Transport Protocols for Next Generation Wireless Data Networks

A Thesis
Presented to
The Academic Faculty

by

Aravind Velayutham

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

School of Electrical and Computer Engineering
Georgia Institute of Technology
April 2005

Transport Protocols for Next Generation Wireless Data Networks

Approved by:

Raghupathy Sivakumar, Advisor

Henry Owen

Chuanyi Ji

Date Approved

TABLE OF CONTENTS

LIST OF TABLES . . . . . **vi**

LIST OF FIGURES . . . . . **vii**

SUMMARY . . . . . **viii**

I INTRODUCTION **1**

1.1 Heterogeneity of wireless access interfaces 1

1.2 Peer-to-peer communication in wireless data networks 3

II TRANSPORT ADAPTATION : PROBLEM DEFINITION **6**

2.1 Transport Adaptation 6

2.2 What is the required nature of ideal transport adaptation? 6

2.3 What is the ideal granularity of transport layer adaptation? 8

2.4 Evaluation Model 9

2.4.1 Experimental Network Topology 9

2.4.2 Network environment parameters 9

2.4.3 Traffic flows 11

2.5 Evaluation of Transport Mechanisms 11

2.5.1 Data Transmission 12

2.5.2 Congestion Detection 17

2.5.3 Data Acknowledgment 21

2.5.4 Suffix-loss Recovery 24

2.5.5 Start-up Capacity Determination 26

2.6 Summary of Results 28

III *TP : DESIGN AND IMPLEMENTATION **29**

3.1 Design Goals 29

3.1.1 Reconfigurability 29

3.1.2 Extensibility 30

3.1.3 Minimal Overheads 30

3.2 Design Elements 31

3.2.1 Triggers 31

3.2.2 Separation of Core and non-Core Modules . . . 31
3.2.3 Modular Architecture and Execution Model . . . 32
3.2.4 State Propagation . . . 32
3.2.5 Mobile-host Centric Operations . . . 33
3.3 Software Architecture . . . 33
3.3.1 Interfaces with the Application and IP . . . 35
3.3.2 Global Data Structures . . . 35
3.3.3 Transport Engine . . . 35
3.3.4 Reconfiguration Entities . . . 36
3.4 Transport Execution . . . 36
3.5 Trigger Management . . . 37
3.6 Adaptation Process and State Propagation . . . 38
IV *TP : PERFORMANCE EVALUATION . . . 40
4.1 Lossy WLANs . . . 40
4.2 Bandwidth Fluctuation in WWANs . . . 41
4.3 Jitter in Satellite networks . . . 41
V CONVERSATIONAL TRANSPORT PROTOCOLS : PROBLEM DEFINITION . . . 43
5.1 End-to-end congestion control . . . 43
5.2 Route changes . . . 45
5.3 Connection progress . . . 46
5.4 Reverse path traffic . . . 46
5.5 Impact on MAC performance . . . 47
5.6 Impact on routing protocol design . . . 48
VI NON-CONVERSATIONAL TRANSPORT PROTOCOL (NCTP) : DESIGN AND IMPLEMENTATION . . . 49
6.1 Overview . . . 49
6.2 NCTP vis-a-vis Drawbacks of Conversational Protocols . . . 50
6.3 Challenges . . . 52
6.4 Mechanisms . . . 53
6.4.1 Hop-by-hop Congestion control . . . 53

6.4.2 Coarse-level end-to-end acknowledgments . . . 54
6.4.3 Bundle ownership transfer . . . 54
6.4.4 Splitting files into bundles . . . 55
6.4.5 Route Probing . . . 55
6.4.6 Single hop reliability . . . 56
6.5 Packet Headers and Protocol Operations . . . 57
6.5.1 Source . . . 58
6.5.2 Intermediate-node . . . 58
6.5.3 Destination . . . 59
6.5.4 Bundle transmission procedure . . . 59
6.6 Properties . . . 60
6.6.1 Throughput capacity . . . 60
6.6.2 Fairness . . . 64
VII NCTP : PERFORMANCE EVALUATION . . . 66
7.1 Environment . . . 66
7.2 Macroscopic Results . . . 67
7.2.1 Impact of network size . . . 67
7.2.2 Impact of Load . . . 68
7.2.3 Impact of mobility . . . 68
7.3 Microscopic results . . . 69
7.3.1 Connection Progress . . . 69
7.3.2 Spatial re-use . . . 70
7.3.3 Bundle-Size . . . 71
VIII CONCLUDING REMARKS . . . 72
REFERENCES . . . 73

LIST OF TABLES

1 Wireless network characteristics 11

2 Notations used for Analysis 60

LIST OF FIGURES

1 Network Topology for Transport Protocol Evaluation 10
2 Impact of loss rate on data transmission mechanisms 13
3 Impact of bandwidth fluctuation on data transmission mechanisms 14
4 Impact of loss rate on congestion detection mechanisms 18
5 Impact of loss rate on enhancements to loss-based congestion detection . . . 19
6 Impact of jitter on congestion detection mechanisms 20
7 Impact of loss on data acknowledgment mechanisms 22
8 Impact of bandwidth fluctuation on data acknowledgment mechanisms . . . 23
9 Impact of loss rate on suffix loss recovery mechanisms 26
10 Impact of network BDP on start-up bandwidth probing mechanisms 27
11 *TP Framework 34
12 Performance study of the *TP framework 40
13 Identifying shortcomings of conversational transport protocols 44
14 Analyzing the performance of conversational transport protocols 45
15 MAC Utilization of conversational transport protocols 48
16 NCTP Source Operations 56
17 NCTP Packet Header Format 57
18 NCTP Bundle Transmission across a link 57
19 NCTP Intermediate node operations 59
20 NCTP Destination node operations 60
21 Throughput results - I 66
22 Throughput results - II 67
23 Fairness results - I 68
24 Fairness results - II 69
25 Connection Progress 70
26 Spatial Reuse 70
27 Burst Size 71

SUMMARY

Emerging wireless networks are characterized by increased heterogeneity in wireless access technologies as well as increased peer-to-peer communication among wireless hosts. The heterogeneity among wireless access interfaces mainly exists because of the fact that different wireless technologies deliver different performance trade-offs. Further more and more infrastructure-less wireless networks such as ad-hoc networks are emerging to address several application scenarios including military and disaster recovery. These infrastructure-less wireless networks are characterized by the peer-to-peer communication model. In this thesis, we propose transport protocols that tackle the challenges that arise due to the above-mentioned properties of state-of-the-art wireless data networks.

The main contributions of this work are as follows:

1. We determine the ideal nature and granularity of transport adaptation for efficient operation heterogeneous wireless data networks by performing comprehensive experimental analysis. We then design and implement a runtime configurable adaptive transport framework, *TP which accommodates the capabilities of the ideal transport adaptation solution.

2. We prove that conversational transport protocols are not efficient under peer-to-peer wireless data networks. We then design and implement NCTP which is a non-conversational transport protocol.

CHAPTER I

INTRODUCTION

Wireless data networks have emerged as a convenient and viable alternative to wired networks in several network scenarios such as access networks. Wireless data networks have been a popular research topic for more than two decades now. Wireless network research includes designing better protocols at all layers of the protocol stack, implementing better network architecture for wireless networks and tackling mobility of wireless clients.

Two major characteristics are evident in emerging wireless network technologies namely:

1. Increased heterogeneity of wireless access technologies

2. Increased peer-to-peer communication among wireless clients

In this work, we characterize the issues that arise due to the above mentioned properties of wireless data networks with respect to designing efficient transport protocols. We also propose novel transport protocols for addressing the issues and evaluate the transport protocols' performance.

1.1 Heterogeneity of wireless access interfaces

Wireless access technologies today exhibit a great deal of heterogeneity. A mobile user can choose from a myriad of options for wireless access, such as satellite wireless data networks (e.g. DirecTV data), wide area wireless data networks or WWANs (e.g. Verizon 3G data), local area wireless data networks or WLANs (e.g. IEEE 802.11a/b/g), personal area wireless networks or WPANs (e.g. Bluetooth), etc., with each class of networks further being realized through several different technologies. Some of the heterogeneity can indeed be attributed to efforts by independent entities to build solutions. However, the heterogeneity *also* exists because heterogeneous technologies deliver different performance trade-offs. For example, while wide-area data networks can, to a large degree, enable ubiquitous wireless coverage,

they offer several orders of magnitude lower bandwidths than local-area wireless networks. Thus, it is reasonable to expect that mobile devices will increasingly be equipped with multiple interfaces to provide users with access to the best possible technology in any given environment. Off-the-shelf laptops and PDAs that support multiple wireless interfaces can in-fact be obtained today.

It is well established that appropriately designed wireless transport protocols can substantially improve the performance of mobile users [3, 21, 10] across different wireless network technologies. Most of such protocols are designed specifically to address characteristics of the wireless environment where they are used. For example, TCP-ELN uses explicit loss notification to aid its congestion control in lossy networks [3]. WTCP [21] was proposed to address the challenging characteristics of the wireless wide-area networks such as low and variable bandwidth and high and variable delay. Similarly, STP addresses the limited reverse path bandwidth problem by aggregating feedback messages from the receiver [10]. Such tailored protocols are necessary because of the *defining* role played by the wireless link on the end-to-end path characteristics. Irrespective of the path characteristic of interest – bandwidth, jitter, loss, fluctuation, etc. – the wireless link more often than not defines it. Hence protocols, even if designed as end-to-end solutions, considerably improve upon the performance when they explicitly tackle the unique characteristics of the wireless environment *through appropriate transport layer mechanisms.*

Transport adaptation is a significant problem with regard to achieving efficient end-system support for heterogeneous wireless data networks. We define *transport adaptation* as the behavior of the transport protocol, when the mobile host moves across different wireless networks, with the goal of achieving the best performance. Although numerous works have looked at transport layer design for heterogeneous wireless networks, there is no single work which performs an explicit study of the requirements of ideal transport layer adaptation.

In this work, we answer the following questions with regard to the properties of an ideal transport adaptation solution.

- *What should be the ideal nature of transport adaptation?*

Should the adaptation involve changing entire transport protocols at a time, or changing transport mechanisms as required, or can it simply involve change only in protocol parameters?

- *At what granularity should the transport adaptation be performed?*

Should the adaptation be done only when there is a handoff between network interfaces or can it be required even when network conditions change within the same wireless network?

We perform extensive test-bed experiments to evaluate the performance of transport mechanisms in different wireless environments and conclude the following : Ideal transport adaptation should accommodate transport mechanism changes. Neither transport protocol nor protocol parameter change is sufficient enough for optimal performance across heterogeneous wireless networks. Further transport adaptation has to be performed at a granularity finer than interface handoffs. Ideal transport adaptation should change mechanisms even when the network characteristics change within the same wireless network.

Using the insights gained through the performance evaluation, we design and implement a runtime adaptive transport layer framework called *TP (Adaptive Mobile Transport Protocol). *TP is a transport layer solution that accommodates the ideal nature and granularity of transport layer adaptation problem. *TP provides the capability to reconfigure the transport layer behavior, while minimizing the impact of such transformations on applications, and hiding it completely in the best case. Briefly, *TP provides a clear separation in the realization of core and non-core transport functionalities, is fully modular, employs an event-driven execution model, and allows for effective state propagation between different avatars of the transport protocol as it transforms.

1.2 Peer-to-peer communication in wireless data networks

Apart from the heterogeneity among the various wireless access interfaces, wireless data networks are more and more characterized by peer-to-peer communication. The peer-to-peer communication model is prevalent in wireless data networks due to several reasons. Several applications of wireless data networks such as military and disaster-recovery impose the need for infrastructure-less communication among wireless hosts. Such wireless networks are

popularly called wireless ad-hoc networks. Further, there has been an increasing deployment of wireless devices which sense the environment intelligently. These devices communicate among themselves to operate as a single monitoring entity. Such networks called as wireless sensor networks are also characterized by the peer-to-peer communication model. Whereas the above mentioned reasons attribute the need for peer-to-peer communication for serving specific applications, there has also been attempts to improve the performance of cellular wireless data networks by incorporating the peer-to-peer communication model [12].

Numerous transport protocols have been proposed in related literature over the last decade or so [11], [14], [23], [13], [6] for peer-to-peer wireless networks. Most of such protocols are variations of TCP - the most dominantly used transport protocol in the Internet, tailored to handle some of the defining aspects of the target environment. At the same time, protocols that are "built-from-scratch" such as the Ad-hoc Transport Protocol (ATP [23]) have been proposed exploiting the fact that in many applications of ad-hoc networks, the networks are stand-alone and hence compatibility with TCP is not a requirement.

While the protocols identified above differ considerably in the specific mechanisms they use, they predominantly fall under the class of *end-to-end conversational protocols*. In the rest of this dissertation, we refer to such protocols as simply *conversational protocols*. In other words, the designs of the protocols are predicated on the *seats of intelligence and data being shared solely by the source and the destination*, and *there being fine-grained communication exchange between the source and the destination* as a connection progresses. For example, considering default TCP, the bulk of the transport layer intelligence in terms of congestion control, reliability, flow-control, and connection management, resides at the source. At the same time, the receiver sends back positive cumulative acknowledgments back to the sender for every b data packets received. b is typically 2 for most implementations of TCP.

In this work, we argue that such a conversational paradigm is fundamentally inappropriate for mobile wireless ad-hoc networks. We present arguments in terms of how the conversational nature of protocols can result in imprecise congestion control, under-utilization of resources due to frequent path transients, poor connection progress, and high reverse

path overheads. Perhaps equally importantly, we also show that the use of conversational protocols at the transport layer also results in inefficiency at the routing and MAC layers.

We then consider an anti-thesis of a traditional conversational protocol design, one in which the seats of intelligence and data are always on either side of a single wireless hop, and move from one hop to another on the path from the source to the destination as the connection progresses. Thus, the conversation is always restricted to exactly a single hop, and there is no fine-grained communication on an end-to-end basis. We refer to a protocol with the above design as being *non-conversational*. We show that such a design effectively alleviates the drawbacks identified for conversational protocols. Perhaps more surprisingly, we establish that such a protocol design does not have any obvious pitfalls one would expect in terms of spatial re-use, throughput capacity, and fairness.

Finally, we present a new transport layer protocol called NCTP (Non-Conversational Transport Protocol) that is based on the non-conversational protocol design, but is equipped with mechanisms to address several practical challenges that have to be tackled in order for the non-conversational design to be effectively realized. We show through both analysis and packet level *ns2* based simulations that NCTP performs considerably better than existing transport protocols. Specifically, we compare the performance of NCTP against default TCP and a network-assisted rate-based transport protocol - ATP.

CHAPTER II

TRANSPORT ADAPTATION : PROBLEM DEFINITION

2.1 Transport Adaptation

We define *transport adaptation* as the behavior of the transport protocol, when the mobile host moves across different wireless networks, with the goal of achieving the best performance. We present the issues with respect to achieving ideal transport adaptation. The solutions to these problems, presented in the next chapter, apart from helping us understand the requirements of ideal transport layer adaptation, would facilitate the design and implementation of an efficient transport layer framework for mobile hosts in heterogeneous wireless data networks.

2.2 What is the required nature of ideal transport adaptation?

Transport adaptation, in essence, refers to the change of some element of the transport protocol. The first problem focuses on the ideal nature of adaptation : In order to achieve optimal transport protocol performance across different wireless environments what should be changed within the transport protocol? A transport protocol can be viewed at different levels of complexity and these levels dictate the choices available for transport layer adaptation.

The coarsest level of detail is the entire transport protocol consisting of all the mechanisms used to implement the different functionalities. We describe transport functionalities and mechanisms in detail in the next section. Briefly, transport functionalities are the mandatory modules that any transport protocol should implement and each functionality can be implemented through one of several alternative transport mechanisms. Change at the protocol level would require the replacement of one transport protocol (say TCP-ELN) by another (say WTCP) for optimal performance.

At a finer granularity than protocol change, is change of one or more mechanisms used by transport protocols. Examples of transport mechanisms include loss-based congestion detection, rate-based congestion control, self-clocked data transmission and timeout-based loss recovery. This level of adaptation would require the change of one mechanism, say loss-based congestion detection, by an alternative mechanism such as delay-based congestion detection. We note that this level of adaptation is the most complex because it involves the co-existence of multiple previously unrelated modules.

The finest level of detail of transport protocols are the parameter values used by the transport mechanisms. Protocol parameters include increase and decrease values in AIMD congestion control mechanism, the number of SACK blocks used by the SACK acknowledgment scheme, the guard values used by the delay-based congestion control, the number of DUPACKs for loss detection, and the number of DATA packets acknowledged by a single ACK packet. Adaptation at this level refers to the change of protocol parameter values in order to improve the transport protocol operations. Although protocol parameter change is the finest level of detail within the transport protocol, it is also the easiest level of adaptation wherein the values of variables are changed within the same transport protocol.

Most transport protocol developers implicitly assume the need for protocol changes when designing alternate protocols (such as WTCP) to the de-facto standard transport protocol TCP better suited for specific wireless environments. But several of these supposedly "alternate" protocols retain some of the mechanisms used by TCP. For example the SACK mechanism used by TCP for acknowledging data has been borrowed by most of the newly designed protocols. Hence some of these "new" protocols can be viewed as a set of mechanism changes to TCP. There are several other works that have proposed enhancements which are essentially add-on mechanisms to existing mechanisms to achieve better performance. We assume that these do not constitute "transport layer adaptation" since they are simply static enhancements and are not changed during the operation of the transport protocol. One example of such an enhancement is the addition of Explicit Loss Notification (ELN) [3] feature to TCP to assist its loss classification mechanism. TCP-ATL [1], an adaptive transport layer protocol assumes that adapting the additive and decrease

parameter values used by the AIMD congestion control mechanism of TCP would achieve better performance across different wireless environments. This belongs to the third choice of transport layer adaptation that we discussed which is by protocol parameter change. The motivation for the above mentioned choices of transport layer adaptation will be established in subsequent sections. No previous work has investigated the ideal nature of transport adaptation for achieving optimal performance across heterogeneous wireless networks. In this work we answer this problem by evaluating the performance of a set of competing transport mechanisms in different wireless network environments.

