowest one-way latency of 1 millisecond. For the wireless LAN setup, we assume the standard 802.11b bandwidth of 11 Mbps and a one-way latency of 5 milliseconds. Table 4.2 shown below outlines the performance of the four protocols in terms of the total number of packets sent in the fixed time interval of 60 seconds. The packet size used for the experiments was uniformly set to be 1400 bytes (within the maximum transmission unit limit to avoid IP fragmentation). The specifications of the testing platforms are summarized in Table 4.3 shown below:

Table 4.2: Total Number of Packets Sent Per Connection Period

Environment	RRTP	Reno	New Reno	Vegas
High Latency High Bandwidth	1290696	242323	241245	212471
CDMA 1X	758	473	481	589
Satellite	102	51	54	79
LAN	514563	513987	514198	514372
Wireless LAN	58038	55671	55606	56933

Table 4.3: Testing Environment Specifications

Environment	Bandwidth	One-way Latency	Loss Rate	Test Interval
High Latency High Bandwidth	1 Gbps	200 ms	1%	60 sec
CDMA 1X	144 Kbps	75 ms	1%	60 sec
Satellite	20 Kbps	300 ms	1%	60 sec
LAN	100 Mbps	1 ms	0.1%	60 sec
Wireless LAN	11 Mbps	5 ms	1%	60 sec

In the high latency high bandwidth environment, protocols that rely on sender-receiver feedbacks will likely suffer from their slow responses to the changing network condition (e.g. bandwidth throttle due to the emergence of new traffic). This is due to the fact that the connection's RTT is very large and consequently, it is difficult to rely on receiver feedbacks to adjust the send rate. Fairness can be severely limited as newly entered traffic will almost always be starved by previously established traffic. However, because of the fact that RRTP is reconfigurable, good estimates of the network conditions can be provided to the application before the transfer starts, allowing a much higher throughput than conventional TCP as demonstrated in Table 4.2.

Both the CDMA network and the satellite network are considered to be wireless-last-hop topologies. As demonstrated in Table 4.2, RRTP performs much better than TCP Reno and TCP New Reno over both the CDMA and the satellite networks. This is expected because whenever packet losses are encountered, conventional TCP implementations invoke the congestion control mechanism right away without making an effort to distinguish among the different types of losses. RRTP, on the other hand, is able to avoid the unnecessary triggering of congestion control in the event of random wireless packet losses.

In the case of wireless LAN, the Spike LDA enables RRTP to differentiate between congestion losses and wireless losses, resulting in superior performance as shown in Table 4.2. Furthermore, it can be seen from Table 4.2 that RRTP also offers competitive performance over the conventional LAN platform. This is expected since RRTP's operation closely resembles that of TCP Vegas in the absence of network abnormalities.

In addition to the analysis provided above, studying the effects of packet loss rate can provide further insight on the performance characteristics of RRTP. In section 4.2.1, the correlation between RRTP's performance advantage and the network's packet loss rate is studied in simulation. Here, we choose to concentrate on the wireless-last-hop topology because it is the principal platform on which all of the real-life experiments are carried out.

4.2.1 Effect of Packet Loss Rate Variation

Intuitively, when operating over wireless-last-hop networks, the amount of performance gain that RRTP is able to achieve over other TCP variants should be positively correlated with the packet loss rate of the network. To verify this intuition, a set of simulations (each of 60 seconds in length) is carried out to study the performance sensitivity to varying packet loss rate for TCP Reno, TCP Vegas, TCP Westwood, and RRTP. TCP Westwood is designed to overcome some of the difficulties that traditional TCP faces in the wireless-last-hop environment. Consequently, it is chosen here instead of TCP New Reno to serve as a more suitable benchmark. The results of the simulations are illustrated in Figure 4.1 and Figure 4.2 shown below:

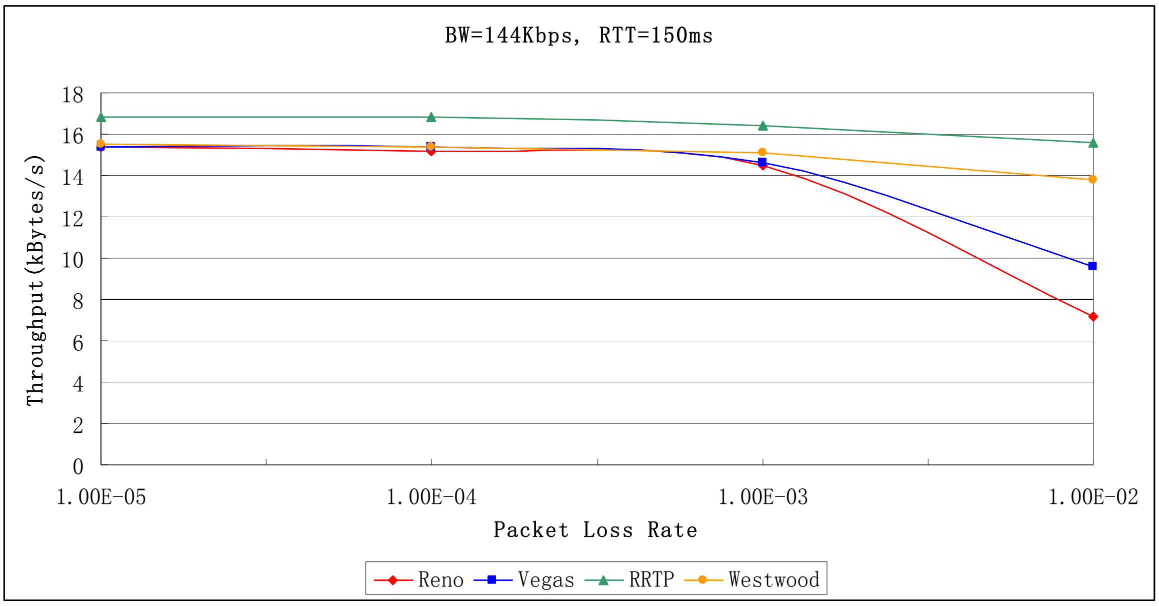


Figure 4.1: Throughput Comparisons over CDMA Network with Varying Loss Rate

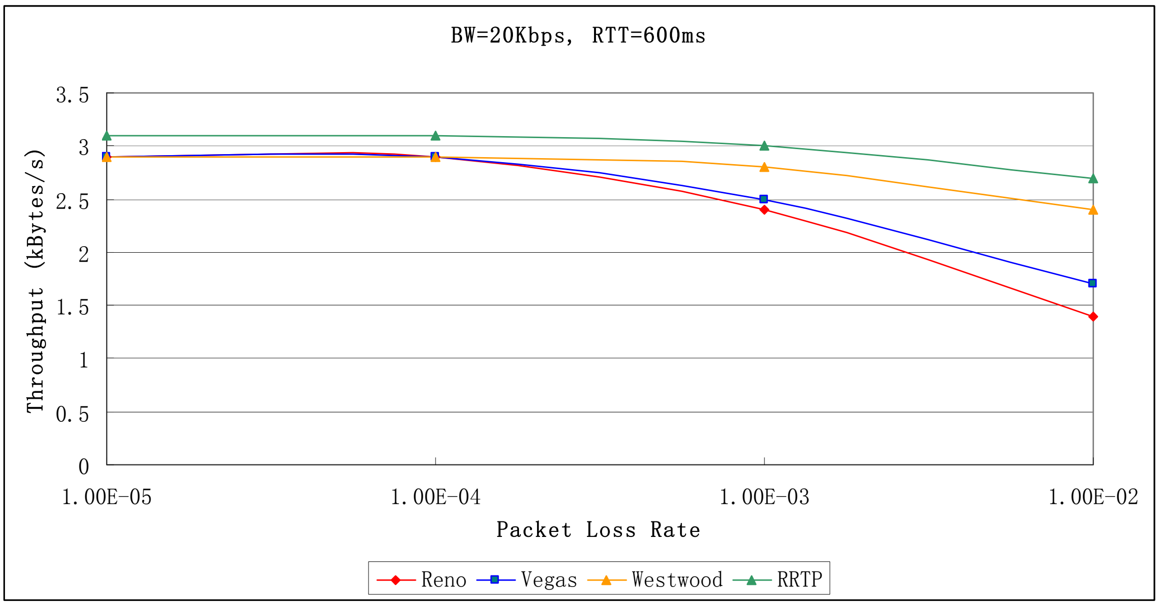


Figure 4.2: Throughput Comparisons over Satellite Network with Varying Loss Rate

Judging from the results shown in Figure 4.1 and Figure 4.2, the performance advantage of RRTP tends to become more pronounced as the packet loss rate increases. It is also evident from the simulation results that RRTP enjoys a certain amount performance gain even when the loss rate is negligible. This phenomenon can be attributed to the user-configurable nature of RRTP. By pre-setting RRTP's transmission rate, the users can ensure that RRTP is able to operate at close to the maximum throughput at the commencement of a connection (assuming the single-flow scenario, which is usually valid for wireless-last-hop networks).

4.3 Real-life Experimental Setup

For the real-life experiments, two desktop servers running on the University of Waterloo campus network are used as fixed hosts. Two laptops with identical hardware specifications are used as mobile hosts. Table 4.4 and Table 4.5 shown below summarize the setups for the experiments carried out on real-life cellular networks.

