Addressing Network Heterogeneity and Bandwidth Scarcity in Future Wireless Data Networks

A Dissertation
Presented to
The Academic Faculty

by

Hung-Yun Hsieh

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

July 2004

Addressing Network Heterogeneity and Bandwidth Scarcity in Future Wireless Data Networks

Approved by:

Professor Raghupathy Sivakumar, Advisor

Professor Douglas M. Blough

Professor Nikil S. Jayant

Professor Samit Soni
(College of Management)

Professor Hsien-Hsin S. Lee

Date Approved: July 8, 2004

To my family.

ACKNOWLEDGEMENTS

I would like to first thank my advisor, Prof. Raghupathy Sivakumar, for his unflagging support and guidance during my dissertation study. This work would not have been possible without all the insightful and scintillating discussions with him. Prof. Sivakumar leads me into the world of networking research, and acts as an excellent role model of a good researcher. He is a mentor and a friend.

I would also like to thank Profs. Nikil S. Jayant, Douglas M. Blough, Hsien-Hsin S. Lee, and Samit Soni for serving on my dissertation committee and giving valuable opinions during the preparation and presentation of this dissertation. In addition, I would like to thank Prof. Ian F. Akyildiz for his enlightening suggestions during my proposal examination that help improve the quality of this dissertation. I also want to thank Profs. Mostafa H. Ammar and Henry L. Owen for their feedbacks during my proposal and qualifying examinations.

My gratitude extends to the present and past members of the GNAN research group. I cherish the great opportunity of working with them. Special thanks go to Karthik, Vaidy, Kyu-Han, Aravind, and Yujie for many interesting and refreshing discussions. I also thank Yannick, Ram, Seung-Jong, Tae-Young, and Ashraf for their friendship and assistance.

Last but not least, I would like to thank my family for always being there for me. Their love, encouragement, and belief help me go through the ups and downs of this journey. To them, I dedicate this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

To provide mobile hosts with seamless and broadband wireless Internet access, two fundamental problems that need to be tackled in wireless networking are transparently supporting host mobility and effectively utilizing wireless bandwidth. The increasing heterogeneity of wireless networks and the proliferation of wireless devices, however, severely expose the limitations of the paradigms adopted by existing solutions. In this work, we explore new research directions for addressing network heterogeneity and bandwidth scarcity in future wireless data networks. In addressing network heterogeneity, we motivate a transport layer solution for transparent mobility support across heterogeneous wireless networks. We establish *parallelism* and *transpositionality* as two fundamental principles to be incorporated in designing such a transport layer solution. In addressing bandwidth scarcity, we motivate a cooperative wireless network model for scalable bandwidth utilization with wireless user population. We establish *base station assistance* and *multi-homed peer relay* as two fundamental principles to be incorporated in designing such a cooperative wireless network model. We present instantiations based on the established principles respectively, and demonstrate their performance and functionality gains through theoretic analysis, packet simulation, and testbed emulation.

CHAPTER 1

INTRODUCTION

The Internet has gradually evolved to become an *omninet* where all computer systems and information devices are interconnected in synergy to form an omnipresent network. While the backbone of the Internet remains wired predominantly, wireless networking has in particular become the driving force for such an evolution. Wireless networking extends the reach of the Internet where wired connections are not possible or desirable. More importantly, it enables ubiquitous network access, allowing mobile hosts to be connected to the global infrastructure wherever they are. To provide mobile hosts with seamless and broadband wireless Internet access, however, wireless networking faces two fundamental challenges in terms of transparently supporting host mobility against the dynamics of mobile hosts, and effectively utilizing wireless bandwidth against the characteristics of wireless links. Although the two problems have been the focus of many research endeavors, recent trends in the development of the Internet have brought about changes that existing approaches fail to address.

The first trend is the increasing heterogeneity of wireless networks due to the proliferation in the number of wireless access technologies available [15, 38, 109, 110, 111]. These heterogeneous wireless access networks typically have different coverage areas, and exhibit very diverse characteristics in terms of data rates, latencies and loss rates. Existing approaches proposed for host mobility rely on network support to provide mobile hosts with seamless handoffs in terms of minimizing the handoff latency and transmission anomalies such as packet duplicates, losses, or reordering during handoffs [20, 23, 25, 88]. These approaches, however, need to be tailored to the specificity of the network in consideration, and hence cannot scale with the increasing heterogeneity of wireless networks. More importantly, when mobile hosts move across heterogeneous wireless networks, the ability to provide seamless handoffs between access points accounts only for a part of the problems involved. Functionalities such as network handoffs, protocol handoffs, server migration, and bandwidth aggregation need to be supported to provide mobile hosts with truly *untethered* network

access. We elaborate on the requirements of these functionalities for supporting transparent host mobility in Chapter 2.