2.3 What is the ideal granularity of transport layer adaptation?

The previous problem addressed the issue of "what" element of the transport protocol should be changed by an ideal adaptation framework. Now we ask the question : "When should the element change be performed by the adaptation framework?"

The coarsest level of granularity for transport layer adaptation is across transport layer sessions. This level of adaptation is triggered by application requirements rather than wireless networks. Some transport layer frameworks such as the Universal Transport Library (UTL) [5] and Configurable Transport Protocol (CTP) [24] adopt this model of transport adaptation.

Another level of transport adaptation granularity is change of elements when there is a *vertical handoff* from one wireless network to another. Stemm et al [22] define a vertical handoff as the shift from one wireless network to another. Horizontal handoffs are shifts among access points or base stations within the same wireless network. Almost all previous works on transport protocols for heterogeneous wireless networks have implicitly assumed the need for adaptation when there is a vertical handoff.

Another alternative level of granularity for transport layer adaptation is when there is "significant" change of network characteristics within the same wireless network. By "significant" changes, we refer to change in network characteristics which would lead to degradation of the performance of the transport protocol currently being used. These can

happen due to several reasons including "horizontal handoffs" from one access point (or base station) to another possibly overloaded access point, traveling through low-signal area like a tunnel, etc. TCP-ATL implicitly assumes the need for adaptation, albeit simply parameter change, within a single wireless network when the network characteristics change.

Although assumptions are made by various works as to what level of adaptation granularity is required for optimal performance, no study has been done to investigate the ideal level of transport adaptation granularity. In this work, we identify the required adaptation granularity by evaluating the performance of transport mechanisms both across different wireless networks as well as within the same wireless network when the characteristics change.

2.4 Evaluation Model

2.4.1 Experimental Network Topology

Our experimental test-bed consists of Dell Inspiron laptops and Pentium-based personal computers. The static machines are interconnected using 10Mbps WAN connection and the mobile hosts are connected to the base station (access point) using an IEEE 802.11b wireless connection with a raw signaling bandwidth of 2Mbps. The network topology is depicted in Figure 1. The wireless link is the bottleneck for the transport connection between the static host (backlogged source S) and the mobile host (destination M). The testbed topology represents a typical scenario of wireless links and mobile hosts. In addition, our experiments focus on data transfer from the static server on the backbone Internet to the mobile host which is the common case for mobile applications (eg. Web access). The WAN connection from the source S to the base station B spans 12 Internet hops with minimal congestion.

2.4.2 Network environment parameters

Any network environment can be captured by the bandwidth, packet loss and delay of the network. We use the mean (average) bandwidth, packet loss and delay as well as the variance of the bandwidth and delay to capture the specific wireless environment. Different wireless networks have different values for the above parameters and we analyze the performance of the different transport mechanisms across varying values of the above-mentioned

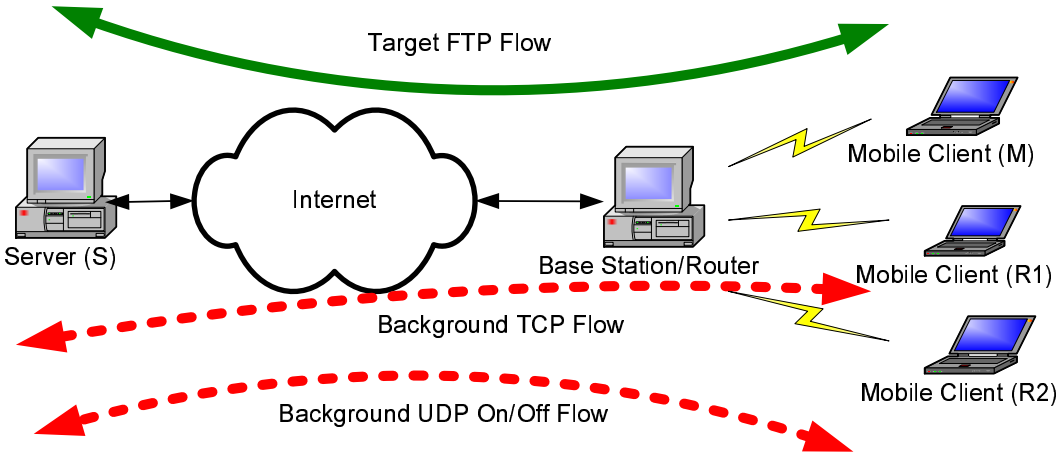


Figure 1: Network Topology for Transport Protocol Evaluation.

parameters. We use the values given in Table 1 for the different wireless data networks. The data rate is the average bandwidth that the wireless link supports; the fluctuation period is the frequency at which the bandwidth varies (the magnitude of variation is 30% of the mean bandwidth of the link) - the fluctuation period is represented as the percentage of the delay of the link; the packet loss rate is the average drop rate of the uniform loss module. The magnitude of jitter is represented as a percentage of the average one-way delay across the wireless link. We study the impact of different wireless network characteristics by varying the network conditions in terms of average bandwidth, packet loss and delay as well as bandwidth and delay fluctuation.

Now we present the details about how the required network characteristics are emulated for the specific wireless link. The base station B is a Pentium-based personal computer running Linux 2.4 kernel. The average and variance in bandwidth of the wireless link is varied by using the Hierarchical Token Bucket (HTB) traffic shaping tool. Delay on the wireless link is adjusted using a kernel module that artificially delays packets sent and received by the base station. Packet loss is emulated by using an uniform bit-error model at the base station B. The IP input/output engine of the base station protocol stack is augmented with a uniform packet loss module which determines whether a packet should be dropped. Losses are generated in both directions of the wireless channel, so

Table 1: Wireless network characteristics

	Data rate (Kbps) (Fluctuation Period)	Average Packet Loss Rate (%)	Delay (ms) (Jitter)
WLAN	2000 (500%)	1	50 (10%)
WWAN	300 (250%)	5	250 (20%)
Satellite network	100 (250%)	5	1000 (30%)

acknowledgments in the reverse direction are also dropped.

2.4.3 Traffic flows

The target application we use to study the performance of transport mechanisms is a FTP session between the source S and the mobile host M. We also have background traffic in the form a FTP flow between S and R_1, and a UDP flow between S and R_2. The wireless channel is shared by M, $R1$ and $R2$. The UDP flow is a constant bit rate application with a data rate of 200 Kbps, 100 Kbps and 20 Kbps in the WLAN, WWAN and satellite data networks respectively.

2.5 Evaluation of Transport Mechanisms

Now we present the performance evaluation of the different transport mechanisms in various wireless network conditions. We study different transport mechanisms by implementing them within the transport layer of the Linux network protocol stack.

Wireless data network optimization research can be classified into: link-layer enhancements, cross-layer optimizations and end-to-end protocol design and enhancements. In this work, we focus on the third category of wireless data networks research namely end-to-end transport protocols that provide reliable data delivery and perform congestion control and flow control. We term *reliable data delivery* and *congestion control* as transport services provided by the transport layer to the application. We view a transport protocol at three levels of detail: transport services, transport functionalities and transport mechanisms.

We define transport functionality as a module that has to be implemented by all transport protocols in order to provide the desired transport services. Examples of transport functionalities include loss detection, congestion detection and data acknowledgment.

Transport mechanism is a method of realizing a specific transport functionality. There are several ways in which a given functionality can be realized and these include the set of alternative mechanisms for the specific functionality.

We present the results and inferences by categorizing the mechanisms into the transport functionalities they implement. We evaluate the efficacy of alternate mechanisms for a particular transport functionality in different wireless environments. Using qualitative arguments, we explain why the operations of a specific mechanism is affected in specific network conditions. The primary metric used for comparing the performance of the mechanisms is the throughput achieved by the target flow between the static source S and the mobile host M. Each data point is an average of 20 application runs.

2.5.1 Data Transmission

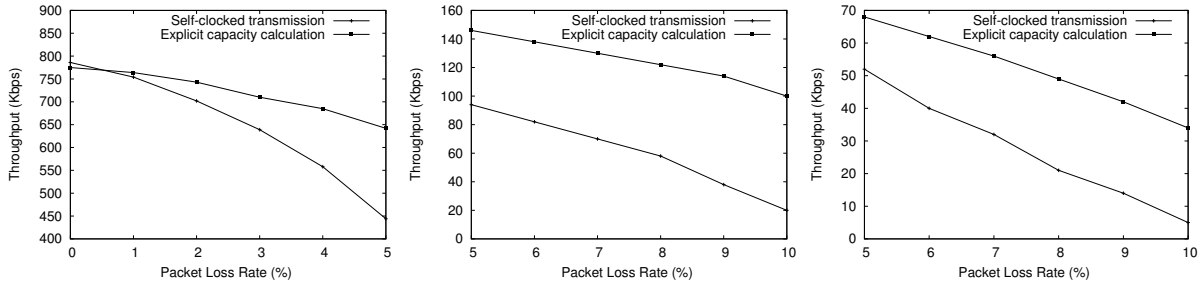
The primary function of the transport layer is to buffer the data sent by the network application at the sender side and transmit it to the receiver of the application. The transmission of data should be performed in accordance to the available capacity along the network path between the sender and the receiver.

2.5.1.1 Mechanisms

The data transmission can be performed either using an implicit clocking mechanism or using an explicit external trigger mechanism. These form the alternate choices for data transmission strategies.

Self-clocked transmission uses the packet-conservation principle wherein the transmission of a DATA packet is triggered only after the confirmation of the exit of a packet from the network. TCP-Reno (with SACK), TCP-Westwood [4], TCP-Peach [2] and TCP-Veno [7] use self-clocked data transmission mechanism.

Explicit capacity calculation calculates the actual capacity of the network path and transmits data according to the calculated capacity. WTCP [21], STP [10], TCP-Friendly Rate Control (TFRC) Protocol [9] and TCP-Vegas [17] use explicit network capacity calculation method for data transmission.



(a)WLAN (b)WWAN (c)Satellite network

Figure 2: Impact of loss rate on data transmission mechanisms

2.5.1.2 Implementation

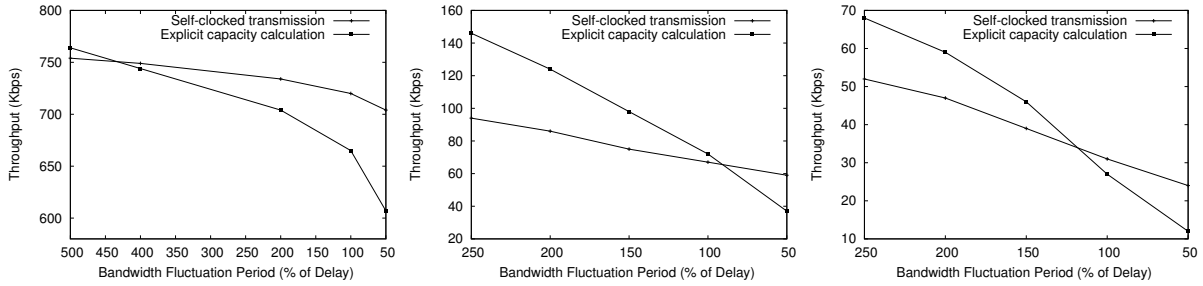
The self-clocked transmission strategy was used as such from the TCP implementation of the Linux 2.4 protocol stack. The explicit capacity calculation mechanism was implemented by changing the self-clocking strategy of TCP. In this case, the sender calculates the congestion window based on the number of packets acknowledged by the receiver and the sender transmits congestion window worth of packets every RTT irrespective of the arrival of the ACKs from the receiver. This is in contrast to the self-clocked transmission used by TCP which transmits a packet only after the arrival of an ACK from the receiver.

2.5.1.3 Relevant Network Parameters

We studied the performance of both these mechanisms in the three wireless networks under varying network conditions. We identified loss rate and bandwidth fluctuation as the *dominating network parameters*. We define dominating parameters as the set of parameters which affect the operations of mechanisms for a specific functionality. for this functionality. Average bandwidth, delay and jitter do not affect the performance of the data transmission mechanisms.

2.5.1.4 Performance Results

We can see from Figure 2, that the self-clocked transmission scheme suffers at high loss rates unlike the explicit capacity calculation mechanism. This can be attributed to the fact that the self-clocked scheme requires the constant flow of acknowledgments to trigger the data transmission and this mechanism is affected at high random loss rate. The explicit



(a)WLAN (b)WWAN (c)Satellite network

Figure 3: Impact of bandwidth fluctuation on data transmission mechanisms

capacity mechanism transmits data irrespective of the arrival of the acknowledgments at the sender. Thus it is not affected adversely by high wireless packet losses.

We then study the performance of the two mechanisms under bandwidth fluctuation scenarios in the three wireless networks. In this scenario, the bandwidth fluctuates at a specific frequency which is represented as a percentage of the delay across the wireless link. We note in Figure 3 that explicit capacity calculation mechanism suffers at high bandwidth fluctuation scenario in all the three networks. This is because of the inherent delay in the response to fluctuations in the network bandwidth using the explicit capacity calculation case. Hence when the bandwidth fluctuates at a high rate, the sender is not able to adapt to the fluctuations and hence would end up under-utilizing the available capacity or overflowing the buffers at the routers. Using self-clocked transmissions, the sender instantaneously adapts to the network bandwidth fluctuations. Thus we can see from the graph that self-clocked transmission strategy is not adversely affected by the bandwidth fluctuations in the network. Figures 3(b) and 3(c) show that even within the same wireless network (WWAN and satellite), there is a shift in the optimal performance using the two data transmission strategies.

2.5.1.5 Inferences

From the performance results under differing network conditions, we observe that for optimal transport protocol performance, it is imperative to switch between the self-clocked strategy and the explicit capacity calculation mechanism. Further as we note in Figures 3(b) and 3(c), different mechanisms perform well under different network conditions even

within the same wireless network. Hence we need to change the mechanisms even within the same wireless network for the best performance. These lead to the inference that a good transport adaptation solution should accommodate transport mechanism change at a granularity finer than interface changes.

2.5.1.6 Congestion control

In a shared network such as the Internet, the available capacity of the network can change and hence the transport protocol should perform congestion control wherein the amount of data transmitted into the network complies with the estimated network capacity.

2.5.1.7 Mechanisms

The congestion control strategies vary in the way the network capacity is abstracted by the transport layer.

Window-based congestion control mechanism uses "congestion window" to abstract the available network capacity and transmits data in a way such that at any given instant congestion window worth of data is outstanding in the network. TCP-Reno, TCP-Westwood and TCP-Peach use window-based congestion control scheme.

Rate-based congestion control abstracts the network capacity as a calculated rate and the data transmission rate is adapted to the calculated *rate supported by the network*. Rate Adaptation Protocol (RAP) [19], TFRC and WTCP are examples of transport protocols that use rate-based congestion control.

Hybrid congestion control uses a mix of window and rate-based congestion control scheme. It calculates the network capacity as a congestion window but transmits data based on a rate calculated from the congestion window and the Round Trip Time (RTT) of the connection. STP and TCP-Vegas use the hybrid congestion control scheme.

2.5.1.8 Implementation

The window-based congestion control mechanism of TCP-Reno was used as-is from the protocol stack in the Linux kernel. For the rate-based congestion control mechanism, we

implemented a TFRC-like rate-based congestion control mechanism to replace the window-based strategy at the sender. The sender adjusts its sending rate based on the TCP equation [9] and it uses an average loss interval for loss rate estimation. The hybrid congestion control strategy is a simple modification to the window-based congestion control mechanism used by TCP. The sender instead of sending congestion window worth of data at one burst, calculates the sending rate based on the current value of the congestion window and the RTT and uses it to send DATA packets to the receiver.

2.5.1.9 Relevant Network Parameters

We identify bandwidth and bandwidth variations as the primary parameters that affect the performance of the congestion control mechanisms. This is because of the inherent property of the congestion control strategies to adapt to the bottleneck bandwidth of the connection and the efficacy of the strategies would depend on the value of the average and variance of the network bandwidths.

2.5.1.10 Performance Results

We observe that both the rate-based and hybrid congestion control mechanisms use explicit timers to clock the data transmission. This would incur significant overheads at the sender due to the fine-grained timer values that have to be maintained when the bandwidth is high. This has been noted in several works in the research literature including [21]. We also note that the bursty nature of window-based congestion control mechanism is problematic under highly multiplexed connections with multiple flows sharing the same bottleneck router. In this scenario, window-based congestion control would suffer because of its burstiness leading to temporary overflowing of the router queues and hence leading to unnecessary packet drops. The burstiness of the window-based scheme also has implications with regard to the calculation of the delays that packets encounter and hence affects delay-based congestion detection schemes as well as timeout-based loss recovery mechanisms. We describe the performance of these mechanisms later in the section.

2.5.1.11 Inferences

From the results above, we conclude that irrespective of the wireless network, an ideal transport adaptation solution should change the congestion control mechanism based on the underlying wireless conditions.

2.5.2 Congestion Detection

Although transport protocols perform data transmission complying to the available capacity of the network, there are several cases (eg. sudden influx of new flows) where congestion can set in the network. Protocols should be able to detect the congestion and recover by cutting down the rate of data transmission. Hence congestion detection is an important functionality that has to be implemented by any transport protocol providing congestion control.