Table 4.4: EVDO Setup

Setup Parameter	**Description**
Fixed Host CPU & RAM	Pentium D945 3.4 GHz, 2 GB
Mobile Host CPU & RAM	Intel Core Duo 1.73GHz, 2 GB
Operating System	Fedora 7/Windows XP
Fixed Host Network Interface	Intel® PRO/1000 PM Network Connection
Connection Card	Sierra Wireless AirCard 595 Qualcomm 3G CDMA

Table 4.5: EDGE Setup

Setup Parameter	Description
Fixed Host CPU & RAM	Pentium D945 3.4 GHz, 2 GB
Mobile Host CPU & RAM	Intel Core Duo 1.73GHz, 2 GB
Operating System	Fedora 7/Windows XP
Fixed Host Network Interface	Intel® PRO/1000 PM Network Connection
Connection Card	Sony Ericsson PC300 GSM 850/900/1800/1900 MHz HSDPA/UMTS 850/1900/2100 MHz

4.4 Real-life Experimental Results

In order to show that RRTP not only outperforms TCP in simulation but also on real networks, Bell Canada`s CDMA network and Rogers Communications Inc.'s GSM network, are chosen to benchmark the performance of RRTP against that of TCP. The experiments are carried out on a wireless-last-hop topology as illustrated in Figure 4.3 shown below:

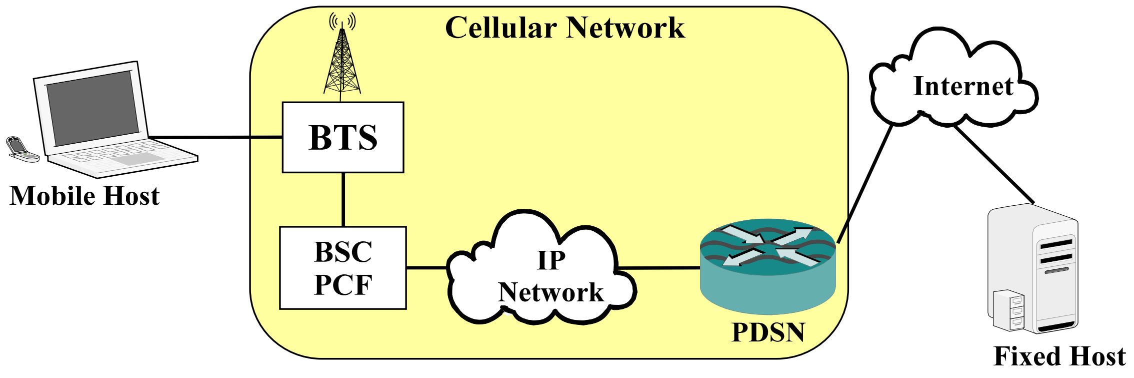


Figure 4.3: Real-life Cellular Network Testbed Setup

A version of the file transfer protocol called QuickGet was created to benchmark the performance of RRTP against TCP. QuickGet allows the user to choose the transport layer protocol to be used prior to the file transfer. Since the bandwidth and the delay characteristics of cellular networks fluctuate quite often, each experiment is repeated multiple times and the final result is taken to be the average of all valid trial results. Please note that any trial result that falls outside 3 times its corresponding standard deviation is considered to be an outlier and consequently excluded from the calculation of the final result.

Bell Canada’s CDMA network can operate under two principal modes that differ quite drastically in transmission speed: the 1xRTT mode which has a peak download speed around 144 Kbits/sec, and the EVDO mode which boasts a peak download speed around 3.1 Mbits/sec. Since certain RRTP components such as the loss differentiation algorithm are quite sensitive to the underlying network’s transmission speed, one set of experiments is performed for each mode.

Rogers’ GSM EDGE network employs a different set of link-layer technologies than Bell’s CDMA EVDO network. In order to show that RRTP`s performance advantage over TCP is not technology specific, the benchmarking of RRTP against TCP is also carried out on the GSM EDGE network.

As it was mentioned in section 3, RRTP is expected to outperform TCP on cellular networks due to two principal reasons: RRTP’s rate-based algorithms are more adaptive to cellular networks than TCP’s conventional algorithms, and user-configurable parameters allow RRTP to fine-tune its performance over a certain network at a much higher efficiency than TCP. In order to separately demonstrate the resulting performance gain due to these two factors, two sets of experiments are performed for both EVDO and EDGE networks. One set of experiments benchmark the performance of native RRTP (non-user-configured) against that of TCP. The other set of experiments benchmark the performance of pre-configured RRTP against that of TCP.

In order to study the effect that mobility has on performance, experiments are carried out at three to four different mobility settings: stationary, moving at 5 km/hr, moving at 60 km/hr, and moving at 100 km/hr. For experiments carried out under the 100 km/hr setting, a portion is performed aboard a car driving on highway 401 between Toronto and Waterloo, and the rest is performed aboard a VIA Rail train traveling between Toronto and Montreal. Sections 4.4.1 - 4.4.3 describe the results of the experiments in detail.

4.4.1 CDMA 1xRTT Mode

In order to assure the validity of the test results, experiments are performed in a manner that eliminates all the controllable discrepancies between the RRTP trials and the TCP trials. Two pairs of client and server, with identical hardware and software setups, are used in the experiments. One pair uses RRTP and the other pair uses TCP.

The two principal variables in the experiments are: the size of the file transferred, and the signal quality of the CDMA network connection. It is expected that the signal quality of the network connection would deteriorate due to frequent cell switching and signal fading as the mobile host travels at high speed. In order to characterize the scenarios where RRTP offers the most performance gain over TCP, we carry out the same experiment under a number of different environments. Each test scenario is repeated multiple times and the averaged performance gain for each case is recorded in Table 4.6 shown below:

Table 4.6: Performance Gain of RRTP over TCP on a Real-life CDMA 1xRTT Network

File Size / Speed	100KB	1MB	10MB
Stationary	29.31%	16.24%	10.82%
5 km/hr	31.63%	17.46%	11.34%
60 km/hr	36.59%	21.97%	14.91%
100 km/hr	38.12%	26.97%	18.17%

As it can be seen from the results, RRTP significantly outperforms TCP (in term of throughput) for small file transfers. This is expected since RRTP is designed to converge to the ideal transfer rate much faster than TCP (which takes a long time to reach full utilization of the available bandwidth due to its slow-start mechanism) does. However, this initial fast-convergence performance advantage becomes less pronounced as the size of the file transferred increases. This trend is easily seen from the data presented in Table 4.6.

In addition to the significant edge that RRTP enjoys when it is used to transfer small sized files, it also handles poor quality wireless connections better than TCP does. As it can be seen from the test results, the performance advantage of RRTP becomes more pronounced as the mobile connection is maintained at increasingly higher speed of travel (quality of communication deteriorates due to lower signal quality and frequent hand-over as speed of travel increases).

4.4.2 CDMA EVDO Mode

When the CDMA network operates under the EVDO mode, the expected throughput is significantly higher than when it operates under the 1xRTT mode. This is because the forward error correction (FEC) and the link-layer retransmission mechanisms have been implemented for EVDO to address the challenges associated with radio propagation environments.

For CDMA EVDO networks, the peak downlink and uplink rates are 3.1Mbps and 1.8Mbps per user respectively. Mobile Internet users with CDMA EVDO terminals use Internet applications such as web browsing, multimedia streaming, or email. Therefore, the efficiency of the transport protocol is important to achieve the maximum throughput over the error-prone wireless link with fluctuating bandwidth, large delay, and jitter.

In order to thoroughly compare the performance of TCP and RRTP over an EVDO network, a set of experiment is designed, with two variable parameters – the size of the file to be transferred and the traveling speed of the mobile host. Since the throughput of the CDMA network under the EVDO mode is much higher than what it is under the 1xRTT mode, a 1MB file is designated as a small file and a 10MB file is designated as a large file for the purpose of the experiments.

Table 4.7 summarizes the results of the experiments performed on the EVDO network. The first two columns indicate the relative performance gain that RRTP enjoys over TCP. The last two columns demonstrate that when RRTP is pre-configured with accurate network parameters, significant performance gain can be realized over TCP. Two overall trends can be identified from Table 4.7: RRTP offers a more significant performance improvement for the transmissions of small files, and the performance advantage of RRTP is more pronounced when the file transfer takes place in environments with high packet loss rate.

Table 4.7: Performance Gain of RRTP over TCP on a CDMA EVDO Network

File Size / Speed	RRTP		RRTP++	
	1MB	10MB	1MB	10MB
Stationary	6.46%	1.31%	13.84%	3.89%
5 km/hr	6.50%	1.43%	14.28%	4.34%
100 km/hr	8.98%	1.73%	17.50%	6.68%

Figure 4.4 shows the standard deviation figures for the experimental trials. The standard deviation values for TCP test trials are evidently higher than that for RRTP test trials. This is expected since the congestion control mechanism within RRTP is designed to avoid congestions, thereby reducing the frequency of send rate throttling, which is carried out by the sender upon detection of network congestion. Furthermore, since RRTP initiates an instantaneous recovery (instead of a slow-start or a fast recovery) after handling a congestion-related packet loss, it should observe a more stable pattern of send rates throughout the duration of a connection. In other words, transmissions using RRTP are less bursty than those using TCP. Appendix B-D contain the detailed figures that document the performance comparisons for each experiment.

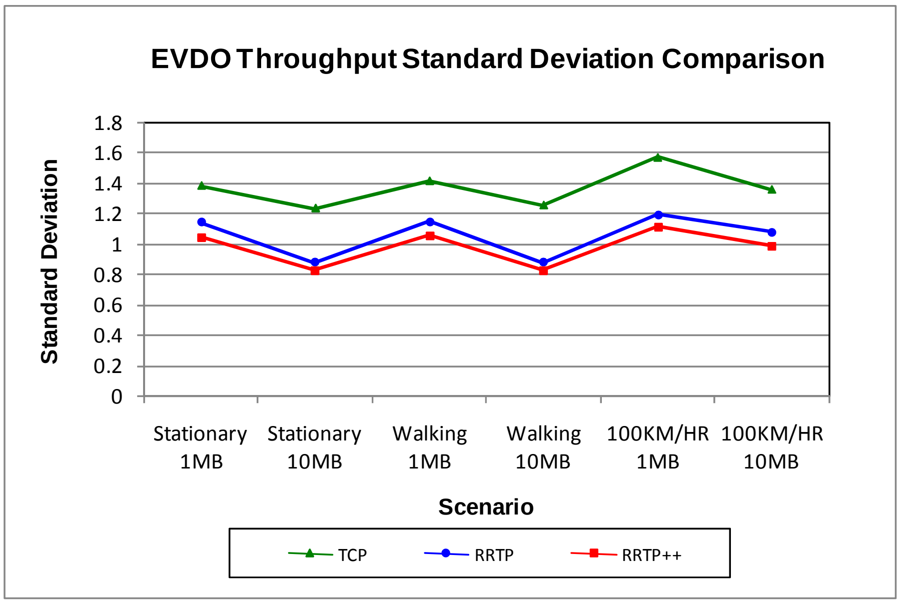


Figure 4.4: A Comparison of Throughput Variance for the Three Protocol Versions

4.4.3 GSM EDGE Network

As it was mentioned previously, benchmark tests were performed on Roger's GSM EDGE network in order to show that the performance advantage that RRTP enjoys over TCP is not platform or technology specific. The EDGE network currently supports a theoretical peak downlink data throughput of 3.6 Mbps and uplink throughput of 384 kbps under the HSPA mode. The same sets of experiments were performed on the EDGE network as they were performed on the EVDO network. Table 4.8 summarizes the results of the experiments. The first two columns show the relative performance gain that RRTP enjoys over TCP. The last two columns demonstrate that when RRTP is pre-configured with accurate network parameters, significant performance gain can be realized over TCP.