The second trend is the increasing scarcity of wireless bandwidth due to the proliferation in the number of wireless devices connected. Conventional wireless networks [45, 50, 109, 110] adopt a cellular network architecture where mobile hosts in the same cell share the resource provided by the base station (or access point) for network access. In such a network architecture, in any given time slot (or frequency band/code sequence) only one mobile host can access the channel for utilizing the wireless bandwidth. While a considerable body of research has focused on improving the utilization of wireless bandwidth by addressing the impairments of the underlying channel and allowing a more efficient use of the bandwidth [28, 67, 70, 97], the performance improvement is shadowed by the performance degradation due to the increasing user population. The only avenue for improving the data rate is to decrease the coverage area per base station, thus reducing the number of users served [22, 89]. However, the drawback is the high infrastructure cost involved in deploying a large number of base stations and the associated distribution networks that cannot scale with the growth of mobile hosts. We elaborate on the limitations of existing cellular wireless networks in achieving scalable bandwidth utilization in Chapter 6.

In this work, we consider the fundamental problems of host mobility support and wireless bandwidth utilization in the context of the increasing network heterogeneity and bandwidth scarcity in future wireless networks. We identify the limitations of the paradigms adopted by existing approaches in addressing the two problems, and then explore new research directions that can provide scalable solutions with the increasing heterogeneity of wireless networks and the proliferation of wireless devices. In Part I, we focus on the problem of transparent mobility support across heterogeneous wireless networks. Chapter 2 elaborates on the problem and why related work fails to provide the desired solution. Chapter 3 motivates a solution based on the transport layer protocol that does not rely on the support from the underlying network infrastructure. We explain, however, the drawbacks of using existing transport layer protocols "as-is" in addressing the problem. Chapter 4 and Chapter 5 establish two fundamental principles called *parallelism* and *transpositionality* that need to incorporated in designing such a transport layer solution. We propose transport layer protocols based on the established principles and show that they can support transparent host

mobility and exhibit resilience to the heterogeneity of wireless networks. In Part II, we focus on the problem of scalable bandwidth utilization with wireless user population. Chapter 6 elaborates on the problem and why related work fails to provide the desired solution. Chapter 7 motivates a solution based on the peer-to-peer network model for improving the capacity of existing wireless networks. We present, however, the drawbacks of using the existing peer-to-peer network model "as-is" in addressing the problem. Chapter 8 and Chapter 9 establish two fundamental principles called *base station assistance* and *multi-homed peer relay* that need to be used in tandem with the peer-to-peer network model. We present instantiations of the established principles and show that they can provide mobile hosts with significant improved data rates and exhibit better scalability with the proliferation of wireless devices. Finally, Chapter 10 discusses the relation between the two solutions and concludes the dissertation.

PART I

Network Heterogeneity

CHAPTER 2

HOST MOBILITY ACROSS HETEROGENEOUS NETWORKS

Wireless networks enable untethered communication, allowing hosts to be connected to the network during the course of mobility. While the problem of supporting mobility transparently as the host changes its *point of attachment* has been under active research for years, the increased heterogeneity of wireless networks brings forth new challenges not addressed by existing approaches. In this chapter, we first describe the problem and challenges of supporting host mobility across heterogeneous wireless networks. We then discuss why related work cannot effectively provide the desired solution with resilience to the increased network heterogeneity. Finally, we present an outline of the solution proposed in this work.

2.1 The Problem

The tremendous growth in the number of mobile Internet users has been accompanied by the equally staggering increase in the number of wireless access technologies. A mobile user today can choose from a myriad of options for Internet access including global area [38, 60], wide area [45, 109], local area [50, 110], and personal area [15] wireless networks. Several new wireless access technologies currently being developed and standardized are expected to add to the list in the near future [57, 58].

A key reason behind the diversity of the wireless access technologies is the fundamental performance tradeoffs they exhibit in terms of network capacity, coverage area, and transmission power. For example, the WiFi WLANs offer much higher data rates than the 3G WWANs, but suffer from significantly smaller coverage areas. The Bluetooth technology does not offer as high data rates as the WiFi technology, but saves on the power consumption required for wireless access. Therefore, mobile hosts are increasingly being equipped with multiple interfaces providing access to different wireless networks depending on the availability of the networks and the characteristics of the device and application used. Such a multi-homed mobile host during the course of mobility may encounter several wireless networks and need to migrate from one network to another. Several reasons for