2.5.2.1 Mechanisms

Congestion detection can be performed based on several indicators for the onset of congestion.

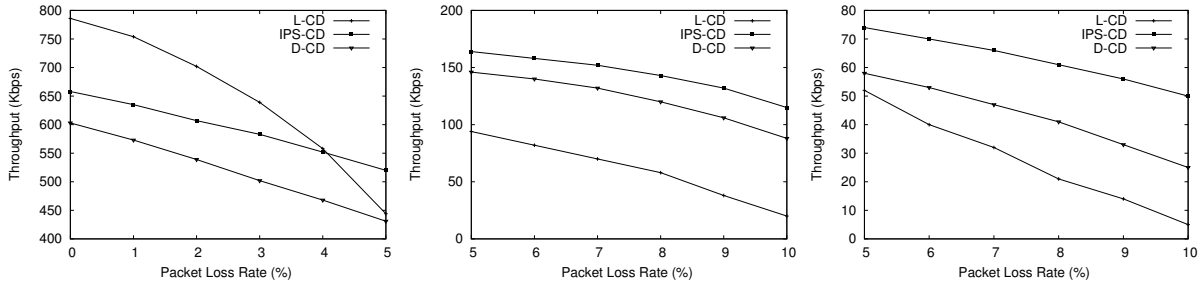
Loss-based congestion detection ($L-CD$) uses packet loss as a primary indicator of congestion. The losses in the network are detected by means of acknowledgments returned by the receiver. TCP-Reno, STP, TCP-Westwood, TCP-Veno, TFRC and TCP-Peach use loss-based congestion detection mechanism.

Delay-based congestion detection ($D-CD$) uses increase in delay as the indication of network congestion. TCP-Vegas uses delay-based congestion detection.

Inter-packet separation based congestion detection ($IPS-CD$) monitors the separation between packet transmissions at the sender and packet receptions at the receiver and uses increase in receiver inter-packet separation compared to the sending rate as the indicator of congestion. WTCP uses inter-packet separation as an indicator of congestion.

2.5.2.2 Implementation

For the loss-based congestion detection mechanism, we use the already implemented version of the TCP-Reno protocol within the Linux protocol stack. In order to implement the



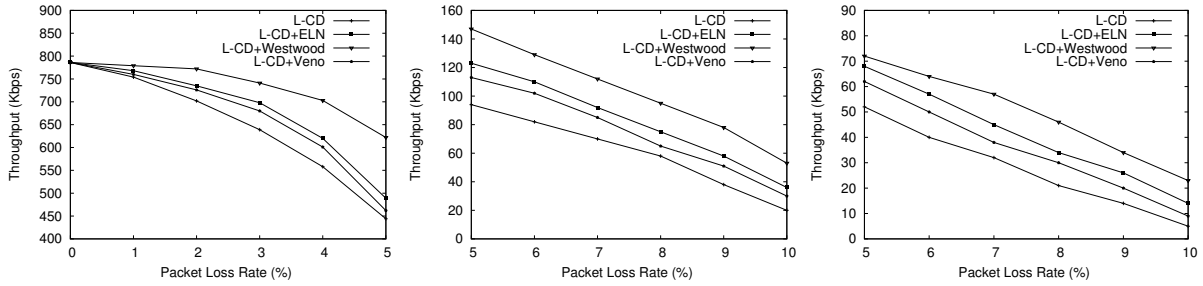
(a)WLAN (b)WWAN (c)Satellite network

Figure 4: Impact of loss rate on congestion detection mechanisms

delay-based strategy, we change the congestion notification mechanism used by TCP-Reno to that used by TCP-Vegas. Thus instead of cutting down the congestion window based on losses, the delay-based congestion detection cuts down the congestion window based on RTT increase. The sender maintains both the current RTT and minimum RTT experienced by the connection and controls the congestion window/rate when the difference between the two increases beyond a threshold value.

The inter-packet separation based congestion detection mechanism is essentially a WTCP-based strategy where the receiver calculates the congestion window based on the variations in the separation between the packets sent at the sender and the packets received at the receiver. It performs AIMD control of the congestion window with the same parameters for the thresholds for inter-packet separation comparison as WTCP. The congestion window adaptation mechanism of the TCP Reno implementation within the Linux protocol stack was changed to perform the above-described manipulation.

We also implement several enhancements to the original TCP-Reno loss-detection mechanism. The Explicit Loss Notification (ELN) [3] mechanism assists the TCP sender in distinguishing congestion losses from random wireless (corruption) losses. When a packet is dropped on the wireless link the receiver sets an "ELN" bit in the cumulative acknowledgment sent corresponding to the lost packet to notify the sender that the loss was not due to congestion. When the sender receives the information with the duplicate acknowledgments, the sender performs fast retransmit without invoking the associated congestion control procedures.



(a)WLAN (b)WWAN (c)Satellite network

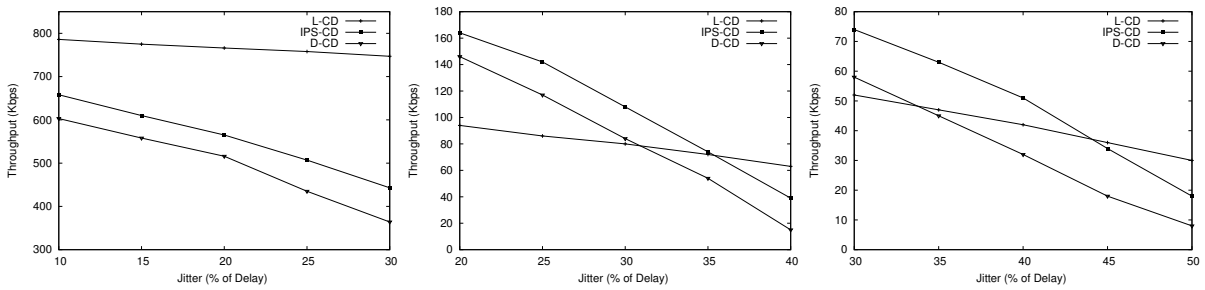
Figure 5: Impact of loss rate on enhancements to loss-based congestion detection

We also implement the bandwidth estimation enhancement proposed by the authors of TCP Westwood [4]. In this mechanism, the sender monitors the acknowledgment reception rate and from the ACK reception rate estimates the data rate currently achieved by the connection. When the sender detects packet loss (after a timeout or after the reception of 3 DUPACKs), the sender uses the bandwidth estimate to properly set the congestion window (rate) and the threshold value used by TCP's slow start bandwidth probing mechanism.

Another enhancement we implement is the Veno [7] mechanism for efficient loss classification. This mechanism uses the delay-based congestion detection mechanism of TCP-Vegas to classify losses as congestive and non-congestive losses. The sender maintains an estimate of the current RTT and the minimum RTT experienced by the network connection. The sender assumes that it is in non-congestive state if the difference between the minimum RTT and current RTT is smaller than a threshold value and congestive state, otherwise. If the sender detects loss (by reception of 3 DUPACKs), when in non-congestive state it assumes the loss is due random wireless error(s). If the loss occurs during congestive state, the sender assumes the loss is due to congestion.

2.5.2.3 Relevant Network Parameters

We study the performance of the congestion detection mechanisms in the three wireless networks under varying network conditions. We identify loss rate and delay jitter as the *dominating conditions* for this functionality since the strategies use one of these metrics as the primary indicator of congestion. Average bandwidth, bandwidth variance and delay do not affect the performance of the congestion detection strategies.



(a)WLAN (b)WWAN (c)Satellite network

Figure 6: Impact of jitter on congestion detection mechanisms

2.5.2.4 Performance Results

The first study is the evaluation of the strategies under different loss rates in the three network scenarios. We note from Figure 4 that the loss-based congestion detection scheme degrades in performance under high-loss conditions in all the three wireless networks considered. We observe that the throughput achieved using the loss-based congestion detection mechanism degrades rapidly as the packet loss rate increases. The delay-based and inter-packet-separation based congestion detection scheme perform much better than loss-based scheme under high random losses. The reason for the under-performance of the loss-based congestion detection scheme is that it is unable to detect congestion accurately due to high number of random losses in the network.

We also study the performance of potential enhancements to assist the loss-based congestion detection mechanism. We observe in Figure 5, that in all the three network environments, although the enhancements improve the performance of the loss-based congestion detection scheme, they show the same degradation performance as the loss-based congestion detection. This is because, at high loss rate, no enhancement to the loss-detection mechanism is useful because the sender does not receive complete information regarding the packet reception status at the receiver. Although both ELN and Veno mechanisms are loss classification enhancements, we note from Figure 5 that ELN performs better than Veno because of its accurate notification of the nature of loss unlike Veno which uses a heuristic to determine the cause of loss.

Next, we evaluate the congestion detection strategies under varying delay jitter conditions. Figure 6 shows that both delay-based and inter-packet separation-based mechanisms suffer at high jitter conditions. This is because of their assumption that increase in delay (or inter-packet separation) in the network is an indicator of network congestion. They assume that the increased delay encountered by the packets is due to the fact that the router buffers are getting full and hence the packets are spending more time in the router queues. They do not take into account the possibility of delay variations due to non-congestion related causes such as link-layer retransmissions, short-term unfairness of MAC protocols and multiplexing of data flows at the backbone router. Loss-based congestion detection scheme is not affected by such non-congestion related delay spikes.

2.5.2.5 Inferences

We observe from the study of the congestion detection strategies that different mechanisms perform optimally in different network conditions and change of conditions even within the same network requires the switch in the congestion-detection scheme being used. Further we also conclude that static enhancements to transport mechanisms is not sufficient for optimal performance.

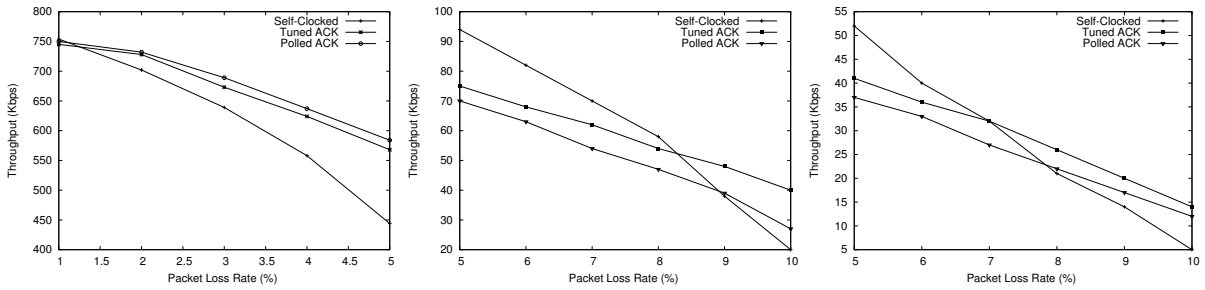
2.5.3 Data Acknowledgment

A transport protocol that provides reliable data delivery has to use an acknowledgment mechanism for detecting losses and thereby recovering from them. Further, the loss-based congestion detection scheme discussed above requires reverse path acknowledgments (ACKs) to detect losses.

2.5.3.1 Mechanisms

The receiver can send an acknowledgment based on one of the following triggers : data arrival, sender requests or at a specific timer interval.

Self-clocked acknowledgment scheme uses the arriving DATA packets as triggers for sending ACKs back to the sender. Usually an ACK is sent for every k DATA segments that is received correctly by the receiver. TCP-Reno, TCP-Westwood, TCP-Peach and TCP-Vegas



(a)WLAN (b)WWAN (c)Satellite network

Figure 7: Impact of loss on data acknowledgment mechanisms

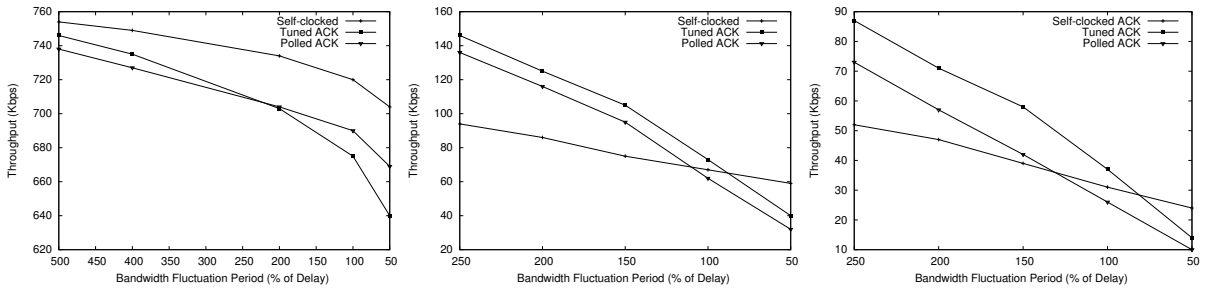
use self-clocked acknowledgment mechanism.

Using *Tuned-rate acknowledgment*, sender calculates a specific frequency for the receiver to send the ACKs. WTCP uses this type of acknowledgment scheme.

Polled acknowledgment scheme is one in which the senders use polling to solicit ACKs from the receiver. The sender sends out a poll, say once every RTT to solicit information about the packets that have arrived correctly at the receiver. STP uses the polled acknowledgment scheme.

2.5.3.2 Implementation

The self-clocked acknowledgment scheme of TCP-Reno is used as-is in the Linux implementation. We implemented the tuned ACK mechanism by changing the TCP sender to transmit a period p to the receiver which then transmits one ACK every p seconds to the sender irrespective of the arrival of DATA from the sender. The receiver uses both cumulative and selective acknowledgment options in the ACK packet sent to the sender. For the polled ACK strategy, we change the TCP sender implementation to mark a reserved bit in a DATA packet once every RTT to act as a poll to solicit an ACK from the receiver. The receiver sends an ACK only when it sees a poll bit set in a correctly received DATA packet. But if the receiver receives three unordered packets it sends an ACK indicating loss to the sender. This mechanism implementation is similar to the polled ACK mechanism used by STP.



(a)WLAN (b)WWAN (c)Satellite network

Figure 8: Impact of bandwidth fluctuation on data acknowledgment mechanisms

2.5.3.3 Relevant Network Parameters

We identify packet loss rate and bandwidth fluctuation as the dominating conditions for these mechanisms. The loss rate affects the efficiency of the acknowledgment scheme and also the amount of information the sender has about the receiver buffer. Bandwidth fluctuation is important in analyzing the efficacy of the specific acknowledgment strategy in adapting to the changes in the network bandwidth. We find that average bandwidth, delay and jitter do not affect the performance of the acknowledgment strategies.

2.5.3.4 Performance Results

First we study the performance of the acknowledgment strategies under varying loss rates in the three wireless network environments. We observe in Figure 7, that the self-clocked acknowledgment scheme suffers under high packet loss rate conditions. This is due to the fact that at high random loss conditions, the receiver is not able to transmit sufficient information to the sender using the self-clocked acknowledgment scheme. This, in turn, affects both the retransmission and congestion control mechanisms at the sender side. Since in the tuned ACK and polled ACK mechanisms the receiver periodically sends ACKs back, the sender has correct information regarding the receiver buffer and hence performs efficient data transfer.

We also study the performance of the acknowledgment strategies under fluctuating bandwidth conditions. From Figure 8, we note that both the tuned ACK and polled ACK scheme suffer at high bandwidth variance conditions. This is because at high bandwidth variations,

the sender is unable to receive sufficient information corresponding to the variation in bandwidth and thereby adapt correctly. The self-clocked strategy does not suffer as much as the controlled ACK schemes because of the timely information obtained by the sender irrespective of the fluctuations.

2.5.3.5 Inferences

We observe that the self-clocked scheme performs well under bandwidth fluctuations but degrades in high loss environments. Tuned ACK and polled ACK schemes perform considerably well under lossy conditions but suffer when there is frequent fluctuations in bandwidth. Thus we infer that optimal performance can be obtained by using the acknowledgment strategy best suited for a given environment. We also see from Figures 7(b) and 7(c) and 8(b) and 8(c) that even within the same wireless network, different mechanisms perform well in different conditions and hence swapping of mechanism within the same wireless network is vital to obtain best performance.

2.5.4 Suffix-loss Recovery

Acknowledgments sent by the receiver do not guarantee complete loss detection since there is a possibility for ACKs to get lost. The sender should be able to recover from losses even in the absence of ACKs (due to reverse path losses). This functionality is termed as suffix or blackout loss recovery. Blackouts which are not unusual in wireless networks can lead to a scenario where the sender does not receive ACKs from the receiver.

2.5.4.1 Mechanisms

We can use two strategies to recover from losses in the absence of ACK information. Either the sender can wait for a threshold amount of time before it retransmits data to the receiver or it can periodically probe the receiver for packet reception information.

Timeout-based loss recovery maintains a timeout based on the RTT of the connection and also the variations of the RTT. The sender assumes loss of DATA if it does not receive ACKs within the timeout period and retransmits the DATA packets to the receiver. All TCP variants use timeout-based suffix-loss recovery mechanism.

Probe-based loss recovery uses probe packets for obtaining information from the receiver. Here the sender, after sending DATA and waiting for some threshold amount of time, sends small non-DATA probe packets to solicit ACK information from the receiver. This would continue until an ACK is received by the sender at which point it reverts back to the normal DATA transmission mechanism. WTCP uses probe-based mechanism to recover from suffix and blackout losses.