Similar results are observed here as in the case of the EVDO network. When pre-configured with an accurate set of network parameters, RRTP demonstrates significant performance gain over TCP on the EDGE network.

Table 4.8: Performance Gain of RRTP over TCP on a GSM EDGE Network

File Size / Speed	RRTP		RRTP++	
	1MB	10MB	1MB	10MB
Stationary	6.93%	1.35%	13.99%	4.50%
5 km/hr	8.04%	1.57%	14.13%	4.81%
100 km/hr	9.65%	2.15%	16.02%	5.58%

Figure 4.5 shown below illustrates the standard deviation plots for the EDGE network experiments. The results are consistent with those for the EVDO network experiments – TCP exhibits higher throughput fluctuation due to its poor adaptability over the EDGE network.

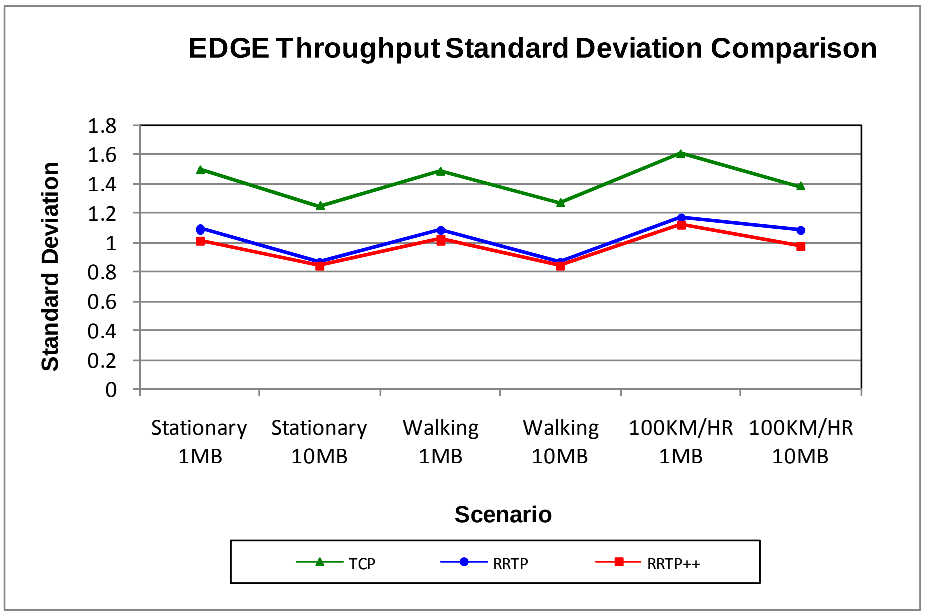


Figure 4.5: A Comparison of Throughput Standard Deviation

4.5 Configurable Parameter Tests

Throughout this thesis, an emphasis has been placed on the importance of tuning and configuring the performance of RRTP in a parameterized fashion. From previous analysis, it is evident that seeding RRTP's algorithms with the proper set of parameter values can result in significant performance gains.

This section attempts to study the effect of key network parameter configuration in a more systematic fashion. For each of the key parameters, a set of experiments is performed to measure the performance sensitivity to the accuracy of the initial parameter configuration. Three scenarios are considered: no initial parameter seeding (corresponds to the “Default Line”), initial parameter seeding with seed value set to 20% below the observed average value (corresponds to the “BelowAve Line”), and initial parameter seeding with seed value set to the observed average value (corresponds to the “Average” line). All experiments are performed in a stationary setting with a transferred file size of 10 MB. The experimental results are documented in Figure 4.6, Figure 4.7, and Figure 4.8 shown below:

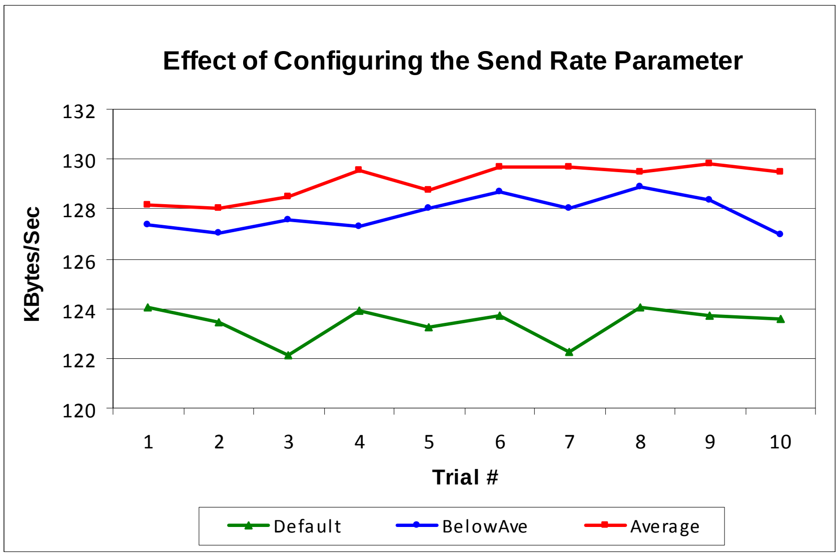


Figure 4.6: Effect of Configuring the Send Rate Parameter

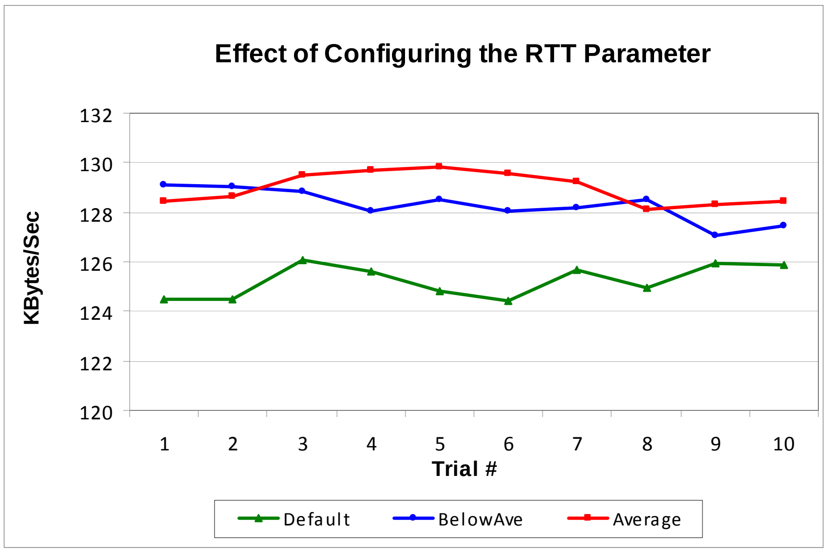


Figure 4.7: Effect of Configuring the RTT Parameter

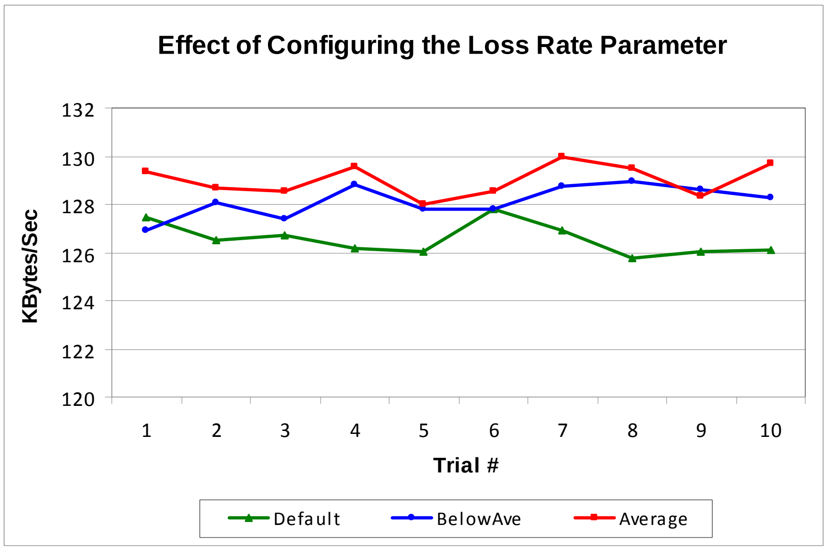


Figure 4.8: Effect of Configuring the Loss Rate Parameter

As it can be seen from Figure 4.6 – Figure 4.8, seeding the flow control and congestion control algorithms with the characteristic parameter values significantly improves the performance of RRTP. This outcome is expected since a more accurately estimated initial value enables RRTP to converge to its optimal operating point at a more rapid pace.

Drawing from the experimental results summarized in this chapter, it is reasonable to conclude that RRTP possesses the right set of design characteristics to outperform TCP over non-conventional network platforms. In particular, the evidence of RRTP's superior performance on cellular networks is very strong. In the next chapter, several design characteristics and limitations of RRTP will be further explored to establish grounds for a more thorough evaluation of RRTP's capabilities.

Chapter 5
Design Level Analyses and Highlights

This chapter compares RRTP with other TCP variants in light of several key design considerations. The purpose of this chapter is to demonstrate that by design, RRTP either matches or outclasses its predecessors regardless of the comparative reference point. Several important limitations of this piece of research are also presented in this chapter to provide the reader of this thesis with a broader understanding of RRTP.

5.1 Fairness of RRTP

As it was mentioned in chapter 3, fairness is an important quality that RRTP needs to possess when operating in network environments with multiple traffic flows. In order to be deemed a fair protocol, RRTP connections that operate in the same channel must all be able to converge to their fair share of the network bandwidth. Judging from its design, RRTP should be able to achieve a similar degree of fairness as TCP Reno does. On multi-flow networks, RRTP takes the LIMD approach to flow and congestion control, thereby avoiding destructive bandwidth competitions among the different connections sharing the same pipe. In order to confirm this intuition, quantitative analyses need to be carried out to demonstrate the fair sharing of the bandwidth among all RRTP connections within the same channel.