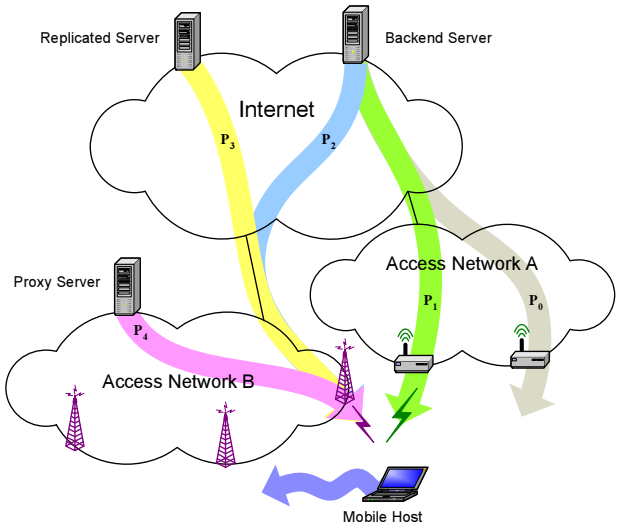


Figure 1: Host Mobility across Heterogeneous Networks

handoffs between heterogeneous networks during a connection include: moving out of the coverage area of the current network, having access to a new network providing better performance (e.g. higher data rates or better cost effectiveness), and switching between overlay networks because of variations in offered services such as data rates and jitters.

Existing work proposed for supporting mobility focuses on providing mobile hosts with seamless handoffs in terms of minimizing the handoff latency and transmission anomalies such as packet duplicates, losses, or reordering during handoffs. When mobile hosts move across heterogeneous wireless networks, however, the ability to provide seamless handoffs between access points accounts only for a part of the problems that need to be considered. We use Figure 1 as an illustration to show different problems that may arise for providing mobile hosts with transparent mobility support across heterogeneous wireless networks.

Consider a scenario where a multi-homed mobile host on the move connects to the Internet through access network *A*, and initiates a file download from a backend server following path P_0. When the mobile host moves into the coverage of another access point inside network *A*, it undergoes a *horizontal handoff* from P_0 to P_1. As the mobile host keeps moving, it may move out of the coverage of network *A* and migrates to another network *B* that it has access to. It is also possible that the mobile host is still within the reach of network *A*, but issues a handoff to network *B* for achieving a higher download speed. In either case, the mobile host undergoes a *vertical handoff* between

heterogeneous wireless networks from P_1 to P_2. Since heterogeneous wireless access technologies are involved during the handoff, the mobile host needs to tackle not only the potential packet re-ordering and loss problems as in the horizontal handoff, but also the change in the characteristics of the wireless links involved (such as loss rate increase) in order to enjoy the optimal performance provided by network *B*. In addition, if network *B* belongs to a different autonomous domain from network *A*, the mobile host may also undergo *server migration* from P_1 to P_3. Server migration may be desirable if the mobile host, through the new wireless interface, has access to a replicated server that provides better performance such as higher throughput and shorter round-trip time. Server migration may also become necessary if the original server is provided by network *A* (e.g. proxy server) that enforces ingress filtering [33] to block access from any network but *A*. (For example, consider the scenario when the mobile host in the reverse direction handoffs from P_4 to P_1.) Note that when the mobile host moves into the coverage of network *B*, it may still have access to network *A*. To achieve a higher download speed, the mobile host may choose to use both wireless interfaces for enjoying *bandwidth aggregation*. Bandwidth aggregation can be provided between the mobile host and the backend server involving P_1 and P_2. It can also be provided between multiple replicated servers and the mobile host involving P_1 and P_3 (or P_4) for the aforementioned reasons necessitating server migration.

Therefore, it is clear that the problem of supporting transparent host mobility across heterogeneous wireless networks is not limited to minimizing the latency and anomalies occurred during handoffs. Functionalities such as protocol changes during handoffs, server migration, and bandwidth aggregation need to be supported to address the constraints imposed by the wireless channel and provide mobile hosts with truly *untethered* network access.

2.2 Related Work

When a mobile host undergoes a handoff, its network address may change as a result of change in the point of attachment to the Internet, thus causing existing connections to break. Mobile IP [82] has been proposed to solve this problem by introducing different mobility agents to support connection redirection. While straightforward, Mobile IP introduces new problems such as route optimization, reverse tunneling, and firewall traversing [119]. More importantly, Mobile IP increases the latency

involved in the handoff. It has been demonstrated that when Mobile IP is used for handoffs across heterogeneous wireless networks belonging to different administrative domains (such as GPRS and WLAN), the latency involved in registering with the home agent can be prohibitive [93]. Various micro-mobility management schemes have been proposed to reduce the handoff latency when handoffs occur within an administrative domain [20, 23, 24, 25, 88]. However, these approaches rely on the provision of handoff crossover points or mobility agents, and coordination between participating access points. They hence will fail to function or become ineffective when multiple administrative domains with minimal or no coordination are involved during handoffs.