2.5.4.2 Implementation

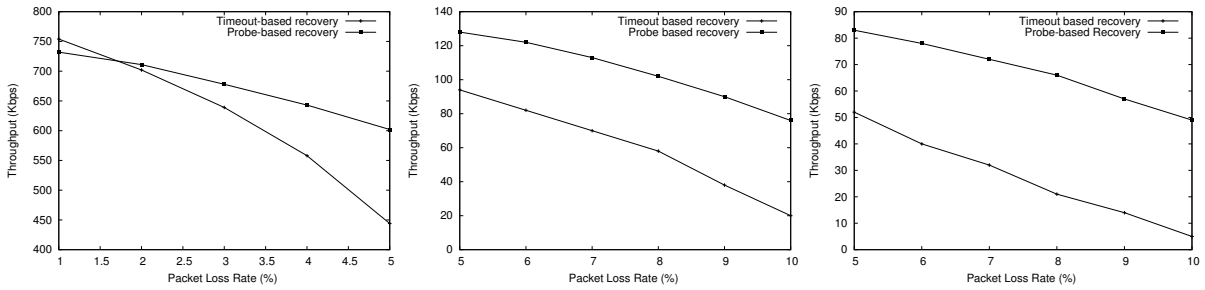
We use the timeout based mechanism used by TCP-Reno for the first strategy. We also implement the Eifel algorithm to enhance the operation of the timeout strategy used by TCP-Reno with the objective of avoiding spurious timeouts. We implement the probe-based recovery strategy by modifying the data transmission mechanism of TCP-Reno and including a probe module after every data transmission. The probing parameters are the same as those used in the WTCP implementation.

2.5.4.3 Relevant Network Parameters

We identify delay jitter as the dominating condition for the suffix-loss recovery strategies. This is because of the fact that the only reason for the potential degradation of the suffix loss recovery process is the inaccuracy in the estimation of the time-to-wait before retransmission and congestion control measures. The above mentioned inaccuracy is increased as the variations in the delay values become larger. We identify that average bandwidth, bandwidth variations, average delay and loss rates do not degrade the performance of the suffix-loss recovery mechanisms.

2.5.4.4 Performance Results

In Figure 9, we present the results of the performance of the timeout-based and probe-based suffix loss recovery mechanisms under varying delay jitter conditions in the three wireless networks. We observe that as the delay jitter increases in the three wireless environments, the performance of the timeout based recovery strategy degrades more than that of the



(a)WLAN (b)WWAN (c)Satellite network

Figure 9: Impact of loss rate on suffix loss recovery mechanisms

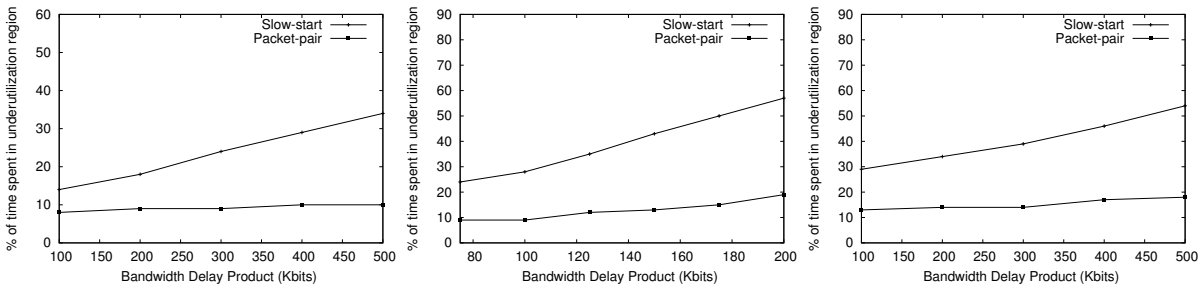
probe-based strategy. The reason for degradation in performance of the timeout mechanism can be explained as follows : As the delay variations increase, the sender performs a conservative timeout estimate which is usually more than the actual delay of the network and hence it is inefficient in recovering from losses. This observation holds even with the enhancement of the timeout strategy with the Eifel algorithm, which avoids false timeouts due to delay spikes. On the other hand, the probe based approach performs considerably well when compared to the timeout strategy because of its weak dependence on the delay measurements of the network link. We note that the probe-based mechanism, with appropriate tuning of the probing frequency, performs well across all environments. Although, the probe-based recovery strategy would incur overheads when the frequency of polling is too high, we note that the overhead is small because of the small-sized non-data probes.

2.5.4.5 Inferences

From the study of the performance of suffix-loss recovery mechanisms, we observe that this is a case where proper tuning of a protocol parameter in different network conditions would achieve optimal network performance. But as seen in previous evaluation of strategies, mechanism change which is more complex than parameter change is vital for the best performance of certain functionalities.

2.5.5 Start-up Capacity Determination

The sender should use some mechanism to initially determine the network capacity available along the end-to-end path for the connection.



(a)WLAN (b)WWAN (c)Satellite network

Figure 10: Impact of network BDP on start-up bandwidth probing mechanisms

2.5.5.1 Mechanisms

Initial bandwidth estimation can be performed either by finding the capacity using probing or by periodic increase of the transmission rate until congestion is detected.

Slow-start mechanism : This mechanism starts from a small capacity estimate and increments the data transmission rate/window exponentially until congestion is detected. This would give a rough estimate of the network capacity of the end-to-end path. All TCP variants and STP use slow-start mechanism for startup capacity determination.

Packet-pair based probing : This mechanism transmits two packets and estimates the delay between the packets when received by the receiver to determine the capacity available along the network path. WTCP uses packet-pair approach for initial bandwidth estimation.

2.5.5.2 Implementation

The slow-start mechanism was the same as used by the TCP-Reno implementation within the Linux protocol stack. The packet-pair strategy was implemented by changing the slow-start of TCP to transmit two packets in succession and use the delay in the reception of the packets to determine the available bandwidth. After the packet-pair based initial bandwidth determination, the congestion control phase is entered as usually.

2.5.5.3 Relevant Network Parameters

We identify that the bandwidth-delay product (BDP) is the primary influencing parameter for the start-up bandwidth determination mechanism because of the target value the start-up mechanism is attempting to determine.

2.5.5.4 Performance Results

We observe in Figure 10, that the slow-start mechanism used by TCP suffers under high bandwidth-delay conditions in all three wireless network environments. This is primarily because of its inherent inefficiency in reaching the bottleneck bandwidth of the network path. On the other hand, the packet-pair approach achieves optimal performance across all BDP conditions, because of its short-time of operation. But packet-pair approach would not be efficient in highly varying conditions of network capacity because of its instantaneous determination of capacity.

2.5.5.5 Inferences

We infer from the above results that an optimal transport adaptation solution would need to change the start-up bandwidth determination mechanism depending on the network conditions.

2.6 Summary of Results

The extensive performance evaluation of the various strategies for the transport functionalities helps to answer the two questions regarding ideal transport adaptation. From the evaluation results we conclude that an ideal transport adaptation solution **should accommodate transport mechanism change** for achieving the best performance across heterogeneous operating network environments. Further we prove the need for fine-grained transport adaptation at a level that is much finer than just interface handoffs. We also note that for each functionality there are a set of parameters which influence which strategy to use for optimal performance and hence they can potentially act as "triggers" for the transport adaptation solution to change the mechanisms being currently used.

CHAPTER III

*TP : DESIGN AND IMPLEMENTATION

We have so far studied the performance of transport mechanisms in different wireless environments and found the ideal nature and granularity of transport protocol adaptation. Now we present the design elements of an adaptive transport layer framework that can accommodate the mechanisms used in different transport protocols, and dynamically transform itself to exhibit the behavior of the transport protocol best suited for a given environment. We also describe the software architecture of the framework within the Linux protocol stack and the execution of the framework in the previously described topology. The principal focus of the adaptive framework is the ability to accommodate multiple alternative transport mechanisms independent of the specifics of the protocol(s) or mechanism(s) that will be absorbed into the framework from different transport layer solutions. Hence, we primarily focus on how the framework supports such dynamic reconfigurability. We also provide case studies for the operation of the *TP implementation in the next chapter.

3.1 Design Goals

The design goals of the *TP framework reflect the solutions to the transport adaptation problems we have identified using the performance evaluation study in the previous section. The goals include:

3.1.1 Reconfigurability

A key design goal of *TP is the ability to reconfigure itself to use the transport layer mechanisms best suited for a given environment. The reconfiguration of mechanisms is triggered by changes in network characteristics such as the increase in loss rates and delay jitter, or simply interface handoffs. Unlike other configurable frameworks and protocols proposed in related work [24, 20], *TP is designed to support run-time reconfigurability with minimal application intervention. Note that since most of the changes in network

characteristics do not require the awareness of the application, it is desirable to design an transport layer framework that can *seamlessly* perform reconfiguration with minimal disruptions to the application.

3.1.2 Extensibility

*TP is designed as a generic framework that can accommodate various mechanisms used in different transport layer protocols. Therefore, *TP by nature is an extensible framework that can "plug-in" any new or existing transport mechanisms. The performance of *TP is not limited to any specific transport protocol or mechanism. Instead, whenever a better mechanism tailored to the characteristics of a given environment becomes available, *TP can use these protocols for achieving higher base-line performance. The design of *TP ensures that whenever the network characteristics become favorable to any module already registered, it will be invoked to perform the corresponding functionality. Toward this goal, *TP defines a set of interface functions that facilitate new protocols to be incorporated into the *TP framework.

3.1.3 Minimal Overheads

While *TP allows flexible reconfigurability and extensibility of transport mechanisms, it does not trade overheads for the ability of fine-grained transport adaptation. *TP is designed to incur minimal overheads compared to a static transport protocol (e.g. WTCP and STP). The overheads that need to be minimized in a dynamic protocol like *TP include: (i) Complexity: The execution efficiency of *TP should not be sacrificed simply because modules are dynamically composed. In other words, *TP should minimize the computation overheads, such that any host can support as many connections using *TP as connections using a static transport protocol. (ii) Redundancy: The redundancy due to repetitive implementations of any functionality in different modules should be minimized. In other words, the memory footprint of *TP should be kept at a minimum. (iii) Latency: The latency incurred during reconfiguration of mechanisms can cause interruptions or disruptions at the application layer. Since it is possible that reconfiguration occurs several times in a connection, such a "reconfiguration" latency should be kept at a minimum.

3.2 Design Elements

We now present the key design elements in *TP that allow it to meet the design goals described earlier.

3.2.1 Triggers

The reconfiguration of mechanisms in *TP is triggered by changes in one or multiple network parameters pertinent to the transport mechanism in consideration. As mentioned in Chapter 2.5, the transport functionalities have dominating parameters associated with them which influence which strategy to use in each network conditions. Each potential module to be used by *TP first specifies the network parameters that need to be monitored, as well as the conditions (e.g. threshold values) for triggering the reconfiguration. *TP is responsible for initiating the reconfiguration when the network conditions are met. Whenever a decision is made to swap in a module, the corresponding module is loaded into *TP, and replaces the current module in use. The monitoring of the triggers and related parameters is again performed by *TP.

3.2.2 Separation of Core and non-Core Modules

As we mentioned earlier, the *TP framework is invariant and independent of the specific protocols used, while individual mechanisms may be swapped in and out depending on the network characteristics. *TP adopts a structured separation of permanent *core* and configurable *non-core* modules. The core is the the *TP framework itself, and the non-core can be considered as different transport mechanisms that may change across different operating conditions. The core consists of the following modules: (i) providing a fixed interface to the application layer, and the IP layer; (ii) all transport framework control operations such as trigger monitor and management, and transport framework reconfigurations; (iii) the transport engine that drives and glues together the non-core modules; and (iv) the data buffers where any shared data between non-core modules are maintained by the core. Since the core does not change, it can be optimized for achieving higher execution efficiency and minimizing overheads. The non-core modules on the other hand consist of all the transport

mechanisms implementing various transport functionalities.

3.2.3 Modular Architecture and Execution Model

*TP uses a modular architecture for incorporating the non-core modules. Note that the *TP core is a static component, and hence does not need to be modular. Since the non-core transport mechanisms are those that change across different network environments, the modular design of the non-core mechanisms allows for *fast swapping of modules* in and out of the kernel, and *fine-grained adaptation* of the transport protocol mechanisms. Together with an event-driven execution model, the modular architecture facilitates ease of reconfigurability. In *TP, the core maintains an event queue that is served through the invocation of non-core modules. When modules are invoked, they may in turn register further events in the event queue, thus ensuring the execution of the proper set of mechanisms in the appropriate order. Specifically, the event-driven execution greatly simplifies the reconfiguration process in *TP as it merely involves replacement of appropriate event-handlers. Note that most implementations of TCP actually use event-driven operations, although in a very coarse grained fashion as the event-driven operation begins with the arrival of an ACK, but all subsequent transport layer processing in sequential control-flow fashion.

3.2.4 State Propagation

*TP allows the inheritance of transport layer state from one non-core module to another when a reconfiguration is performed. Examples of states that can be inherited across modules include the data buffer, the SACK scoreboard for reliability, and the advertised window size of the receiver, etc. *TP enables such state propagation by allowing non-core modules to maintain both *public*, and *private* state. Any public state is maintained by the core, while the private state is maintained by the non-core module. Thus, when a non-core module is swapped out, and a new non-core module is swapped in, the new module has access to the public state left behind by the old module.

3.2.5 Mobile-host Centric Operations

*TP allows dynamic switching of different protocols on the fly, and extending the protocol stack when newer protocols are developed. While it is reasonable to assume that the mobile host will need to accommodate these protocols for achieving the best performance in different wireless environments, it may not be the case for the static Internet host. This is because static hosts in such a scenario will have to accommodate *all possible* transport protocols in anticipation of communication from any mobile host in the Internet – which is clearly infeasible given that there can be a multitude of protocols corresponding to the large number of heterogeneous wireless access technologies. Therefore, a setting adopted by *TP is to make the mobile-host the primary seat of transport layer intelligence, irrespective of whether it is acting as a sender or a receiver. In such a setting, any change runtime or otherwise can be performed solely at the mobile host without the intervention at the static host.

3.3 Software Architecture

*TP is a mobile-centric framework with the mobile host being the primary control for the protocol operation. The static host in *TP is very simple. When the static host acts as the receiver, it simply sends feedback information used by the mobile host (sender) such as ACK and SACK information. When the static host acts as the sender, on the other hand, it merely responds to requests from the mobile host for sending data. In other words, the static host plays a passive role responding only to instructions sent by the mobile host, where the reconfiguration occurs depending on the network environment.

Figure 11 shows a high level architectural diagram of *TP at the mobile host. We refer to it as *TP for purposes of presentation, and qualify it only when the reference is to the static host. As shown in the figure, the *TP functionality is separated into the fixed *core* and reconfigurable *non-core*. The core consists of the following components:

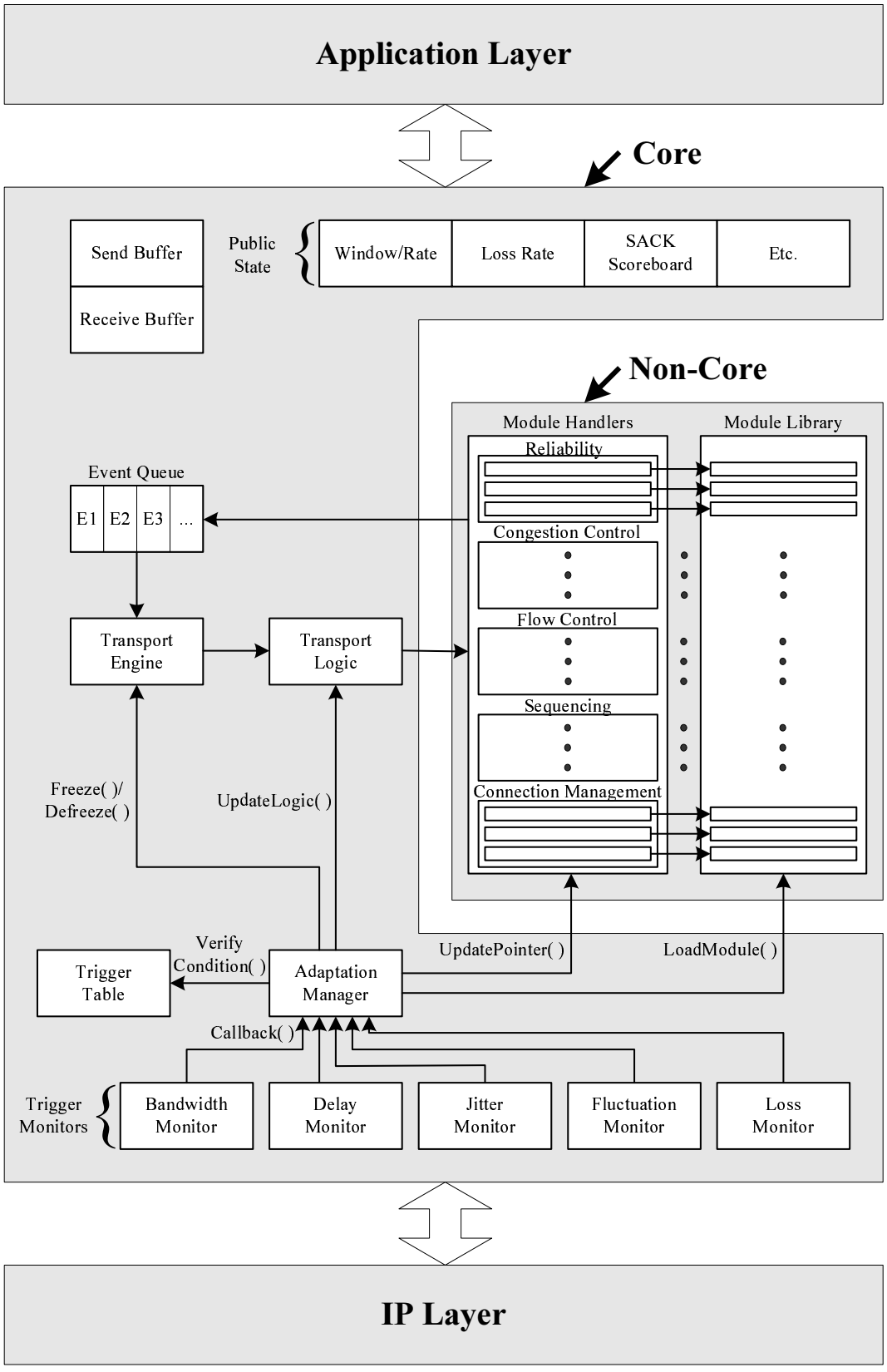


Figure 11: *TP Framework.