The index of fairness, as defined in Jain et al. [1984], can be used to quantitatively study the degree of fairness of RRTP. Figure 5.1 shown below demonstrates the calculation of the index of fairness:

$$f(x) = \frac{\left[\sum x_i\right]^2}{n\sum x_i^{\,2}}$$

Figure 5.1: Formula for Calculating the Index of Fairness

n is the number of connections on a given network and x_i is the throughput of the ith connection. A perfectly fair bandwidth allocation scheme would result in f(x) = 1. On the contrary, a completely unfair bandwidth allocation scheme would result in f(x) = 1/n, i.e., all available bandwidth is consumed by a single connection.

In order to demonstrate that RRTP is a fair protocol, simulations are carried out to arrive at a fairness index value. Figure 5.2 illustrates the setup of the simulations. 30 identical RRTP connections are created to share the 11 Mbit/sec bottleneck link. To investigate the fairness behaviour under varying link condition, the link error rate is adjusted from 0% to 5%. The same set of simulations is repeated for TCP Reno. The results of the simulations are summarized in Table 5.1 and demonstrate the fact that RRTP is indeed a fair protocol.

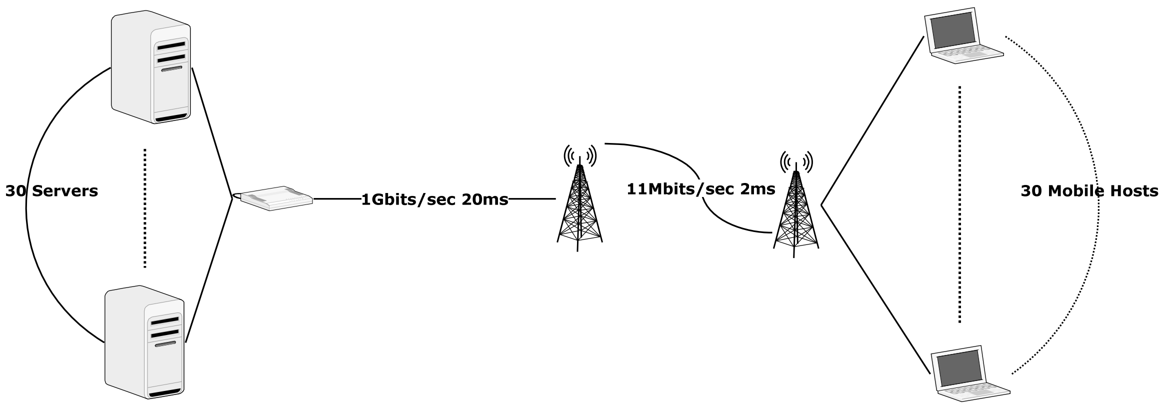


Figure 5.2: Simulation Setup for Fairness Test

Table 5.1: Fairness Test Results

Error Rate (%)	RRTP	TCP Reno
0.0	1.0	0.98
0.1	1.0	0.99
0.5	0.99	0.98
1.0	0.99	0.97
5.0	0.98	0.97

5.2 Friendliness of RRTP

Friendliness is similar in concept to fairness. RRTP can be regarded as TCP-friendly if it can coexist with other TCP variants in the same pipe without causing any of them to starve for bandwidth. TCP-friendliness is a crucial quality that RRTP must possess in order to be considered a viable and readily deployable substitute for TCP. Judging from its design, RRTP is not expected to be dominating in bandwidth acquisition when it is put into coexistence with other TCP variants in the same pipe. RRTP's LIMD approach to flow and congestion control ensures that it will not starve peer connections. However, on certain network platforms such as lossy wireless networks, RRTP is expected to acquire bandwidth in a more aggressive manner as compared to traditional TCP.

In order to demonstrates that RRTP can operate in a friendly manner along with other TCP variants in the same pipe, simulations are carried out to study the bandwidth acquisition characteristics of coexisting traffics running on top of five different transport layer protocols. The setup of the simulations is shown in Figure 5.3. A wireless bottleneck link of 54 Mbits/sec is shared among five connections running on top of TCP Reno, TCP New Reno, TCP Vegas, TCP Westwood, and RRTP, respectively. The results of the simulations are summarized in Table 5.2.

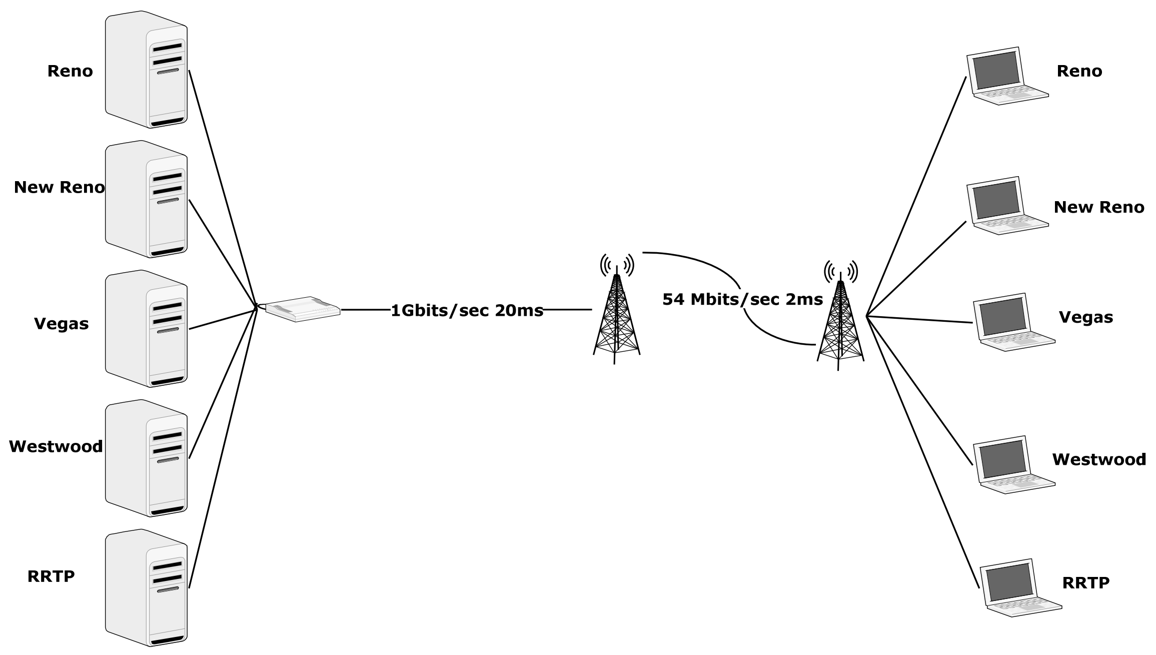


Figure 5.3: Simulation Setup for Friendliness Test

Table 5.2: Friendliness Test Results

Error Rate	Reno Throughput	New Reno Throughput	Vegas Throughput	Westwood Throughput	RRTP Throughput
0.0	8.96	8.96	8.96	8.96	8.96
0.1	6.28	6.33	8.37	7.38	8.82
0.5	4.08	4.22	5.09	7.21	7.89
1.0	2.98	3.02	3.06	5.86	6.97
5.0	1.16	1.16	0.97	2.23	2.45

As it can be seen from Table 5.2, RRTP exhibits a more aggressive bandwidth acquisition behavior in lossy wireless environment. This is expected since RRTP is able to avoid much of the unnecessary triggering of the congestion control mechanism using its LDA mechanism. Furthermore, it can also be observed from Table 5.2 that the other TCP variants are not starved by RRTP and receive their share of the bandwidth. In the case where the error rate of the wireless link is zero, the bandwidth is evenly distributed among the 5 connections.

As it was previously mentioned in section 2.6.4, TCP Vegas does not perform well in a TCP Reno dominated network since TCP Reno employs a more aggressive bandwidth acquisition scheme than TCP Vegas. Evidently, RRTP does not share the same shortcoming as TCP Vegas. RRTP's ability of co-existing with TCP Reno in a friendly yet competitive manner can be regarded as the cornerstone to its deployability over today's heterogeneous internetworks.

5.3 Cross-Platform Adaptability

One of the key selling points of RRTP is its ability to adapt to a wide range of network platforms. As it was described in earlier sections, RRTP's superior adaptability is a direct result of its reconfigurable nature. Consider a pervasive computing scenario where a mobile device frequently switches among several different types of networks. Since each type of networks has its own set of characteristic values for throughput, latency, and loss rate, frequent switching may result in performance hits if active connections get interrupted during the process of switching networks. For most TCP variants, there is little that can be done to avoid this kind of performance hit. RRTP, on the other hand, can gracefully adapt to the new network platform by reconfiguring the affected network parameters to adopt their new characteristic values through a simple API call.

5.4 Deployability and Maintainability

Deployability and maintainability are two important qualities that a piece of software is usually judged on. Generally speaking, a piece of readily deployable and maintainable software normally contains little or no dependencies on its operating platforms. RRTP meets this criterion since its operation requires no changes to be made to the existing OSI software infrastructure. Furthermore, RRTP incurs minimal amount of dependencies on the UDP socket library. Such loose coupling makes RRTP readily portable to any OS that implements the standard socket subroutines.

The advantage of being highly deployable and maintainable is quite apparent: unlike other TCP variants, RRTP can be deployed in a hassle-free manner over any platform that supports UDP, i.e. no recompilation of any existing software is required prior to using RRTP. Furthermore, as long as the UDP API does not change, any future modification that needs to be made to the existing OSI software infrastructure will not affect the compatibility of RRTP. This is often not the case with many TCP variants that are closely coupled with the existing OSI software.

As it was described in section 2 of this thesis, TCP variants are commonly divided into three groups: (a) split-connection solutions as seen in I-TCP (Bakre and Badrinath [1995]); (b) link-layer solutions as seen in Snoop (Balakrishnan et al. [1995]); (c) end-to-end solutions with mechanisms for avoiding the invocation of congestion control in the presence of non-congestive packet losses, as seen in Explicit Congestion Notification (ECN, Floyd[1994]), Explicit Bad State Notification (EBSN, Bakshi et al.[1996]), Explicit Loss Notification (ELN, Balakrishnan and Katz[1998]), and Freeze-TCP (Goff et al.[2003]). Although all of the previously listed solutions deliver some degree of performance improvement, the practicality of their usages is severely limited by their relatively poor deployability. Solutions like ECN, EBSN and ELN require changes to be made to intermediate router nodes. As a result, they tend to be very costly to deploy considering the large amount changes needed to be made to the existing network infrastructure. Solutions in the same categories as I-TCP and Snoop will fail to work in the presence of network traffic encryption since the intermediate router nodes and base stations will be unable to distinguish between TCP and non-TCP traffic.