3.3.1 Interfaces with the Application and IP

The core provides a fixed interface for the application layer and the IP layer to communicate with *TP. Any communication coming in from the application including data and control (say, socket options, connection open and close) is handled by the core. Similarly, any communication coming in from the IP layer is handled by the core.

3.3.2 Global Data Structures

The data maintained by the core includes the send and receive buffers, the public state, and the event queue. The handling of the buffers by the core is clear since the reconfiguration of transport modules should not affect the data in the buffer. The public state is used for *state inheritance* and serves as a shared space for non-core modules to communication with each other. Finally, the event queue is related to *TP's execution model. Its role will become clear as we discuss the transport execution later.

3.3.3 Transport Engine

The core in *TP supports the backbone of the transport layer framework, and hence any intelligence that pertains to the transport protocol operations (which will be changed in different environments) is provided by the non-core modules. Note that in *TP, not only the transport modules can be reconfigured, but the *logic* (e.g. sequence of execution) for the execution of these non-core modules (in response to transport layer events) is also reconfigurable. A transport protocol developed in the *TP framework thus not only can use *TP to incorporate new transport modules, but also can decide how and in what form the modules are used. This is facilitated by allowing the transport protocol developed to provide a *transport logic*. The logic is loaded along with the non-core modules, but it is the core's transport engine that *executes* the transport logic. The transport engine, depending on events registered in the event queue, uses the transport logic to execute the appropriate non-core modules.

3.3.4 Reconfiguration Entities

There are three components in the core that are related to the reconfiguration initiation process: the trigger table, trigger monitors, and the adaptation manager. Non-core modules register in the core the trigger and the condition for module invocation. The trigger is a logical combination of network parameters monitored by various trigger monitors. The adaptation manager receives callbacks from the trigger monitors when the conditions specified by non-core modules are met. The adaptation manager uses the trigger table to identify which modules and logic to use, and loads the corresponding modules from the module library into the non-core.

On the other hand, the non-core modules reflect the traditional transport layer intelligence, including reliability, congestion control, and flow control. The specific logic used to combine the modules together for achieving a transport layer functionality is part of the *transport logic*. The transport logic in *TP is in fact an event/handler table that maps the events registered with the engine and the available non-core modules. Thus, it acts as the liaison between input events to the transport framework, and the actual transport functionality. Moreover, it also acts as the enabler for communication between the non-core modules, through the generation and serving of internal events. Non-core modules communicate with each other solely through the generation and servicing of events, and the public state maintained by the core. (They are, however, allowed to maintain private state.) The non-core module can read and write from/into the public state. Note that no explicit synchronization schemes are required to control access to the public state as long as the number of kernel threads is equal to one as in Linux. Non-core modules can interface with the core through registering events in the event queue, and data exchange in the send/receive buffer.

3.4 Transport Execution

Transport execution in *TP refers to the structure of how different non-core modules are executed for achieving transport layer functionalities, including connection management, congestion control, flow control, and reliability. *TP uses the following granularity to

identify non-core modules supplied by different transport layer protocols (the list is indexed by transport layer functionalities):

- *Connection Management:* connection establishment, connection termination, and connection reconfiguration;
- *Congestion Control:* transmission paradigm (window or rate based), start-up rate estimation, congestion avoidance (probing for rate increase or maintenance), congestion detection, and congestion control;
- *Flow Control:* window advertisement;
- *Reliability:* loss detection, loss distinction, loss indication, and loss recovery;
- *Sequencing:* sequencing and re-sequencing;
- *Feedback:* feedback control.

*TP uses an event-driven execution model. The core maintains an event queue, and when the *TP transport engine is provided execution context by the kernel, the head-of-line event in the queue is served. The transport engine uses the event/handler table (i.e. the transport logic in the non-core) to find the proper handler that should be used to process the event. Each handler in the transport logic points to a non-core module (maintained in the module library) currently chosen for use by *TP. The non-core module thus will be invoked to serve the event. Modules may in turn enqueue (register) further events in the event queue when they are invoked. The event registered in the queue will invoke the execution of another non-core module when it is served by the transport engine. Note that the transport logic maintains the mapping between events and handlers (non-core modules), and hence the order of execution for non-core modules can be appropriately reconfigured depending on the transport protocol in use.

3.5 Trigger Management

As we discussed earlier, the initiation of the adaptation process in *TP is the result of the collaboration by three components in the core: the trigger table, trigger monitors,

and adaptation manager. The trigger table has two fields for maintaining the mappings between triggers and non-core modules. A trigger is a logical combination (e.g. and, or) of network parameters monitored by trigger monitors. Each trigger monitor is responsible for monitoring the change of one network parameter along the path. Currently, *TP defines the following five network parameters: bandwidth, delay, loss, bandwidth fluctuation, and delay jitter.

Monitoring of a particular network parameter is started only when requested by non-core modules. First, non-core modules choose from the five parameters to set the desired trigger (e.g. bandwidth *and* jitter) in the trigger table. It also specifies the conditions (e.g. if bandwidth is greater than 300Kbps and jitter is larger than 10ms) when it should be invoked in respective trigger monitors. The chosen trigger monitors then, if not already started, start the monitoring process. Callbacks from the trigger monitors (indicating certain conditions are met) are received by the adaptation manager. The adaptation manager uses the trigger table to verify if all conditions for the trigger registered by any non-core module are validated. If so, it initiates the adaptation process with the non-core module listed on the trigger table as we discuss in the next section.

3.6 Adaptation Process and State Propagation

Whenever network characteristics change that trigger reconfiguration of *TP, new non-core modules need to be swapped in to replace the corresponding modules in use. The reconfiguration process is initiated when the adaptation manager finds the non-core modules to swap in from the trigger table. These modules are then loaded from secondary storage. The process of module loading is performed exactly like that of any other Linux Kernel loadable modules. After all relevant modules are loaded into the module library, the adaptation manager issues a *freeze()* to temporarily freeze the transport engine. Upon callback from the transport engine (after proper post-processing of the transport engine), the adaptation manager updates the module handlers to point to the new modules. Note that it is possible that the new set of modules is associated with a new transport logic, and hence the transport logic also needs to be updated (i.e., the event/handler mapping in the transport logic

needs to be updated). As soon as the adaptation manager finishes updating the pointer of the module handlers and mapping in the transport logic, it de-freezes the transport engine. The transport engine can resume consuming events in the event queue when it is executed by the kernel. As we discussed earlier, an important design in *TP is the maintenance of public states such that the new module(s) can inherit the state left behind by the old module(s). *TP currently defines the following public states to be shared across all non-core modules: sequence number, data buffer, pending data, loss information (e.g. scoreboard information), RTT, and rate/window. In this way, the overheads incurred due to module swapping are minimized.

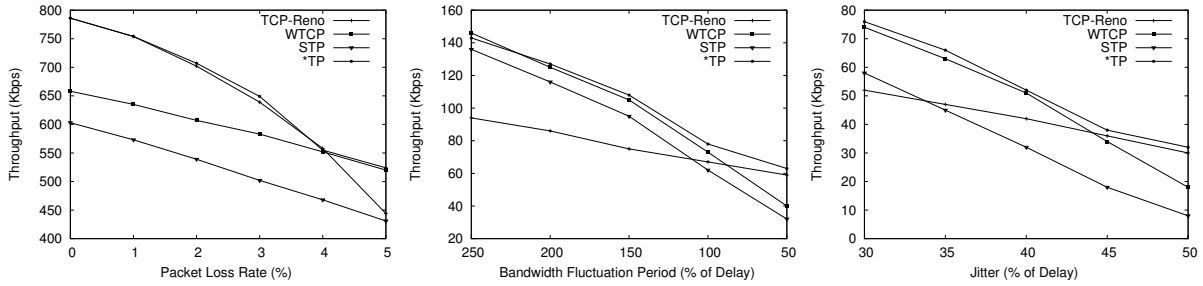
CHAPTER IV

*TP : PERFORMANCE EVALUATION

Now we present case studies to evaluate the performance of the *TP framework in comparison with other production transport protocols. This would help in understanding the benefits obtained by using an ideal transport adaptation solution which can perform run-time re-reconfigurability and modular composition of transport mechanisms. Our objective is to determine how the designed *TP framework is able to adapt its operations to achieve optimal performance. An ideal transport adaptation framework would be able to switch the mechanisms in such a way as to obtain the best performance in any operating environment.

4.1 Lossy WLANs

In the performance evaluation of transport mechanisms, we noted that in WLANs as the loss rate increases, loss-based congestion detection mechanism degrades in performance. This was due to the inability of the mechanism to detect congestion because of insufficient information about the network at high packet loss rates. But we noted that loss-based strategy performs the best among the congestion detection schemes in normal WLAN conditions specified earlier. Hence we use loss-rate as the trigger parameter for the loss-based



(a)WLAN (b)WWAN (c)Satellite network

Figure 12: Performance study of the *TP framework

congestion detection mechanism and the threshold value is set to 2%. When the *TP trigger module detects that the loss rate increases beyond 2% then it switches the loss-based congestion detection to the delay-based congestion detection mechanism which does not suffer too much in lossy environments. We can see from Figure 12(a) that *TP indeed performs well even as the loss rate increases within the WLAN environment. We can see that the slope of the curve corresponding to the throughput of *TP follows the best set of mechanisms for the given conditions. This is because of the capability of *TP to choose and use the best available mechanisms at run-time.

4.2 Bandwidth Fluctuation in WWANs

We observed in Chapter 2.5 that the tuned rate acknowledgment scheme which performs well in WWAN conditions suffers when the bandwidth fluctuation increases. Since bandwidth fluctuations are the norm in both CDPD and 2.5 and 4G wireless wide area networks, an ideal transport adaptation solution should be able to change the acknowledgment scheme when the bandwidth fluctuation increases beyond a certain threshold value. *TP has the capability to accommodate both the self-clocked and tuned-rate acknowledgment schemes and can swap between them depending on the operating conditions. We use bandwidth fluctuation as the trigger for the acknowledgment scheme and the threshold value to be 100% of the delay in the network. We observe from the results in Figure 12(b) that *TP achieves the best performance by swapping the acknowledgment mechanism used when the network conditions degrade. As noted earlier, *TP achieves the best throughput for a given set of transport mechanisms because of its ability to change mechanisms in an intelligent fashion using triggers.

4.3 Jitter in Satellite networks

As noted in Chapter 2.5, both delay-based and inter-packet separation-based congestion detection mechanisms suffer under high delay variations. Although the delay-based scheme performs well in high-loss satellite environments, it has to be replaced by the loss-based

scheme when the jitter increases beyond a threshold. *TP achieves precisely this functionality by swapping the congestion detection mechanisms if the trigger, namely jitter, value is beyond a threshold value. We see from the result in Figure 12(c), that by performing such fine-grained adaptability *TP is able to achieve best performance even in varying network conditions. Here we can see that even as the performance degrades due to the original congestion detection mechanism suffering at high jitter, *TP is able to adapt its behavior to use a different mechanism better-suited for the high jitter conditions.

CHAPTER V

CONVERSATIONAL TRANSPORT PROTOCOLS : PROBLEM DEFINITION

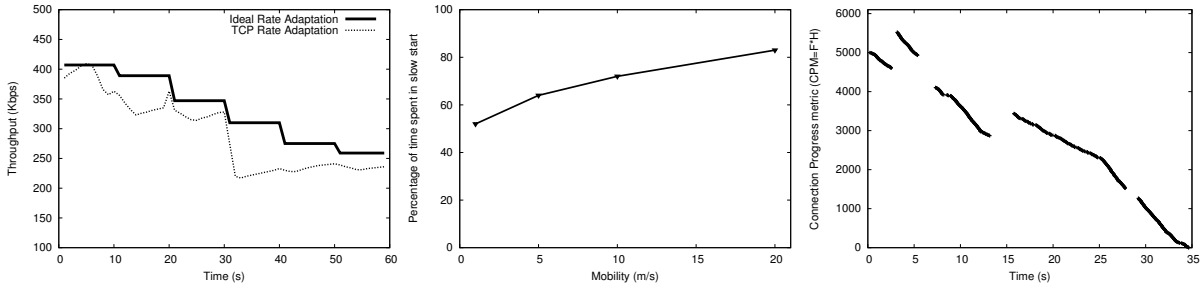
We define *conversational transport protocols* as those that use either the source or the destination (or a combination of the two) as the primary seat of intelligence for the transport layer functionality, and require fine-grained end-to-end communication between the source and the destination to carry out the transport layer operations. Most existing transport layer solutions for wireless ad-hoc networks fall under this category [11],[14], [23], [13], [6].

In this chapter, we focus on highlighting some of the significant drawbacks of using a conversational transport layer protocol for wireless data networks characterized by peer-to-peer communication like ad-hoc networks. In the rest of the dissertation, we consider ad-hoc networks as representative of a wireless data network with peer-to-peer communication. We present the arguments in terms of (i) fine-grained congestion control and associated imprecision, (ii) congestion control behavior during route changes, (iii) connection progress, (iv) reverse path traffic, (iv) impact on MAC layer performance, and (v) impact on routing protocol design.

For many of the discussions, we use *ns2* packet level simulations to substantiate our arguments. Unless otherwise specified, we use a network of 100 nodes distributed in a 1000m x 1000m network. The nodes use default TCP (NewReno) for all sessions.

5.1 End-to-end congestion control

Conversational protocols have the primary seat of intelligence in terms of transport layer functionality residing at the source (or the destination). Hence, congestion control - one of the primary transport layer functions, is performed by one of the end-points. Without loss of generality, we assume that the source performs congestion control for the following arguments.



(a) End-to-end congestion control

(b) Time spent in *slow start* phase

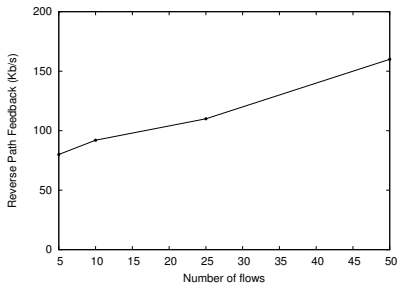
(c) Connection Progress

Figure 13: Identifying shortcomings of conversational transport protocols

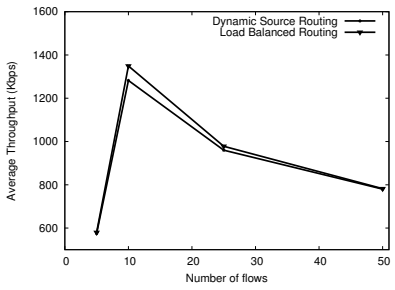
The key goal of congestion control at the source is to adapt to the bottleneck capacity along the end-to-end path. Performing end-to-end congestion control becomes a highly imprecise exercise in an ad-hoc network because of two reasons: (i) for most transport protocols, the source tries to characterize the bottleneck rate remotely, leading to imprecision in the estimates because of end-to-end latency and the highly dynamic nature of the ad-hoc network - both due to mobility and the shared nature of the channel; (ii) even for those protocols that use hop-by-hop congestion control or feedback, estimating the per-flow throughput on a hop to send appropriate feedback back to the source is a non-trivial task, and any feedback latency involved further renders the estimate imprecise.

Figure 13(a) shows the throughput enjoyed by a flow in the network scenario described earlier with 15 TCP flows, and the nodes remaining static. 10 new flows are introduced each at 10s, 20s, 30s, 40s and 50s instances respectively. The figure shows the expected rate adaptation for the flow, along with how TCP adapts its rate. It can be observed that following every major change in the network load, TCP takes considerable time to adapt to the new load in the network. During this adaptation period, the flow's rate is grossly mismatched when compared to the ideal expected rate. Equally interestingly, even during relatively stable network load periods, the flow's rate fluctuates due to the irregularities in the instantaneous available throughput for the flow.

While some of the above observations (e.g. slow to react to network dynamics) are well known aspects of TCP's behavior, note that the observations about the imprecision of the end-to-end congestion control model in a highly dynamic environment (both in terms



(a) Reverse path overhead



(b) Comparison of DSR and LBR

Figure 14: Analyzing the performance of conversational transport protocols

of load and mobility) is generic to any protocol that uses a conversational communication model.

5.2 Route changes

While the earlier argument made was in the context of network dynamics even when the current path is usable, we now consider the effect of path failures and consequent route changes on the congestion control performance of conversational transport protocols. Every time a route changes, a conversational transport protocol has to "speed-up" to the available throughput capacity. Any time the connection spends in the "probe" phase is an under-utilization period as the connection has not reached its ideal throughput.

Figure 13(b) shows, for a single connection, in the earlier described network scenario with nodes moving (10m/s), what percentage of the connection's lifetime is spent in the "slow-start" phase. As can be observed, a considerable portion of the lifetime is actually spent in the probe phase resulting in the under-utilization available resources.