Aside from the aforementioned solutions, schemes like TCP HACK (Balan et al.[2001]) and Decoupled TCP (Wang and Kung[2001]) are genuine end-to-end approaches that do not require the participation of intermediaries in their flow/congestion control mechanisms. Nevertheless, such solutions involve both sender-side and receiver-side modifications. Consequently, they are not readily deployable over the existing network infrastructure. A further step towards good deployability is taken by sender-side-only modification protocols such as TCP Westwood, TCP SACK, and TCP Veno. Nevertheless, such solutions still require the recompilation of existing software before their performance benefits can be enjoyed over a network.

RRTP represents the final step in this evolution towards ideal deployability. Situated at the user level, RRTP brings about a true plug-and-play experience in the realm of TCP performance fine-tuning over both conventional and non-conventional network platforms.

5.5 Limitations

Despite the fact that a large amount of effort has been put into perfecting this piece of research, there are still a number of important limitations that the readers of this thesis should be aware of. The single most limiting factor on the breadth and depth of this research is resource, both in terms of time and research budget. Due to the fact that the research must be completed within a relatively short period of time in order to fulfill the Master's degree's requirements, the scope of this research is defined to explicitly exclude certain time consuming features (e.g. multi-homing) from being included as parts of the protocol. Some of these features will be briefly discussed in the final chapter of this thesis.

The amount of testing that has been carried out to empirically demonstrate the superior performance of RRTP is sufficient yet non-exhaustive. Given the available budget, there was enough financial resource to test RRTP on only one type of physical network. The wireless-last-hop network was chosen because it is the most widely used platform for which traditional TCP performs poorly on.

The total number of experiments that have been carried out using the physical implementation of RRTP was also limited by time and budget factors. Ideally, the larger the data sample we collect, the more confidence we will have in the conclusions that we draw based on the sample results. The reasoning behind the previous statement is: the dynamics of the protocol compounded by the fluctuating nature of the transmission medium result in a tremendously complex and large solution space. In order to properly characterize RRTP's behavior over a particular transmission medium, large amount of empirical testing data need to be captured for analysis.

Another limitation of this research arises from the fact that there are certain variables that cannot be controlled for when carrying out real-life experiments. For instance, in order to compare the performance of RRTP and TCP over a physical cellular network, two sets of identical tests must be carried out with one running on top of RRTP and the other one operating over TCP. These two sets of tests can be performed either in a sequential manner using the same client-server pair or in a simultaneous manner using two pairs of client and server machines having the same hardware specifications. If we choose to follow the first approach, we would have no way to ensure that the wireless link condition seen by RRTP is identical to that seen by TCP due to the temporal distance between the two sets of experiments. Alternatively, if we choose to follow the second approach, we would still be unable to ensure that the client machine running RRTP will be exposed to the exact same wireless link condition as the other client running TCP, no matter how physically close we place the two client machines to each other. This problem is known as the missing data problem in the social sciences. While there is no way to completely eliminate this problem, enlarging the size of the result sample tends to alleviate some of the uncertainties that the experimental methodology gives rise to.

Having now accounted for both the strengths and limitations of this piece of research, it is appropriate to bring this thesis to its conclusion. In the next chapter, the key research findings as well as a number of open areas for future research will be presented.

Chapter 6
Conclusions and Future Work

This final chapter summarizes the research findings presented in this thesis and proposes a number of possibilities for future extension work.

6.1 Summary of Research

This thesis presents a user-level, reliable and reconfigurable transport layer protocol that outperforms traditional TCP over non-conventional network platforms while still offering competitive performance over conventional, wired network platforms. RRTP addresses the weaknesses that TCP suffers from by employing a set of well-tuned, parameter-based, reconfigurable algorithms. Experiments carried out both in simulation and on real-life networks provide empirical support for the fact that RRTP enjoys superior performance over all major network platforms. From a practical stand point, RRTP possesses the desirable qualities of adaptability, deployability and maintainability, making it a suitable replacement for TCP.

The main contributions of the research can be summarized in the following two points:

- Identification of a set of user-configurable network parameters that can be pre-configured for tuning RRTP to quickly achieve close-to-optimal performance.
- Demonstration of a user-level transport layer protocol that outperforms TCP over a variety of network platforms, both in simulations and in real-life tests.

6.2 Future Work

As it was mentioned in section 5, the depth and the scope of this piece of research is limited by both time and financial factors. A number of open areas for future research are suggested in this section in order to provide the readers with a broader view of the issues involved in the creation of a replacement protocol for TCP. These areas can be divided into the following three categories: feature additions and optimizations for performance enhancement, test set expansions for more in-depth performance analysis, and performance tuning for achieving peripheral objectives.

In order to further improve the performance of RRTP, it is worthwhile to briefly consider several features that have not been examined thus far. Multi-homing is an effective technique for acquiring greater aggregate bandwidth through combining the resources from all available network interfaces. An application of this technique to TCP, named pTCP, was previously discussed in section 2.6.11 of this thesis. Multi-homing can be applied to RRTP in a similar fashion. Besides the readily seen advantage of greater aggregate bandwidth, additional benefits such as higher standard of QoS can be achieved by introducing the relevant algorithms into RRTP. For instance, RRTP can be modified to become traffic type/pattern aware. In the situation where RRTP has access to multiple active network interfaces, it can pre-categorize the expected network traffic flows and route them through the appropriate network channels: i.e., delay-sensitive traffic will enjoy a higher priority in accessing low-delay network channels.

The latency induced fairness issue has been a long-standing problem in transport layer protocol research. According to Lakshman et al.[1997], when traffic flows with different RTT's compete for the available bandwidth of a high latency network, flows with small RTT values will always dominate over the ones with larger RTT values. RRTP does not currently have the necessary provisions to overcome this problem. Consequently, further research is needed in the area of optimizing RRTP's bandwidth acquisition scheme when operating in multi-flow, high latency network environments. Perhaps it would be appropriate to introduce an algorithm that proportionally compensates for the latency induced over/under aggressive bandwidth acquisition behavior exhibited by RRTP traffic flows that share a common pipe. It is also worthwhile to note that one should thoroughly consider any possible ill-effect that such a scheme may have on RRTP's friendliness measures (modifications that leads to fair but unfriendly bandwidth acquisition behavior should be avoided).

Future optimizations of RRTP can be considered on a per platform basis. For instance, packet losses due to handoffs are commonplace over cellular wireless platforms. The performance impact that handoffs have on RRTP can be better studied if a detailed cellular base station coverage map is available for the test location. The results of such studies could give birth to a heuristic algorithm that promptly and reliably identifies handoffs for RRTP.

In addition to the three items mentioned above, the needs for new features as well as optimizations of existing features may arise along with the evolution of the network infrastructure that RRTP operates on. For instance, recent years' infrastructural upgrades to the cellular networks have brought about improvements in certain QoS parameters at the expense of certain other measures. In order to adapt to the changing network characteristics, it is necessary to continuously evolve RRTP's design and algorithms in a lockstep manner. This need for ongoing feature innovation can be readily seen in the case of TCP, for which a large body of research work has been dedicated to over the past thirty years.

The second area for future research can be described from a testing standpoint with three main criteria: quantity, variety, and scale. It is usually beneficial to increase the size of the test set both in term of quantity and variety. Thus far in this study, the majority of the testing resource has been devoted to measuring the performance of RRTP and TCP over physical cellular networks. If additional testing resource becomes available in the future, then further tests should be devised to study RRTP's performance on the other real-life network platforms as well as heterogeneous networks. Such tests are necessary for obtaining a more in-depth understanding of RRTP's performance characteristics as well as for identifying opportunities for algorithmic improvements.

Testing scale is also crucial for mission-critical software like RRTP. In order to properly assess the viability of using RRTP as a replacement protocol for TCP over a large-scaled network such as the Internet, lengthy stress tests involving the deployment of RRTP over a network with sufficient complexity and heterogeneity need to be carried out. However, such tests may only be feasible to realize in simulations since the cost of performing them over real-life networks in a controlled fashion can be prohibitively high.

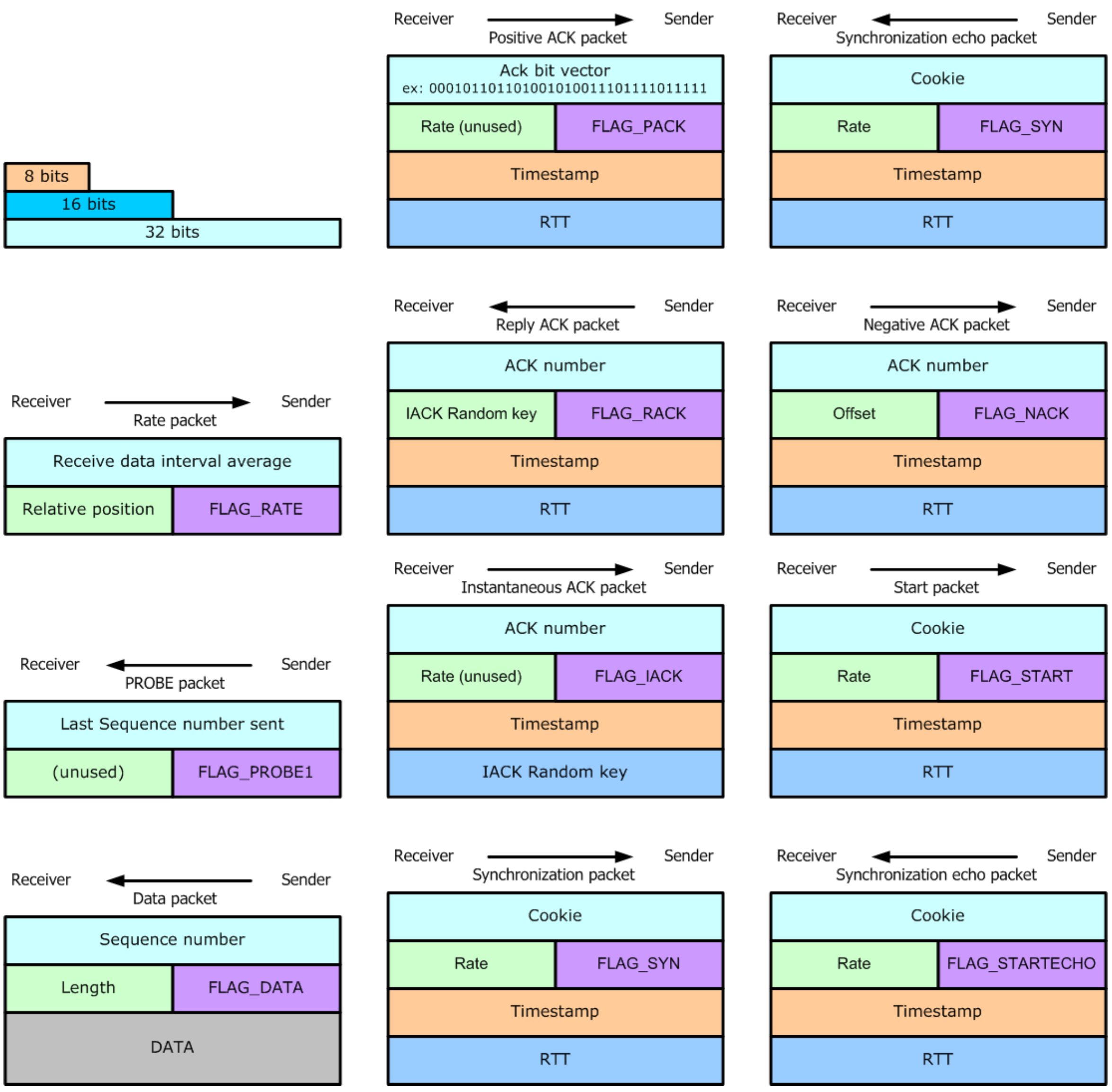
Greater test variety can also be achieved through coupling RRTP with different types of network applications. These applications may differ significantly when it comes to their utilization behaviour of the underlying reliable transport protocols. Some applications need lengthy, persistent connections while others are characterized by short-lived traffic flows. In this piece of research, RRTP has been demonstrated to work well for file transfer applications. Additional insights can be gained by carrying out performance analyses of alternate types of network applications such as email or web browser when deployed on top of RRTP.