Note that this drawback is a direct consequence of conversational protocols having to operate at a rate determined by the end-to-end path characteristics. This observation will become more evident when we discuss a non-conversational transport layer design in the next chapter.

5.3 Connection progress

In this discussion, we focus on a new metric called the *connection progress metric* (CPM). We define CPM as the product of the amount of data pending to be delivered to the destination, and the distance of the data from the destination. Thus, when the connection is initiated for transferring B bytes of data, and the hop-length between the source and the destination is H hops, the CPM is equal to $B * H$.

We use the CPM to monitor the progress of the connection with time, in the presence of network dynamics. The ideal behavior for the CPM as a function of time should be a decreasing function. Figure 13(c) shows the CPM for a connection in the network scenario defined earlier with nodes moving with a maximum speed of 20m/s. There are three distinct phenomena that can be observed through the representative result:

(i) There are several instances of *path failures* represented by the breaks in the curve. Conversational protocols are more vulnerable to such path failures as the failure of any link on the path (and worse still any link on either of the forward or reverse paths) will disrupt the connection progress.

(ii) Following every instance of a path failure, it takes considerable amount time for the CPM to start decreasing again. An extreme case of such a disruption is if the source and the destination lie in different components of a partitioned network, in which case the connection will not resume progress till the partition is resolved.

(iii) Finally, there are instances of the CPM actually increasing during the progress of the connection. This is due to an increase in the hop-length of the newly constructed path after a route recomputation (say the source has moved away from the destination).

Overall, given the above phenomena, the CPM progress function is far from the ideally desired strictly decreasing trend. We show later that a non-conversational protocol design considerably improves the CPM function.

5.4 Reverse path traffic

One of the defining aspects of a conversational transport protocol is the fine-grained feedback the source needs from the destination. This aspect however imposes a non-trivial reverse

path overhead for the connection.

Note that this property is not restricted to just TCP-like transport protocols. For conversational protocols, the granularity of feedback on one hand directly determines the responsiveness of the transport protocol to network dynamics, and on the other hand determines the overhead imposed by the reverse path traffic. Hence, simply reducing the amount of reverse path traffic is not a solution, and a proper balance has to be always struck.

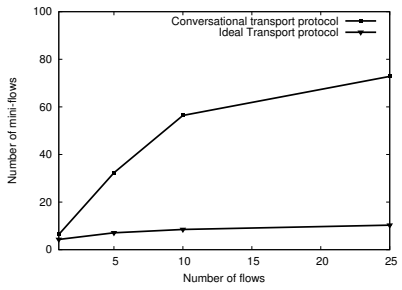
In Figure 14(a) we present the reverse path overheads in the network with different number of flows. The aggregate forward path throughput is approximately 700Kbps. It can be seen that the reverse path traffic is a significant portion of the forward path data throughput and increases with network load.

5.5 Impact on MAC performance

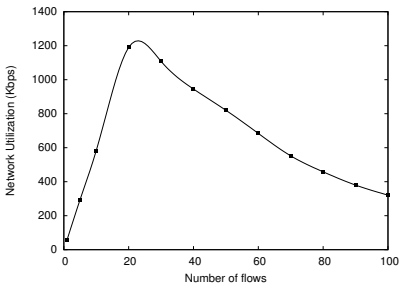
A by-product of the imprecision in the congestion control behavior of conversational transport protocols is the difference between the ideal and the actual number of MAC layer (one hop) flows contending with each other in the ad-hoc network.

Figure 15(a) shows the ideally expected number of MAC layer flows, and the number of such flows actually contending in the network when using TCP. As can be observed, the difference is considerable. While the difference is directly due to the imprecision in the congestion control, the corollary effect of this phenomenon is the negative impact on the MAC layer performance.

Figure 15(b) shows the utilization of a contention based MAC layer such as the IEEE 802.11 CSMA/CA protocol. As can be seen from the bell-shaped curve, the utilization of the MAC protocol will decrease beyond a certain offered load. Hence, for a moderately to heavily loaded ad-hoc network, the imprecision in the congestion control of conversational protocols can actually result in poor MAC layer performance further compounding the problems experienced by connections using the conversational transport protocols.



(a) Comparison of number of mini-flows



(b) Channel utilization for IEEE 802.11 CSMA/CA protocol

Figure 15: MAC Utilization of conversational transport protocols

5.6 Impact on routing protocol design

The final impact of using conversational protocols that we discuss is that on the routing protocol design. Specifically, the behavior of conversational transport protocols can be visualized as a "stream" model where multiple packets can be in transit between the source and the destination both in the forward (data) and reverse directions (ACK). Such a stream model however has a negative impact on the ability of routing protocols to perform true load-balancing. This is because of the fact that due to the shared channel nature of ad-hoc networks, the stream abstraction in reality translates into a stream with a width of multiple-hops. Hence, the number of non-contending routes that can be selected in a practical ad-hoc network environment turns out to be very small.

Figure 14(b) shows the performance when using a default shortest path routing protocol such as DSR, and a load-balanced variation of DSR similar to the one presented in [15]. It can be seen that very minimal advantages are gained through the load-balancing.

CHAPTER VI

NON-CONVERSATIONAL TRANSPORT PROTOCOL (NCTP) : DESIGN AND IMPLEMENTATION

In this chapter, we outline the design of a non-conversational transport protocol (NCTP). Following a high level overview of the basic design of NCTP, we identify key challenges that need to be addressed for the design to be effectively translated to a practical transport layer solution.

6.1 Overview

At a high level, NCTP is a transport layer solution where the seats of intelligence and data are no longer only at the source or the destination. Instead, as the connection progresses, the transport layer responsibility is transferred from one intermediate node to another along the path from the source to destination, until the data to be transferred finally reaches to the destination.

One of the cornerstones of the NCTP design is the transfer of data in *bundles* as opposed to packets. In other words, when a certain amount of data has to be transferred from the source to the destination, the data is broken down into bundles. Each bundle is then transferred *hop-by-hop* till it reaches the destination. The source sends a subsequent bundle onto the path only when the previous bundle has been delivered to the destination. Also, every intermediate hop does not begin to forward a bundle until it has received the entire bundle from the previous hop. An extreme choice for the bundle size is the size of the entire data to be transferred. We assume this simplistic model for the rest of the discussions on NCTP. However, we hasten to add that such a simplistic assumption is obviously not conducive to real-time traffic where *instantaneous throughput* might be more important than the overall throughput, and we revisit the choice of the appropriate bundle size for a connection, and its impact on the performance later.

Note that although NCTP uses the notion of bundles for data transfer, there is no difference in the behavior of the underlying routing and MAC layer protocols in their design: *handling only packets*. The abstraction of a bundle is used only at the transport layer. Thus, a key difference between conversational protocols and NCTP is that every node on the path from the source to the destination explicitly participates in the transport layer functionality of the connection, and the transfer of data is done hop-by-hop from one transport layer module to the transport layer module on the other side of the hop.

Thus, traditional transport layer functionalities such as congestion control, flow control, and reliability, are now handled hop-by-hop as opposed to end-to-end.

Finally, because intermediate nodes participate in the transport layer functionality for the connection, NCTP uses the notion of *ownership transfer*, whereby once data is handed over to an intermediate node, that node assumes the primary responsibility of delivering the data reliably to the destination. NCTP still associates a coarse level secondary responsibility with the original source to prevent any compromise of the end-to-end semantics (e.g. source and destination in the same component of a partitioned network, but the new intermediate node that has assumed ownership is in a different component). However, the secondary responsibility is triggered at a very coarse time granularity through coarse-grained recovery timers.

6.2 NCTP vis-a-vis Drawbacks of Conversational Protocols

Before we proceed to identify some of the obvious challenges that need to be addressed to make NCTP a viable transport layer solution, we briefly delve into how the above design handles the identified drawbacks of conversational protocols.

- Since data is now transferred hop-by-hop as opposed to end-to-end, there is no necessity for performing end-to-end congestion control eliminating the impreciseness of the remote rate estimation process. Instead, congestion control is implicitly performed by the underlying MAC layer to hop-wise flow fairness. While this can affect the end-to-end fairness model achieved, we establish later that the fairness model achieved by

NCTP is no different from that of TCP.

- Since end-to-end congestion control is no longer performed, when route changes occur, no explicit rate estimation process needs to be undertaken. In other words, when a new route is selected, the sender of the data at that point merely has to proceed to send to the new next hop at the rate the underlying MAC will allow.

- Revisiting the CPM metric identified earlier, NCTP can be expected to achieve better connection progress because the progress of the connection is no longer determined by the *path*, and is solely determined by the next hop the data is being transferred over. Furthermore, the transfer of data in its entirety toward the destination better reduces the expected distance between the data location and the destination as we illustrate later.

- Since the conversation in NCTP is restricted solely to one-hop, there is no reverse path traffic that will contend with the forward path data. In fact, as we elaborate later, if a semi-reliable MAC layer protocol such as IEEE 802.11 is used, there is no need for explicit transport layer ACKs even on the single hop.

- Since NCTP connections will have exactly one outstanding one-hop flow on the end-to-end path, the number of such one-hop flows contending in the network is drastically reduced when compared to conversational protocols, thereby enabling the MAC layer performance to scale better.

- Finally, having only one outstanding bundle in the network provides an additional dimension of separation (in time) between flows. In other words, two flows that would have otherwise contended in the spatial domain (because their paths intersect) when using conversational protocols, might no longer contend because their respective bundles traverse the intersection point at different points in time. This additional dimension of separation can enhance the usefulness of a load balanced routing protocol.

6.3 Challenges

While we have established at a high level the operations of NCTP and its expected performance improvement over conversational transport protocols, we now briefly outline the key challenges that need to be addressed to make NCTP a viable solution. In the next section, we describe the mechanisms used in NCTP, and justifications for its design, along with establishing its properties theoretically.

1. *Congestion control:* While NCTP as described above does not require any congestion control, the pitfalls of sending data in bundles and the potential requirements they impose on the congestion control behavior is important. For example, one of the obvious pitfalls of transferring data in bundles is the potential lack of leveraging *spatial reuse* in the network. Thus, the concern is : *Is the lack of leveraging spatial reuse a serious problem, and if yes, how can it be addressed?*

2. *Guaranteed reliability semantics:* While NCTP uses the notion of transfer of ownership of data as it is being forwarded, explicit mechanisms should still be used to guarantee reliability semantics that are no weaker than those provided by conversational transport protocols. For example, *if an intermediate node that has assumed ownership leaves the network, how can the data still be delivered reliably?*

3. *Handling mobility:* One of the potential pitfalls of transferring data in bundles is that when there is a path failure between the location of data and the destination, the entire bundle can be lost. This is in contrast to only a few packets in transit being lost in conversational protocols. *How can this drawback due to mobility be addressed?*

4. *Real-time traffic:* Finally, for real-time applications, instantaneous throughput or the delay experienced on a per-packet basis might be a more important metric than the overall throughput enjoyed by the connection. *How can the bundling strategy be tailored to serve such real-time applications?*

In this section, we first present the key mechanisms used by NCTP, followed by an overview of the protocol operations. We then establish the fundamental throughput capacity

and fairness properties of NCTP through theoretical analysis. In the next chapter, we evaluate NCTP's performance through simulations and compare it against other transport layer strategies.

For purposes of the following discussions, we differentiate between source and sender; and destination and receiver based on whether it is an end-to-end connection or a single hop transfer respectively.

6.4 Mechanisms

6.4.1 Hop-by-hop Congestion control

The inherent principle of a non-conversational protocol is the absence of an end-to-end connection between the source and the destination of communication. This property obviates the need for an end-to-end congestion control algorithm in the NCTP mechanism. Since data is not sent in an end-to-end fashion, the source of data in the case of non-conversational protocols do not need to constantly monitor the end-to-end path for available bandwidth.

Nevertheless NCTP needs to perform congestion control across single hops during the transfer of the bundle. Essentially, congestion control in a single hop scenario is in effect a mechanism to share the wireless shared medium with other mini-flows in the same contention region. The bandwidth sharing on a link level is the function of the MAC layer. We consider IEEE 802.11 MAC protocol which is by far the most popular MAC layer protocol for wireless ad-hoc networks. The IEEE 802.11 MAC layer performs contention resolution and bandwidth sharing using CSMA/CA mechanism. NCTP leverages the IEEE 802.11 medium access control mechanism for performing single hop congestion control. Hence it saves the inefficiencies of conversational end-to-end congestion control mechanisms.

An important distinction that needs to be drawn here is the difference in the handling of bundles at the transport layer, and at the MAC layer. While a bundle is handled as a single entity at the transport layer, when the bundle is handed over to the MAC layer, the MAC layer perceives the bundle as merely a sequence of packets. Hence, for example, when the MAC transmits the first packet of the bundle, it has to again contend for the channel in order to get the opportunity to transmit the second packet of the bundle. This

design provides a clear separation between the transport and MAC functionality, while at the same time alleviating any fairness concerns through channel capture.

6.4.2 Coarse-level end-to-end acknowledgments

Since the goal of this work is to provide the same end-to-end semantics as TCP including that of reliability, NCTP has to be equipped with additional mechanisms to provide end-to-end reliability. NCTP uses end-to-end acknowledgments (ACKs) to provide TCP-like reliability semantics, but the ACKs are sent for bundles as opposed to packets. The destination on receiving a bundle sends an ACK back to the sender informing the receipt of the bundle. Thus, unlike conversational transport protocols that require periodic acknowledgments, the ACKs used by NCTP are at the bundle-level rather than at the packet-level. The sender uses coarse-grained timers for the arrival of acknowledgments from the destination. On expiry of the timer the sender retransmits the bundle to the next-hop along the route to the destination. This ensures that the source always gets confirmation about the receipt of the bundle by the destination.

Coarse-level acknowledgments also address the problem of intermediate nodes getting partitioned from the destination. If an intermediate node with a bundle, gets partitioned from the destination, then the coarse level timer at the source will expire and would prompt the source to send the bundle again towards the destination. Thus partitions do not compromise on the end-to-end reliability semantics provided by NCTP.

The coarse-level acknowledgments are also used to maintain the "single-bundle in transit" condition for each connection. This is achieved because the source sends out the next bundle only after the receipt of the ACK for the previous bundle.

6.4.3 Bundle ownership transfer

Although the source of the connection is the seat of ultimate responsibility for transferring the bundle to the destination, NCTP uses intelligent mechanisms to recover from links breakages.

NCTP uses the concept of transfer of bundle ownership to shift the responsibility of delivering the bundle from one intermediate node to another along the path from the source

to the destination. Thus on the transfer of a bundle from a sender to the receiver, the receiver assumes responsibility to : (i) check for the validity of the route to the destination, (ii) find a route to the destination (if the route changes), and (iii) reliably transfer the bundle to the next hop on the route to the destination.

6.4.4 Splitting files into bundles

As identified in the previous section, mobility impacts the performance of non-conversational protocols. One of the effects of mobility, namely partitioning, is taken care of by using end-to-end acknowledgments as discussed previously. The other effect of mobility is the link breakage caused by movement of the receiver out of the communication range of the sender during the transfer of the bundle. When such a link breakage occurs, the previously transferred packets of the bundle have to be re-transmitted, perhaps on a different route.

To address this issue, NCTP uses smaller sized bundles in environments with higher mobility. The reasoning behind this is as follows : The amount of time spent in transmitting a bundle across a single hop is a function of the bundle size, size of the packets (assumed to be fixed) and the bandwidth across a single hop (is approximately a constant). Thus by reducing the bundle size we can determine the time spent in transmitting a bundle across a single hop. Thus by fixing the bundle size statically based on mobility patterns (say, maximum speed), NCTP addresses the problem of link failure during the transfer of a bundle.

6.4.5 Route Probing

NCTP uses route probing as a mechanism to ensure that the bundle is moving towards the destination. Due to mobility, the route determined by the original source of the bundle might be "invalid" by the time an intermediate node receives the bundle. In order to counter this problem, NCTP uses a route probing mechanism to check whether the route to the destination of the bundle [1] is still valid. If so, the intermediate node forwards the bundle to the next hop. If the route to the destination is invalid, then the intermediate node finds a

[1] Note that the route to the destination is carried by every packet in the bundle since we assume source routing.

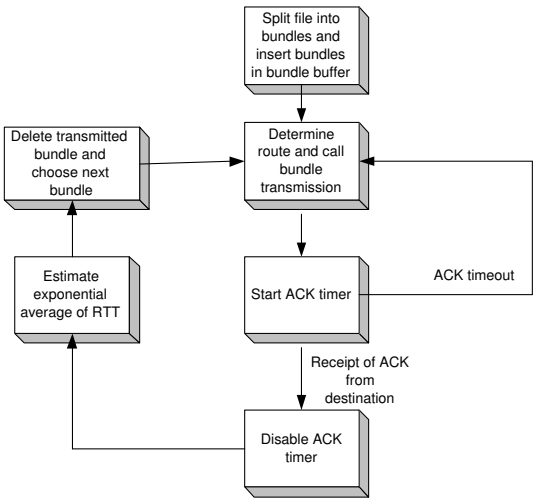


Figure 16: NCTP Source Operations.

new route to the destination, puts the new route on the bundle and forwards it to the next hop as determined by the new route. This ensures that the distance of the bundle from the destination is monotonically decreasing. The route probes are performed pro-actively before the entire bundle is received from the previous intermediate node, so as to not slow down the forwarding of the bundle. If a route probe fails, a new route request is initiated from the intermediate node to the destination.

6.4.6 Single hop reliability

Conversational transport protocols do not assume reliable delivery of packets across the intermediate nodes. This is the reason for the use of end-to-end acknowledgments by conversational protocols. IEEE 802.11 MAC layer is a pseudo-reliable layer in the sense that it tries to deliver a packet reliably to the next for a certain number of attempts [2] and then it reports error. NCTP uses this property of the underlying MAC layer to achieve single hop reliability. It snoops on the MAC layer for information about the reliable transfer of packets in the bundle. This is an optimization technique used to reduce reverse path overhead which is a major drawback in conversational transport protocols.

[2] Retry limit

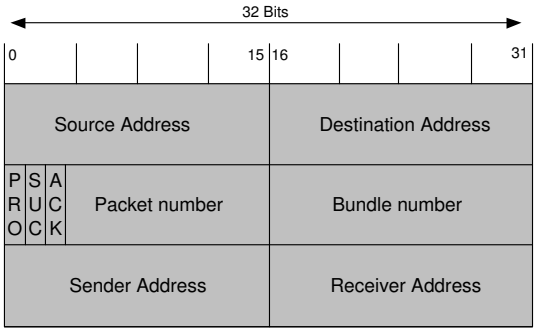


Figure 17: NCTP Packet Header Format.

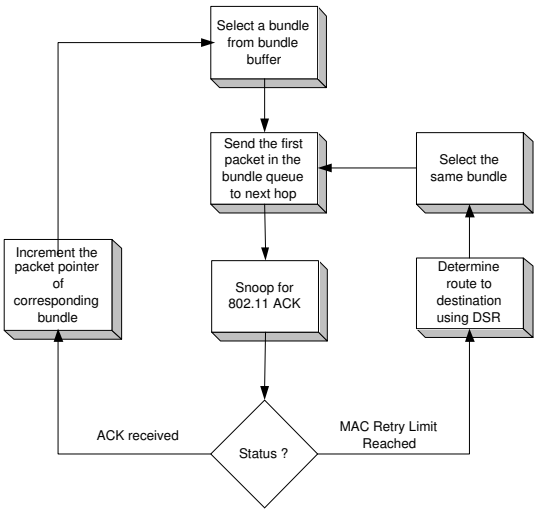


Figure 18: NCTP Bundle Transmission across a link.

6.5 Packet Headers and Protocol Operations

We describe in this section the packet header format used by NCTP for its protocol operations. Figure 17 presents the header format for the NCTP protocol. The NCTP packet header consists of the following fields : (i) Source address, (ii) Destination address, (iii) Sender address, (iv) Receiver address, (v) Bundle sequence number (vi) packet sequence number and set of three flags - PRO, ACK, SUC. The bundle sequence number is used to identify the bundle within a connection. The packet sequence number is used to identify a packet within a bundle. A PROBE-ROUTE packet uses just the source address, destination address, and sets the PRO flag. An ACK packet from the destination uses the source address, destination address, bundle sequence number and the ACK bit is set. A ROUTE-ALIVE packet has the source address, destination address and the SUC flag set.

6.5.1 Source

The operations performed by an NCTP source is illustrated in Figure 16. The source of the transport connection, on receiving a call from the application to deliver a file, splits the file into bundles of specific size (S). The value of S is determined statically based on the maximum speed of the nodes. The source then finds the route to the destination using DSR. After receiving a route to the destination, the source transmits the entire bundle to the next hop. The details of bundle transfer is explained subsequently. After the successful transfer of the bundle to the next hop, the source starts an ACK timer. On reception of the ACK packet from the destination, the source disables the ACK timer, deletes the bundle from its bundle buffer, selects another bundle for transmission from its bundle buffer and performs the above procedure to transmit the bundle. On reception of the ACK from the destination, the source also calculates an exponential average of the round trip time (RTT) for the ACK to arrive after the bundle had been sent. This is calculated from the timer value at the instant of ACK arrival which indicates the time duration between the instant that the source sent the bundle and the time at which it receives the ACK from the destination. Thus the ACK timer doubles up as a means for determining RTT value in addition to ensuring end-to-end reliability semantics. The source uses this RTT estimate for determining the ACK timer. On expiry of the timer, the source resends the bundle using the above procedure.

6.5.2 Intermediate-node

Figure 19 illustrates the set of operations performed by intermediate nodes on the reception of bundles from upstream nodes. Just before the reception of the entire bundle [3], the intermediate node sends a PROBE-ROUTE packet to the destination. The PROBE-ROUTE packet is unicast to the destination using the source route on the packet. On reception of ROUTE-ALIVE message from the destination, the intermediate node sends the bundle using the above describe bundle transmission procedure to the next hop on the route. If a DSR ROUTE-ERROR message is received for the PROBE packet, the intermediate node

[3] On the reception of the (BUNDLE SIZE - 10)th packet

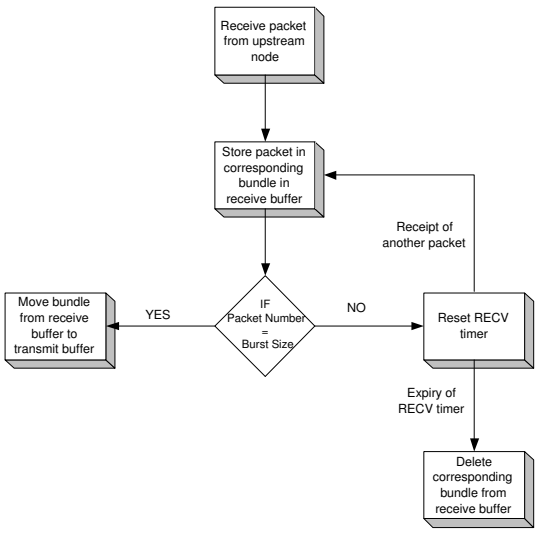


Figure 19: NCTP Intermediate node operations.

recomputes the route to the destination using DSR.

6.5.3 Destination

On successful reception of a bundle intended for the destination, it sends an ACK to the corresponding source of the data bundle. On the reception of a PROBE-ROUTE packet from a node, the destination sends ROUTE-ALIVE message back to the originator of the PROBE-ROUTE packet. The operations performed by the NCTP destination are illustrated in 20.

6.5.4 Bundle transmission procedure

Now we describe how nodes transmit a bundle to the next-hop. The node sends the first packet from the bundle to the next hop. It waits for the arrival of MAC 802.11 ACK before transmitting the next packet in the bundle. Thus the NCTP design uses the MAC level feedback for achieving single-hop reliability. This is a typical example of benefits of cross-layered interaction. The node repeats the procedure until it transmits all the packets in the bundle or it receives an error from the MAC. On the reception of MAC error to deliver packet, the node finds the route to the destination using DSR and starts resending the bundle using the same procedure.

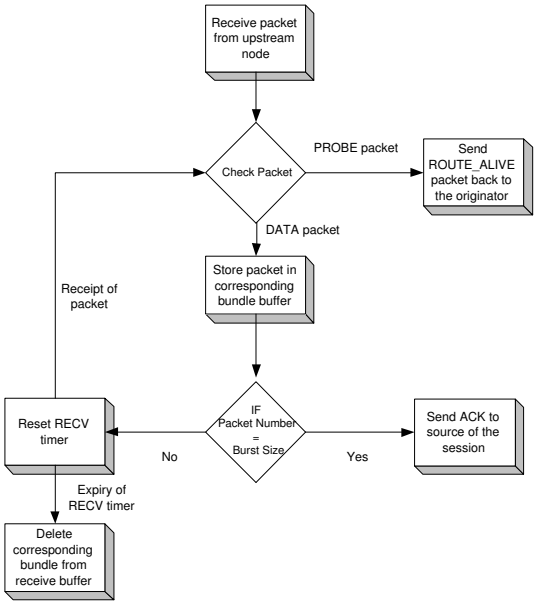


Figure 20: NCTP Destination node operations.

Table 2: Notations used for Analysis

Symbol	Definition
W	Capacity of the channel
k	Total number of contention regions [4].
ρ	Average Contention level
λ	Average per-flow throughput
h_i	Average hop-length of flow f_i
h_{max}	maximum hop-length of flows
h_{av}	average hop-length of the n flows in the network.

6.6 Properties

In the rest of the section, we study the properties of NCTP in terms of throughput capacity and fairness. All aspects of the network model, unless otherwise stated, are as presented in [8].

6.6.1 Throughput capacity

The notation used in the analysis are given in Table 2.

Throughput capacity (γ) is defined as the sum of the throughputs of all the flows in the network.

We consider two models of communication for the analysis namely conversational (C)

and non-conversational (NC) model. Using conversational (C) model of communication, the source pipelines the packets towards the destination across the multiple hops of the network. The non-conversational (NC) model is one in which the source has only one outstanding packet in the network. The source sends the next packet only after the destination receives the previous packet.

First, we consider the throughput achieved using the C model. By definition, at steady state, the number of contending mini-flows contributed by any flow f_i using the C model is equal to the hop-length h_i of the flow. Hence, at steady state, the total number of mini-flows in the network, using C, is given by $n \cdot h_{av}$.

The average contention level using C model is given by,

$$\begin{aligned} \rho(C) &= \frac{\text{No. of mini-flows in the network using the } C \text{ model}}{\text{No. of contention regions in the network}} \\ &= \frac{n \cdot h_{av}}{k} \end{aligned} \tag{1}$$

Hence, we see that the capacity of each contention region is shared by $\frac{n \cdot h_{av}}{k}$ mini-flows on an average.

The throughput of a flow f_i can be obtained by determining the time taken by each transmitted bit by the source to traverse h_i hops in the network. However, in the C model, the source aims to keep the pipe between the destination and itself full by constantly pumping in packets into the pipe. This results in the number of bits in transit at any instant to be approximately equal to the number of hops and hence the cost (time taken by each bit) has to be amortized over the hop length h_i of the flow. This in turn makes the throughput of the flow f_i dependent on the throughput of the mini-flows. Specifically, the throughput achieved using the C model reduces to the throughput of the bottle-neck mini-flow (mini-flow with the minimum throughput) in the path of the flow. Assuming the average contention level to be the same in all the contention regions, all the mini-flows of the end-to-end flow would obtain the same throughput. Hence, we have the throughput of an end-to-end flow in C to be given by,

$$\lambda(C) = \frac{W}{\rho(C)} \tag{2}$$

Substituting for $\rho(C)$, we have,

$$T(C) = \frac{W \cdot k}{n \cdot l_{av}} \tag{3}$$

Hence, we have the throughput capacity using C model as,

$$\gamma(C) = n \cdot T(C) \tag{4}$$

Substituting for $T(C)$, we get,

$$\gamma(C) = \frac{W \cdot k}{l_{av}} \tag{5}$$

Now let us consider the non-conversational (NC) model of communication. By definition, in NC, every flow f_i has only one contending mini-flow at any time instant. Hence the total number of contending mini-flows is the same as the number of flows in the network n. Hence the average contention per contention region using the non-pipelined model is

$$\begin{aligned} \rho(NC) &= \frac{\text{No. of mini-flows in the network using the } NC \text{ model}}{\text{No. of contention regions in the network}} \\ &= \frac{n}{k} \end{aligned} \tag{6}$$

Since in NC it is ensured that there is only one outstanding data unit belonging to a flow at any instant, the throughput achieved by each flow f_i can be calculated by determining the time taken for a bit to to be transmitted from the source s_i to the destination d_i through h_i hops. The time taken for a bit to traverse from the source to the destination is the sum of the time taken for the bit to traverse each hop l_s, s from 1 to h_i, of f_i. Time taken for a bit to traverse through one hop in NC is given as $\frac{\rho(NC)}{W}$. Let us denote the throughput of a s-hop flow as δ_s. We can express the throughput of a s-hop flow as follows :

$$\begin{aligned} \delta_t &= \frac{1}{\text{Time taken for a bit to traverse } s \text{ hops}} \\ &= \frac{1}{\frac{s \cdot \rho(NC)}{W}} \\ &= \frac{W}{s \cdot \rho(NC)} \end{aligned} \tag{7}$$

Given the hop length distribution of the flows to be $p_l(s)$, the total throughput using NC can now be derived as

$$\gamma(NC) = n \cdot \sum_{s=1}^{h_{max}} p_l(s)\delta_s \tag{8}$$

Substituting for τ and C_{NC}, we have,

$$\begin{aligned}\gamma(NC) &= \sum_{s=1}^{h_{max}} \frac{n \cdot p_l(s) \cdot W \cdot s}{s \cdot n} \\ &= \sum_{s=1}^{h_{max}} \frac{p_l(s) \cdot W \cdot s}{s} \end{aligned} \tag{9}$$

Thus, the ratio of the transport capacities of C and NC, ρ, from equations 5 and 9 can be obtained as,

$$\begin{aligned}\rho = \frac{\gamma(NC)}{\gamma(C)} &= h_{av} \cdot \sum_{s=1}^{h_{max}} \frac{p_l(s)}{s} \\ &= (\sum_{s=1}^{h_{max}} s \cdot p_l(s)) \cdot (\sum_{s=1}^{h_{max}} \frac{p_l(s)}{s}) \end{aligned} \tag{10}$$

When the hop lengths are uniformly distributed, we have,

$$p_l(s) = \frac{1}{h_{max}}, \quad \forall s \, \epsilon \, [1, h_{max}] \tag{11}$$

Hence, the average hop length is given by,

$$h_{av} = \sum_{s=1}^{h_{max}} \frac{s}{h_{max}} = \frac{h_{max}+1}{2} \tag{12}$$

Further,we have,

$$\sum_{s=1}^{h_{max}} \frac{p_l(s)}{s} = \sum_{s=1}^{h_{max}} \frac{1}{h_{max} \cdot s} \tag{13}$$

Note that the sum of the harmonic series H_i,

$$H_t = \sum_{i=1}^{n} \frac{1}{i} = \gamma + \psi_0(n+1) \tag{14}$$

where γ is the Euler-Mascheroni constant and $\Psi(x) = \psi_0(x)$ is the digamma function. Thus, we have ρ to be,

$$\rho = \frac{h_{max}+1}{2} \cdot \frac{\gamma + \psi_0(h_{max}+1)}{h_{max}} \tag{15}$$

However, in the asymptotic case of $n \to \infty$, we have $h_{max} \to \infty$ and we have,

$$\lim_{n \to \infty} \sum_{i=1}^{n} \frac{1}{i} = \log(n) \tag{16}$$

Hence, in the asymptotic case, this results in an improvement factor of,

$$\rho = \frac{h_{max}+1}{2} \cdot \frac{\log(h_{max})}{h_{max}} = O(\log(h_{max})) \tag{17}$$

for NC over C. The improvement in throughput in NC comes as a result of it favoring the shorter hop flows at the cost of the longer hop flows, and hence increasing the aggregate throughput capacity of the system.

NCTP model uses the NC model of communication wherein there is only one outstanding packet per flow and hence NCTP gets better throughput performance. We can also see that conversational transport protocols that do not perform any congestion control would be achieving the C model wherein every flow pipelines data along the multiple hops to the destination. Other conversational protocols that use different congestion control mechanisms lie along the spectrum between NC and C model wherein the flow would have more than a single outstanding packet but not as much as the fully pipelined model of C. Ideal versions of conversational transport protocols can achieve the NC model and hence achieve higher throughput gain. Thus we have shown that non-conversational model performs as good as any of the ideal conversational transport protocols.

6.6.2 Fairness

By virtue of transporting the data in bundles and ensuring only one bundle is kept outstanding in the network at any instant, every mini-flow contributed by the end-end flow in the $NCTP$ contends only once in h time slots. Let us initially assume that all the contention regions in the network have the same number of contending mini-flows k. Hence, all constituent mini-flows of a specific flow will obtain the same capacity from contention regions namely $\frac{W}{c}$. Thus, the throughput of an end-end flow λ_{NC} in $NCTP$ can be given by,

$$\lambda(NC) = \frac{1}{\sum_{i=1}^{h} \frac{k}{W}} \tag{18}$$

$$= \frac{W}{k \cdot h} \tag{19}$$

where h is the hop length of the flow. Hence, every end-end flow obtains a throughput (rate) in proportion to its hop length, where the shorter hop flows are biased to improve the aggregate network throughput.

Let us now consider the fairness of the de facto TCP transport protocol. We know that

the throughput ($\lambda(C)$) obtained by a TCP connection, is given approximately by,

$$\lambda(C)\alpha\frac{1}{RTT\sqrt{p}} \tag{20}$$

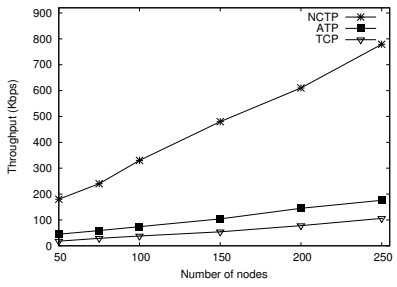
It can be seen that for flows sharing a bottleneck contention region, TCP would bias the shorter hop flows based on their round-trip-time (RTT). and the RTT's of connections are directly proportional to their hop length. Further, it has been shown in [16] that TCP's LIMD paradigm of congestion control adheres to the minimum potential delay fairness model. Observing the fact that NCTP also biases flows in proportion to their hop lengths, we can conclude that the NC model adheres to the minimum potential delay fairness model as well. While we have considered the case of all contention regions having the same number of mini-flows, this can be relaxed to the case of a uniform distribution of the mini-flows in the different contention regions. Even under this condition, its fairly straight-forward to show that the end-end flows will share the network resources in proportion to their hop length, thereby adhering to the minimum potential delay fairness.