As it was mentioned previously, the third area for future research comes from optimization considerations surrounding auxiliary design objectives. In the context of mobile computing, issues surrounding power consumption and battery life should be examined. From intuition, since RRTP reduces the total number of lost packets and thereby eliminated the need for frequent packet retransmission due to wireless medium unreliability, it is expected to be a greener protocol than TCP in high loss network environments. However, one could also argue that the added algorithmic complexity of RRTP may incur higher computational cost than TCP in certain scenarios and thereby making RRTP less energy efficient. Consequently, it is necessary to gather conclusive evidences by carrying out further experiments that profile the actual power consumption of RRTP as compared to that of TCP. Any energy inefficiencies revealed by this type of experiments can be subsequently addressed through algorithmic enhancements.

Generally speaking, the work that has been presented in this thesis is in its early stage of development and there is still plenty of room for future improvements. It would not be an overstatement to claim that the direction that RRTP has taken will likely lead to a very promising path to creating a suitable replacement protocol for TCP.

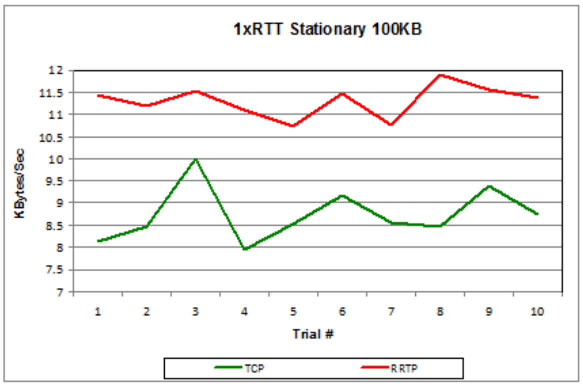
Appendix A

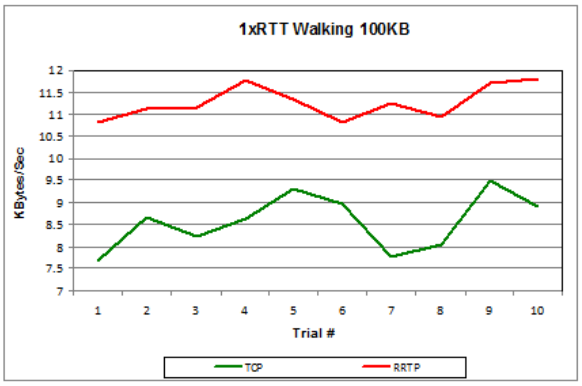
RRTP Packets

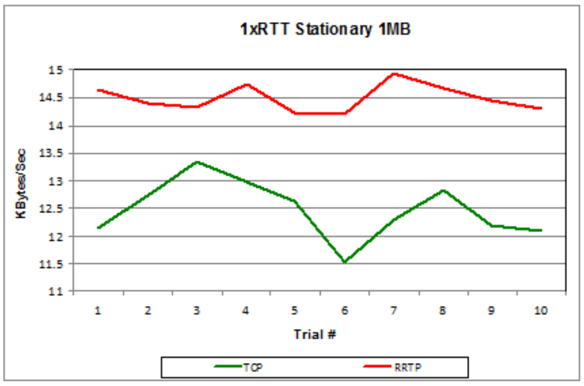


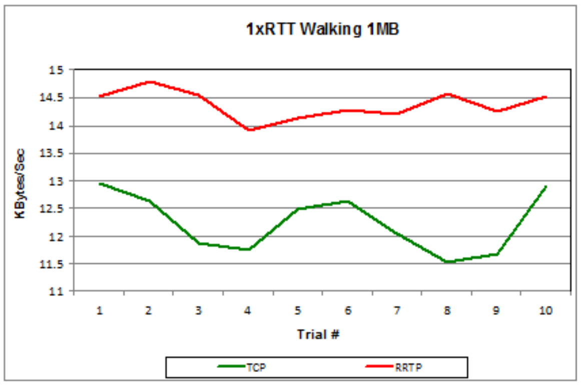
Appendix B

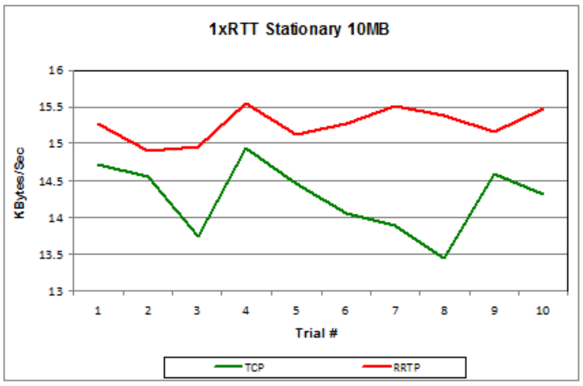
Real-life CDMA Network Experimental Results

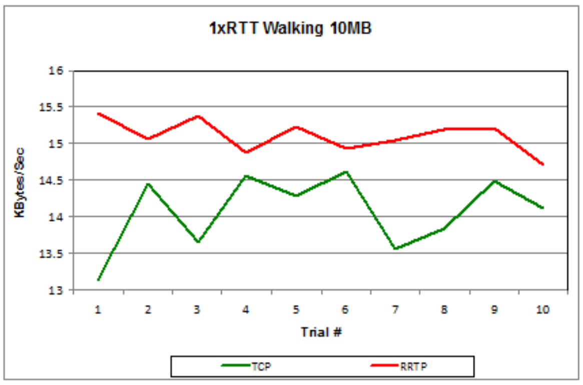


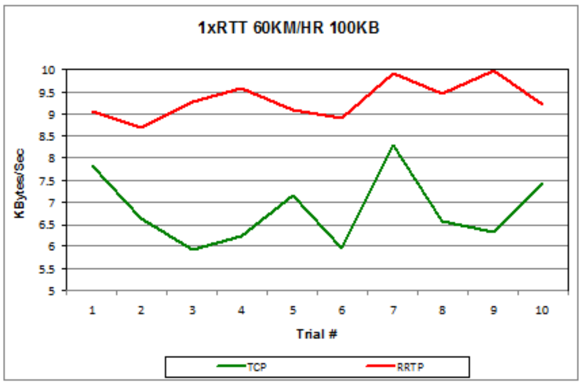


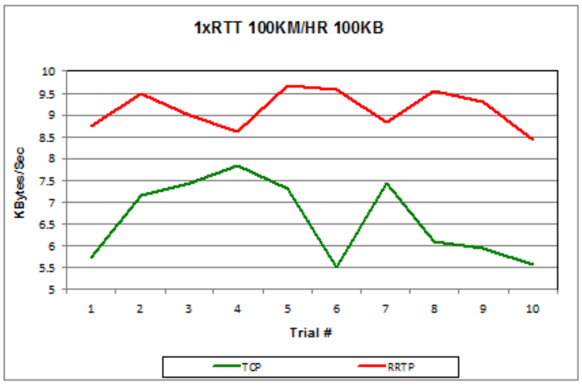


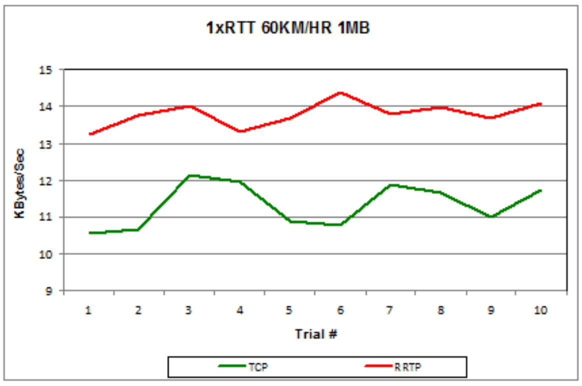


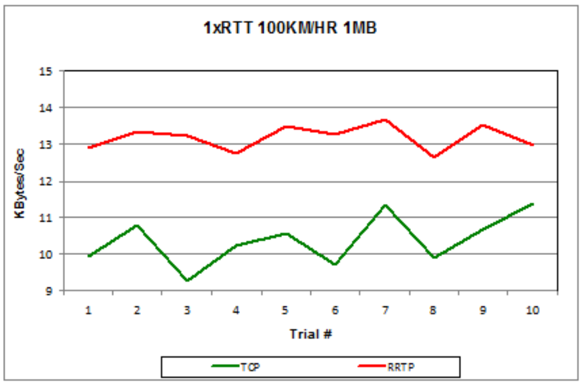


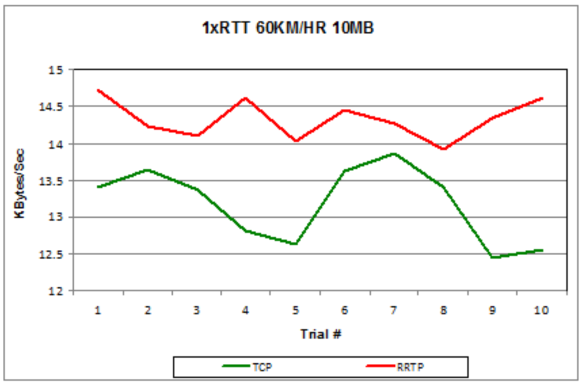


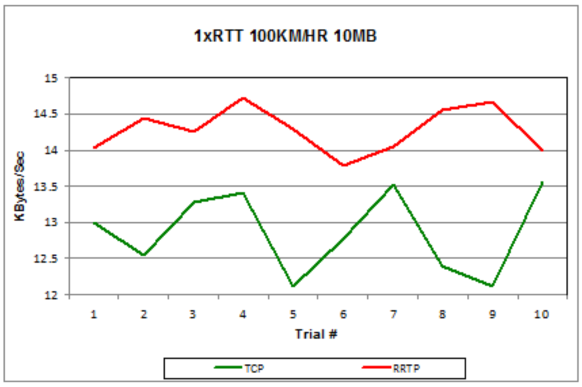

1xRTT 60KM/HR 100KB
KBytes/Sec
Trial #
TCP
RRTP


1xRTT 100KM/HR 100KB
KBytes/Sec
Trial #
TCP
RRTP


1xRTT 60KM/HR 1MB
KBytes/Sec
Trial #
TCP
RRTP

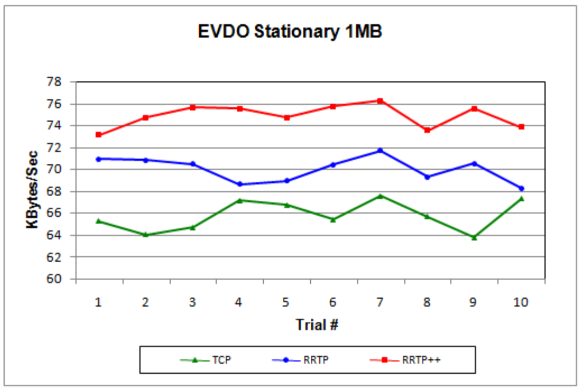

1xRTT 100KM/HR 1MB
KBytes/Sec
Trial #
TCP
RRTP

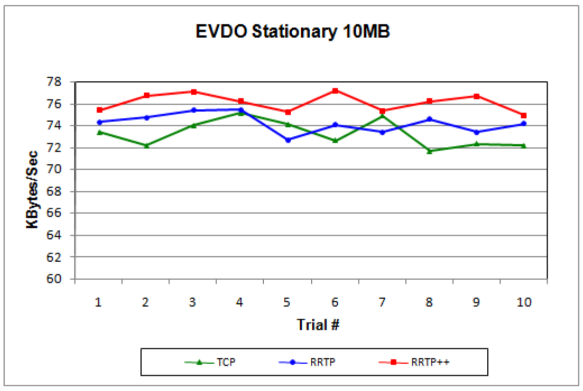

1xRTT 60KM/HR 10MB
KBytes/Sec
Trial #
TCP
RRTP

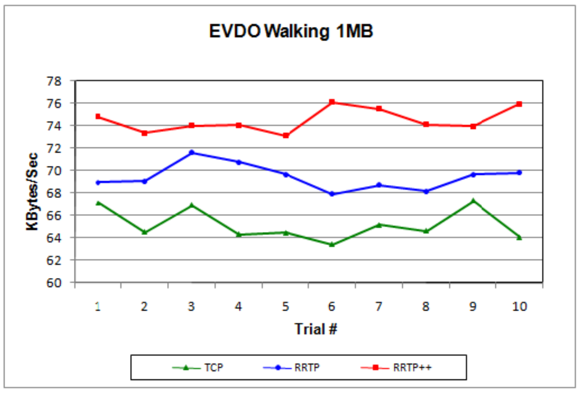

1xRTT 100KM/HR 10MB
KBytes/Sec
Trial #
TCP
RRTP

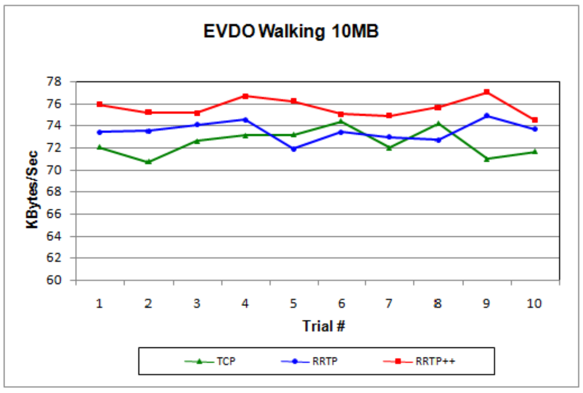
Appendix C

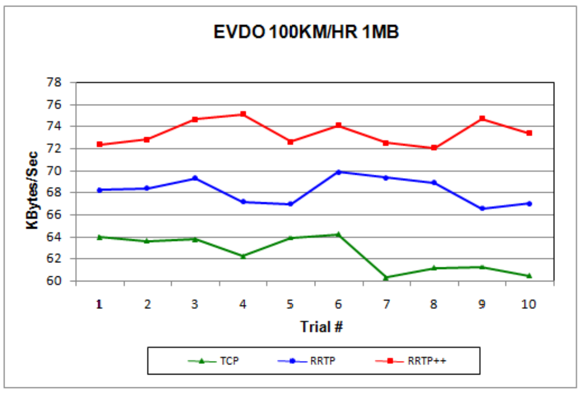
Real-life EVDO Network Experimental Results

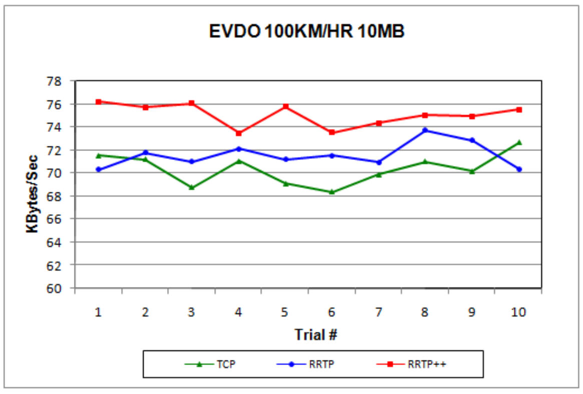






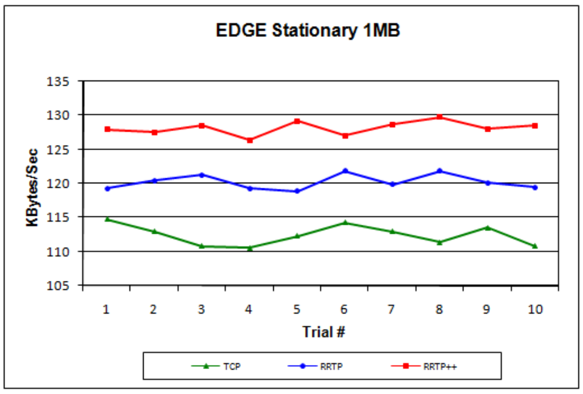


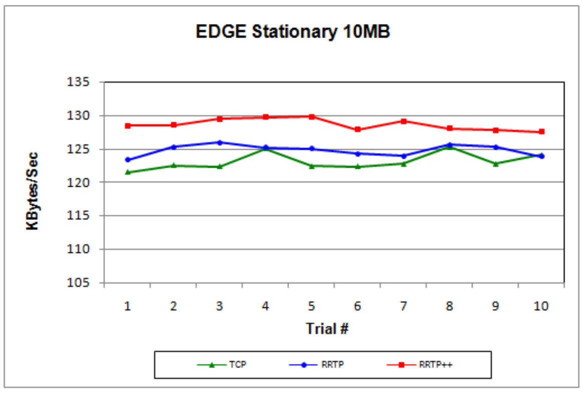


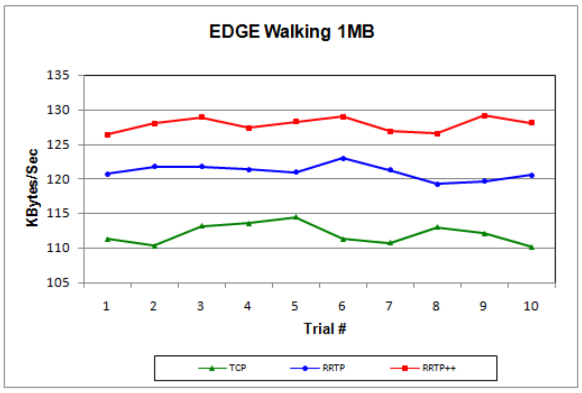


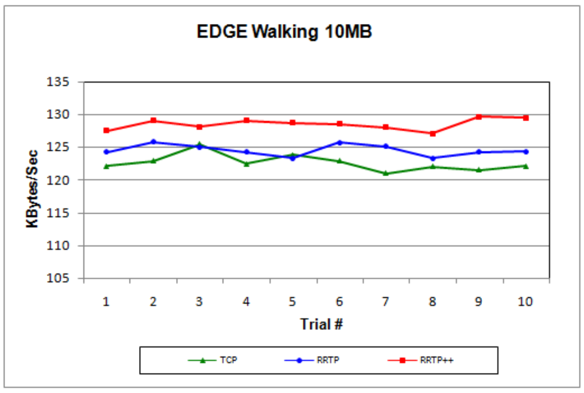
Appendix D

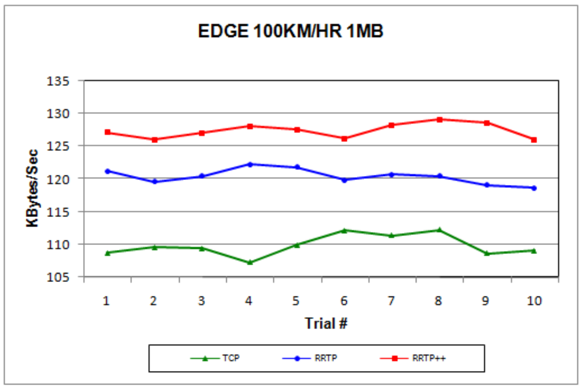
Real-life EDGE Network Experimental Results