CHAPTER VII

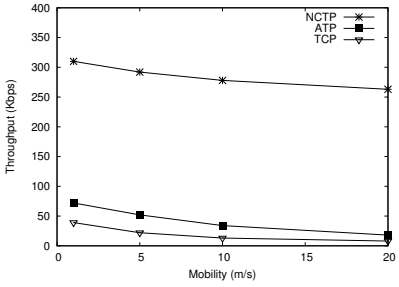
NCTP : PERFORMANCE EVALUATION

7.1 Environment

We use *ns2* network simulator for all our simulations. The setdest tool in *ns2* is used to generate random topologies of different network sizes and number of nodes. The mobility model used for topology generation is the *random waypoint model.*. The source-destination pairs are randomly chosen from the set of nodes in the network. FTP is the application that we use for all the flows in the network. The packets generated are 512 bytes in size in all the simulations. The performance of the proposed NCTP protocol is compared with TCP (window-based conversational transport protocol) and ATP (rate-based conversational transport protocol). The metrics we employ to measure the performance of the new transport protocol are aggregate throughput and normalized standard deviation. The aggregate throughput is measured in kbps and reflects the number of packets successfully received at the destination. The normalized standard deviation measures the standard deviation between the individual flow throughputs in a particular scenario normalized to the average throughput for that scenario. This metric can be thought of as being representative of an unfairness index. Hence the higher the normalized standard deviation, the higher is the degree of unfairness among th flows in the network. All the simulations are run for 60

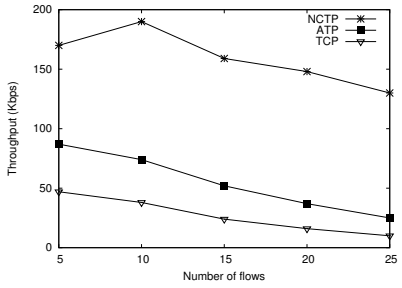


(a) Impact of network size

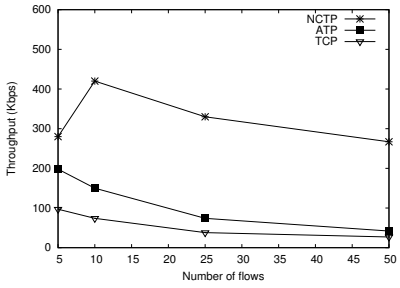


(b) Impact of mobility

Figure 21: Throughput results - I



(a) Impact of load : 50 nodes



(b) Impact of load : 100 nodes

Figure 22: Throughput results - II

seconds. Every flow in the network exists for the entire simulation and each data point on the graph is averaged over 10 simulation runs. We use the following values for the tunable parameters of NCTP: bundle size of 1000 packets for static networks and 500 packets for mobile scenarios; and initial ACK timer of 5 seconds.

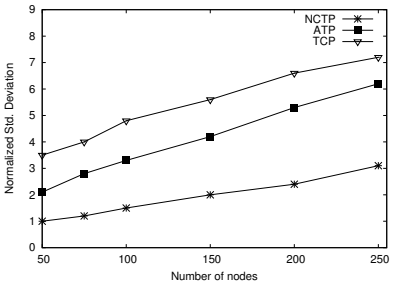
7.2 Macroscopic Results

7.2.1 Impact of network size

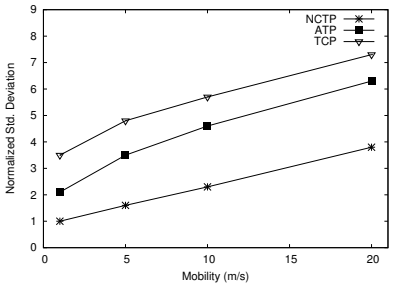
First, we study the impact of the number of nodes on the performance of the three transport protocols. Initially 100 nodes are placed in a 1000m * 1000m topology and then the size of the network is appropriately increased with increasing number of nodes in order to maintain constant node density. We use half of the number of nodes in the network as sources of connection in this scenario.

We observe from Figure 21(a) that the throughput of NCTP is an order of magnitude better than that of both ATP and TCP. This is due to the fact that NCTP which uses the non-conversational model addresses the drawbacks of the conversation approach adopted by TCP and ATP. With increasing network size there is more spatial re-use and hence the throughput increases for all the three protocols. But we can see that NCTP's performance improvement is much better than that of ATP and TCP.

NCTP reduces the number of mini-flows in the network in comparison to both ATP and TCP and hence is able to utilize the network better. Recall from discussions earlier that the MAC utilization decreases with increase in number of mini-flows.



(a) Impact of network size



(b) Impact of mobility

Figure 23: Fairness results - I

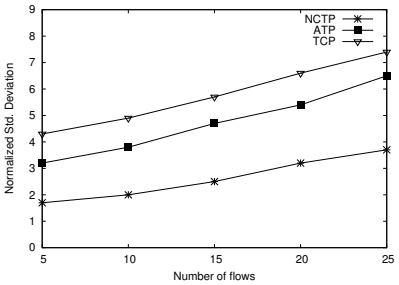
We also study the normalized standard deviation as a metric of global fairness. We can see from Figure 23(b) that NCTP achieves lower normalized standard deviation than both ATP and TCP, indicating better fairness for flows. This can be attributed to two reasons : (i) increase in the average throughput of the flows and (ii) decrease in the unfairness within a contention region by virtue of reducing the number of mini-flows per contention region.

7.2.2 Impact of Load

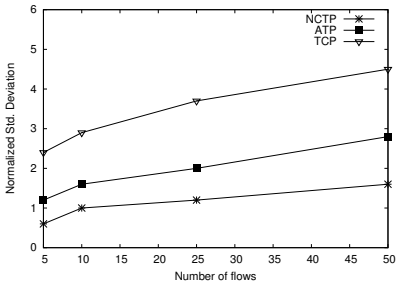
In this scenario the number of nodes is maintained as 50 and 100 in networks of sizes of 1000m * 1000m and 1500m * 1500m respectively. The number of flows in the network is varied to study the impact of load on the transport protocols' performance. As can be seen from Figures 21(b) and 21(c), the peak throughput achieved by NCTP occurs at a higher load than that of ATP and TCP. This is due to the efficient use of the shared medium because of the reduction in the number of mini-flows achieved by NCTP. Since both TCP and ATP contribute more than one mini-flow per end-to-end flow they saturate the network even at low loads. We can see from Figures 23(b) and 23(c) that the normalized deviation increases for all the three protocols but the increase of normalized deviation using NCTP is smaller due to the lower number of mini-flows contributed to the network.

7.2.3 Impact of mobility

In order to compare the effect on mobility on the performance of the compared protocols, we evaluate the three transport protocols under different mobile scenarios. Different values of 1 m/s (pedestrian), 5 m/s, 10 m/s and 20 m/s (vehicular) are used as the maximum



(a) Impact of load : 50 nodes



(b) Impact of load : 100 nodes

Figure 24: Fairness results - II

speed of the nodes. The throughput and fairness achieved using the three protocols is calculated. The results in Figure 21(a) shows that the throughput degradation due to mobility is significantly lower using NCTP than when using ATP or TCP. The reason for this, as discussed earlier, is the independence of NCTP on the existence of a continuous path between the source and the destination. It is also not affected by overheads due to route breakage, route patching and route re-computation as much as the conversational transport protocols. We can see that normalized standard deviation is also lower in NCTP due to the higher aggregate throughput achieved using NCTP.

7.3 Microscopic results

7.3.1 Connection Progress

As discussed earlier, conversational transport protocols have stalls in the connection progress due to mobility. We have compared the connection progress achieved using NCTP and TCP using a mobile scenario where a nodes randomly move in a 1000 * 1000 network with a maximum speed of 20 m/s. We calculate the Connection Progress Metric which is the product of the remaining data to be delivered to the application and the remaining distance the data has to be sent. We can observe from the result in Figure 25, that NCTP achieves much better connection progress compared to a conversational transport protocol such as TCP. We have already analyzed the causes for the bad connection progress using TCP. But we note here that NCTP performs better than TCP because of the reliance only on single hop transfer rather than end-to-end transfer of data. Further link failures on the path from

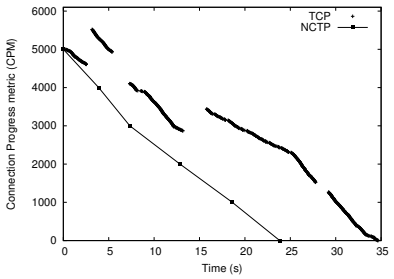
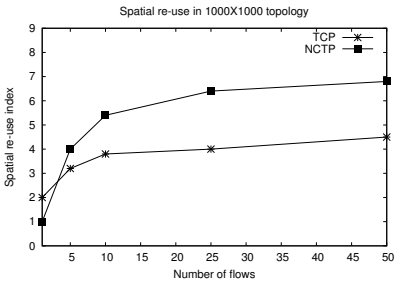
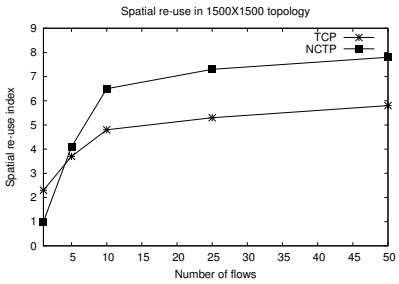


Figure 25: Connection Progress.



(a) 1000 X 1000 network



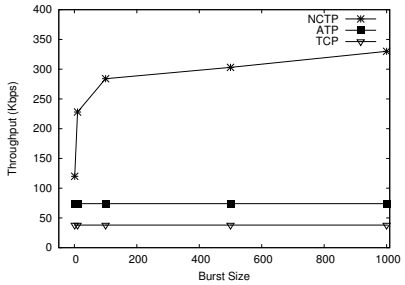
(b) 1500 X 1500 network

Figure 26: Spatial Reuse

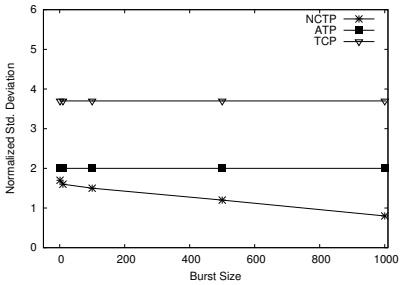
the source to the destination does not cause the connection progress to stall in the case of NCTP. As long as the link over which the bundle is transferred is up, the bundle gets transferred irrespective of the status of the other links on the end-to-end path.

7.3.2 Spatial re-use

We have studied the potential for spatial re-use in typical network scenarios. There might be a concern that a non-conversational transport protocol might not utilize the spatial re-use possible in the ad-hoc network To address the above concern, we have studied the spatial re-use achieved by conversational and non-conversational transport protocol. We define spatial re-use index as the maximum number of simultaneous transmissions possible within the network. We use a 100 and 150 node scenario where we vary the number of flows in the network. From Figures 26 a and 26 b, we can see that non-conversational model used by NCTP indeed achieves much better spatial re-use than conversational transport protocols in almost all load scenarios. NCTP suffers only in the very low load scenario of 1 flow in 100 and 150 node networks which is not a practical scenario. Thus we show that NCTP



(a) Throughput



(b) Fairness

Figure 27: Burst Size

indeed utilizes spatial re-use better than TCP.

7.3.3 Bundle-Size

We analyze the impact of burst-size on the performance of NCTP. We use a 100 node network with 50 flows and set the burst-size value to 1, 100, 500 and 1000 packets and study the throughput and fairness performance of NCTP. The results are plotted in Figure 27. We observe that as the burst size decreases from the default value of 1000 the throughput starts decreasing and is minimum at a burst-size of 1. But the absolute values for throughput is still greater than that of ATP and TCP. The degradation in performance on decreasing throughput can be attributed to the amount of reverse path feedback and also on the dependence on ACKs for the transmission of the next bundle. At higher burst sizes, this overhead is amortized over larger number of packets. But at burst size value of one, even though the number of mini-flows in the network is the same as that of the default burst size, here every packet has an ACK packet overhead and the forward path does not have a packet in transit when the ACK packet is traveling from the destination to the source. We also observe that the normalized standard deviation also decreases as the burst size decreases. This is also due to the reduction in the average throughput of the flows.

CHAPTER VIII

CONCLUDING REMARKS

In this work, we consider the issues that arise with the two major properties of wireless data networks namely increased heterogeneity of wireless network technologies and increased peer-to-peer communication among mobile hosts.

We address the problem of transport adaptation in heterogeneous wireless data networks. Specifically, we answer the questions relating to the ideal nature and granularity of transport adaptation by performing a systematic evaluation of the transport mechanisms for different transport functionalities in various wireless network environments. We prove that an ideal adaptation solution should be able to change mechanisms at a granularity finer than normal interface handoffs. The change in mechanisms should happen even as the network characteristics change within a single wireless network. We also design and implement a runtime adaptive transport layer framework, *TP, which accommodates the requirements of adaptation determined by the performance evaluation.

With respect to the peer-to-peer nature of wireless network communication, we look at an alternative paradigm for communication in ad-hoc networks namely a non-conversational transport protocol. We show how the non-conversational model addresses the shortcomings of legacy conversational model of communication. We design a new transport layer protocol which is an instantiation of the studied paradigm and also addresses certain practical issues. We show through packet level simulations that NCTP protocol has significant performance benefits in terms of throughput benefits, fairness and connection progress.

REFERENCES

[1] AKAN, O. and AKYILDIZ, I., "Atl: Adaptive transport layer for next generation wireless internet," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 802–817, June 2004.

[2] AKYILDIZ, I., MORABITO, G., and PALAZZO, S., "TCP-Peach: A new congestion control scheme for satellite IP networks," *ACM/IEEE Transactions on Networking*, vol. 9, pp. 307–321, June 2001.

[3] BALAKRISHNAN, H. and KATZ, R., "Explicit loss notification and wireless web performance," in *Proceedings of IEEE GLOBECOM*, (Sydney, Australia), Nov. 1998.

[4] CASETTI, C., GERLA, M., MASCOLO, S., SANSADIDI, M., and WANG, R., "Tcp westwood: End-to-end congestion control for wired/wireless networks," *Wireless Networks Journal*, vol. 8, pp. 467–479, 2002.

[5] CONRAD, P., AMER, P., TAUBE, M., SEZEN, G., IREN, S., and CARO, A., "Testing environment for innovative transport protocols," in *Proceedings of MILCOM '98*, Oct. 1998.

[6] D.KIM, C. and CHOI, Y., "TCP-BuS : Improving TCP performance in wireless ad hoc networks," *Journal of Communications and Networks*, vol. 3, pp. 175–186, June 2001.

[7] FU, C. P. and LIEW, S. C., "Tcp veno: Tcp enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 802–817, Feb. 2003.

[8] GUPTA, P. and KUMAR, P. R., "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, Mar 2000.

[9] HANDLEY, M., FLOYD, S., PAHDYE, J., and WIDMER, J., "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM*, (Stockholm, Sweden), Aug. 2000.

[10] HENDERSON, T. and KATZ, R., "Trasnport protocols for internet-compatible satellite networks," *IEEE Journal on Selected Areas in Communications(JSAC)*, vol. 17, pp. 345–359, Feb. 1999.

[11] HOLLAND, G. and N.VAIDYA, "Analysis of TCP performance over mobile ad hoc networks," *ACM Wireless Newtorks*, vol. 8, pp. 275–288, March 2002.

[12] HSIEH, H.-Y. and SIVAKUMAR, R., "On using the ad-hoc network model in cellular packet data networks," in *Processings of ACM MOBIHOC*, (Lausanne, Switzerland), 2002.

[13] J.LIU and S.SINGH, "ATCP : TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 1300–1315, July 2001.

[14] K. Chandran, S.Raghunathan, S. and Prakash, R., "A feedback based scheme for improving TCP performance in Ad-Hoc wireless networks," in *ICDCS*, (Amsterdam), May 1998.

[15] K. Sundaresan, H.-Y. H. and Sivakumar, R., "IEEE 802.11 over Multi-hop Wireless Networks: Problems and New Perspective," *Ad Hoc Network*, vol. 2, pp. 109–132, April 2004.

[16] K.-W. Lee, T. K. and Bharghavan, V., "A comparison of end-to-end congestion control algorithms: The case of AIMD and AIPD," in *Tech. Rep. Univ. Illinois, Chicago, IL*, 2000.

[17] L. Brakmo, S. O. and Peterson, L., "Tcp vegas: New techniques for congestion detection and avoidance," in *In Proceedings of the SIGCOMM '94 Symposium*, pp. 24–35, August 1994.

[18] Nandagopal, T., Kim, T.-E., Gao, X., and Bhargavan, V., "Achieving MAC Layer Fairness in Wireless Packet Networks," in *Proceedings of ACM MOBICOM*, Aug 2000.

[19] R.Rejaie, M.Handley, and D.Estrin, "Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *INFOCOM*, Mar. 1999.

[20] Schmidt, D., Box, D., and Suda, T., "ADAPTIVE: A dynamically assembled protocol trasnformation, integration and evaluation environement," *Concurrency:Practice and Experience*, vol. 5, pp. 269–286, June 1993.

[21] Sinha, P., Nandagopal, T., Venkitaraman, N., Sivakumar, R., and Bharghavan, V., "WTCP: A reliable transport protocol for wireless wide-area networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.

[22] Stemm, M. and Katz, R., "Vertical handoffs in wireless overlay networks," *Mobile Networks and Applications*, vol. 3, no. 4, pp. 335–350, 1998.

[23] Sundaresan, K., Anantharaman, V., Hsieh, H.-Y., and Sivakumar, R., "ATP: A Reliable Transport Protocol for Ad-hoc Networks," in *Proceedings of ACM MOBIHOC*, Jun 2003.

[24] Wong, G., Hiltunen, M., and Schlichting, R., "A configurable and extensible transport protocol," in *INFOCOM*, pp. 319–328, Apr. 2001.