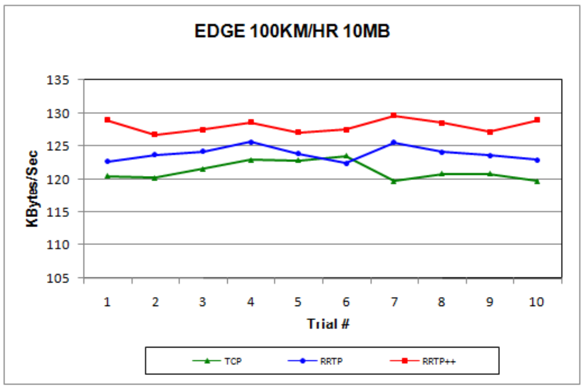












Appendix E
List of Abbreviations

ACK	Acknowledgment
API	Application Programming Interface
ARQ	Automatic Repeat Request
BER	Bit Error Rate
BSC	Base Station Controller
BTS	Base Transreceiver Station
CACK	Cumulative Acknowledgment
CDMA	Code Division Multiple Access
CPU	Central Processing Unit
CWND	Congestion Window
FEC	Forward Error Correction
GSM	Global System for Mobile communications
IP	Internet Protocol
IPSec	Internet Protocol Security
LAN	Local Area Network
LDA	Loss Discrimination Algorithm
LFP	Long Fat Pipe
LIMD	Linear Increase Multiplicative Decrease

MAC	Media Access Control
NAK	Negative Acknowledgment
OS	Operating System
OSI	Open System Interconnect
PCF	Point Coordination Function
PDSN	Packet Data Serving Node
QoS	Quality of Service
RAM	Random Access Memory
RFC	Request For Comments
ROTT	Relative One-way Trip Time
RRTP	Reliable and Reconfigurable Transport Protocol
RTT:	Round Trip Time
SACK	Selective Acknowledgment
SSTHRESH	Slow Start Threshold
TCB	Transmission Control Blocks
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WTCP	Wireless Transmission Control Protocol

Bibliography

[1] Akyildiz, I. F., Zhang, X. and Fang, J. TCP-Peach: Enhancement of TCP-Peach for Satellite IP Networks. IEEE Communications Letters 6, no. 7, pp. 303–305, Jul. 2002.

[2] Allman, M., Paxson, V., and Stevens, W. TCP Congestion Control. RFC 2581, Apr. 1999.

[3] Ayanoglu, E., Paul, S., Laporta, T. F., Sabnani, K., and Gitlin, R. AIRMAIL: A link–layer protocol for wireless networks. ACM Wireless Networks 1, pp. 47–60, Feb. 1995.

[4] Bakre, A. and Bardinath, B. R. I-TCP: Indirect TCP for Mobile Hosts. Proceedings of ICDCS 95, pp. 136-143, May 1995.

[5] Bakre, A. and Bardinath, B. R. Implementation and Performance Evaluation of Indirect TCP. IEEE Transactions on Networking, vol. 46, no. 3, pp. 260–278, Mar. 1997.

[6] Bakshi, B.S., Krishna, P., Vaidya, N.H., and Pradhan, D.K. Improving Performance of TCP over Wireless Networks. Technical Report 96-014, Texas A&M University, 1996.

[7] Balakrishnan, H., Seshan, S., and Katz, R. Improving reliable transport and handoff performance in cellular wireless networks. Wireless Networks 1, 4, pp.469–481, 1995.

[8] Balakrishnan, H., Seshan, S., and Katz, R. A comparison of mechanisms for improving TCP performance over wireless links. ACM SIGCOMM, Palo Alto, CA, pp. 256-269, Aug. 1996.

[9] Balakrishnan, H., Padmanabhan, V., Seshan, S., and Katz, R. A comparison of mechanisms for improving TCP performance over wireless links, in IEEE/ACM Transactions on Networking, vol.5, no.6, pp. 756-69, Dec. 1997.

[10] Balakrishnan, H. and Katz, R. Explicit Loss Notification and Wireless Web Performance. Proceedings of IEEE Globecom Internet Mini-Conference, Sydney, Australia, Nov. 1998.

[11] Balan, R.K., Lee, B.P., Kumar, R.R., Jacob, L., Seah, W.K.G., and Ananda, A.L. TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links. Proceedings of IEEE INFOCOMM, vol. 1, pp. 309-318, 2001.

[12] Biaz, S. and Vaidya, N., “Distinguishing congestion losses from wireless transmission losses: A negative result,” in Proceedings of 7th International Conference on Computer Communications and Networks, Lafayette, LA, Oct. 1998.

[13] Brakmo, L., and Peterson, L. TCP Vegas: End to End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas in Communication 13, no. 8, pp. 1465–1480, Oct. 1995.

[14] Braden, R. Requirements for Internet Hosts – Communication Layers. RFC 1122, Oct. 1989.

[15] Brown, K. and Singh, S. M-TCP: TCP for mobile cellular networks. Computer Communication Review, pp. 19–43, Jul. 1997.

[16] Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M. Y., and Wang, R. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. Paper presented at the ACM Conference on Mobile Computing and Networking (MOBICOM), Rome, Italy, pp. 287–297, Jul. 2001.

[17] Chiu, D. and Jain, R., Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks, Journal of Computer Networks and ISDN Systems, vol. 17, no. 1, Jun. 1989.

[18] Elaarag, H. and Bassiouni, M. Performance evaluation of TCP connections in ideal and non-ideal network environments. Computer Communications Journal, 24, 18, pp. 1769–1779, 2001.

[19] Fall, K. and Floyd, S. Simulation based comparisons of Tahoe, Reno, and SACK TCP. ACM Computer Communication Review 26, 3, pp. 5–21, 1996.

[20] Floyd, S. Congestion Control Principles. RFC 2914, Sep. 2000.

[21] Floyd, S. TCP and Explicit Congestion Notification. ACM Computer Communication Review, vol. 24, no. 5, Oct. 1994.

[22] Floyd, S., Mahdavi, J., Mathis, M., and Podolsky, M. An extension to the selective acknowledgment (SACK) option for TCP. RFC 2883, Jul. 2000.

[23] Fu, C. P. and Liew, S. C. TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks. IEEE Journal on Selected Areas in Communications 21, no. 2, pp.216–228, 2003.

[24] Goff, T., Moronski, J., Phatak, D.S., and Gupta, V. Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments. IEEE INFOCOM, Tel-Aviv, Mar. 2003.

[25] Hoe, J. C. Improving the start-up behavior of a congestion control scheme for TCP. ACM SIGCOMM, pp. 270–280, 1996.

[26] Hsieh, H.Y. and Sivakumar, R. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. Proceedings of the 8th annual international conference on mobile computing and networking, Atlanta, Georgia, Sep. 2002.

[27] Jacobson, V. Congestion avoidance and control. ACM SIGCOMM, 1988.

[28] Jacobson, V., Braden, R., and Borman, D. TCP Extensions for High Performance. RFC 1323, May 1992.

[29] Jain, R., Chiu, D., and Hawe, W. "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC, Res. Rep.TR-301, 1984.

[30] Lakshman, T.V., Madhow, U., and Suter, B. Window-based Error Recovery and Flow Control with a Slow Acknowledgment Channel: a Study of TCP/IP Performance. IEEE INFOCOM, 1997.

[31] Ludwig, R., and R. H. Katz. The Eifel Algorithm: Making TCP Robust against Spurious Re-Transmission. ACM Computer Communication Review 30, no. 1, pp. 30–36, Jan. 2000.

[32] Ludwig, R., and Meyer., M. The Eiffel Detection Algorithm for TCP. Internet Draft, IETF, Feb. 2002.

[33] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. TCP Selective Acknowledgment Options. RFC 2018, Oct. 1996.

[34] Nanda, S., Ejzak, R., and Doshi, B. A retransmission scheme for circuit-mode data on wireless links. IEEE Journal on Selected Areas in Communications 12, 8, pp. 1338–1352, 1994.

[35] Ong, L. and Yoakum, J. An Introduction to the Stream Control Transmission Protocol, RFC 3286, May 2002.

[36] Parsa, C., and Garcia-Luna-Aceves, J.J. TULIP: A Link-Level Protocol for Improving TCP over Wireless Links. In Proc. IEEE Wireless Communications and Networking Conference, New Orleans, Louisiana, vol. 3, Sep. 1999.

[37] Parsa, C., and J. J. Garcia-Luna-Aceves. Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media. Paper presented at the 7th Annual IEEE International Conference on Network Protocols, Toronto, Canada, pp. 213–221, Nov. 1999.

[38] Postel, J. Transmission Control Protocol. RFC 793, Sep. 1981.

[39] Ramakrishnan, K.K. and Floyd, S. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, Jan. 1999.

[40] Sinha, P., Nandagopal, T., Venkitaraman, N., Sivakumar, R., and Bharghavan, V. WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. ACM/Baltzer Wireless Networks Journal 8, no. 2–3, pp. 301–316, Mar. 2002.

[41] Stevens, W. R. and G. R. Wright (1994). TCP/IP illustrated. Boston, Addison-Wesley.

[42] Tobe, Y., Tamura, Y., Molano, A., Ghosh, S., and Tokuda, H. Achieving moderate fairness for UDP flows by path-status classification, in Proc. 25th Annual IEEE Conf. on Local Computer Networks, Tampa, FL, pp. 252–261, 2000.

[43] Wang, S. Y. and Kung, H. T. Use of TCP Decoupling in Improving TCP Performance over Wireless Networks. ACM Wireless Networks, vol. 7, no. 3, pp. 221-236, May 2001.

[44] Wang, K. Y. and Tripathi, S. K. Mobile-End Transport Protocol: An Alternative to TCP/IP over Wireless Links. In Proceedings of IEEE Conference on Computer Communication, San Francisco, California, vol. 3, pp. 1046–1053, Mar. 1998.

[45] Yavatkar, R. and Bhagawat, N. Improving end-to-end performance of TCP over mobile internetworks. Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, pp. 146–152, 1995.