Recently, several approaches have been proposed to integrate heterogeneous wireless networks and hide the network heterogeneity from mobile hosts [4, 19, 31, 63, 80, 92, 113]. For example, in [31] a tightly coupled interworking architecture is proposed to integrate HIPERLAN and UMTS to achieve fast and seamless handoffs. In this architecture, the HIPERLAN access network is connected to the UMTS core network and under the administration of the UMTS operator. Hence mobility agents provided by the UMTS network can be used to facilitate handoffs between HIPERLAN and UMTS. In [113], the authors propose a MIRAI architecture to connect all various access networks to a common core network such that AAA (authentication, authorization, and accounting) and mobility can be managed in a homogeneous fashion. In [63] a communication gateway is proposed to shield the mobile user from the heterogeneity of wireless access technologies and achieve infrastructure-independent wireless access. While these approaches integrate heterogeneous wireless networks to be managed by one administrative domain, the effectiveness of the solutions is limited by the specificity to the networks they are designed for. They hence do not provide a scalable solution that is resilient to the increasing network heterogeneity in future wireless networks. Moreover, as we identified in Section 2.1, minimizing the handoff latency and anomalies accounts for only a minor subset of problems that need to be tackled for support transparent host mobility. Failure to provide for handoffs to protocols that are optimized to the target network characteristics renders such solutions suboptimal at best.

Therefore, related work for supporting mobility not only fails to provide a scalable solution against the trend of increasing network heterogeneity, but also fails to provide a comprehensive solution for enabling transparent host mobility.

2.3 Solution Outline

The goal of this work is to propose a solution that, while providing mobile hosts with transparent mobility support, is immune to the increased heterogeneity in future wireless networks. The proposed solution thus should include the following features: (i) *Resilience to network heterogeneity:* The proposed solution should not use any entity that is specific to the network in consideration, or rely on any support from the underlying network. (ii) *Support for seamless handoffs and server migration:* The proposed solution should support seamless end-point handoffs either at the access point or at the server. A seamless handoff not only supports the change of network addresses, but also minimizes the handoff latency and transmission anomalies. More importantly, it allows for the use of network-specific protocols for achieving optimal performance in individual networks. (iii) *Provision for bandwidth aggregation:* The proposed solution should provide mobile hosts with the ability to use multiple network resources simultaneously for achieving accelerated performance. It should support both point-to-point and multipoint-to-point bandwidth aggregation.

Toward this end, we propose a purely end-to-end solution that involves only the end hosts, without the need for any mobility agent and support from the underlying network. Specifically, we propose a transport layer protocol that does not require any change from the application, except for specifying options to use the added functionalities supported by the proposed transport protocol. While several transport layer protocols have also been proposed for supporting host mobility [90, 101], their solutions are limited to handling address changes and access-point handoffs, and hence cannot be used to provide mobile hosts with all desired functionalities during mobility. We establish in this work two fundamental principles that need to be incorporated in designing transport layer protocols for supporting host mobility in heterogeneous wireless networks, namely *parallelism* and *transpositionality*. The principle of parallelism allows a transport protocol to leverage resource multiplicity, while the principle of transpositionality allows a transport protocol to address resource disparity. We present how these two principles can be incorporated in existing transport layer protocols with minimal changes, and in the process arrive at a new protocol that can effectively support host mobility in heterogeneous wireless networks.

We start in Chapter 3 with two alternate end-to-end solutions: application layer and transport

layer approaches. We first discuss why application layer approaches cannot effectively tackle network heterogeneity. We then use queueing-theoretic analysis to substantiate the argument, and show that transport layer approaches have performance benefits over application layer approaches. We hence motivate transport layer solutions to address the problem of network heterogeneity. In Chapter 4, we focus on the principle of parallelism. We first present the key elements in designing a transport layer protocol with parallelism support, and then show how TCP (Transmission Control Protocol) [86], the transport layer protocol predominantly used in the Internet [36, 106], can be extended to incorporate the principle of parallelism, called pTCP (parallel TCP). We present the detailed protocol operations of pTCP and show through network simulations its performance benefits. In Chapter 5, we focus on the principle of transpositionality. We first discuss why parallelism alone cannot address the problem of network heterogeneity, and then show how TCP, a sender-centric transport layer protocol, can be transposed to become a receiver-centric protocol called RCP (Reception Control Protocol). We present the detailed protocol operations of RCP and show through network simulations its TCP-friendliness and performance benefits. Finally, we combine the principles of parallelism and transpositionality and propose a transport layer protocol called R^2CP (Radial Reception Control Protocol) for supporting transparent host mobility across heterogeneous wireless networks. We present the detailed protocol operations of R^2CP, and use testbed emulation to show its performance benefits in providing a truly comprehensive solution for host mobility that is resilient to network heterogeneity in future wireless networks.

CHAPTER 3

TRANSPORT LAYER PROTOCOL

An ideal solution to address network heterogeneity is an infrastructure independent one that does not rely on any entity specific to the network in consideration. End-to-end approaches involve only the end hosts of the connection, and hence can be made resilient to changes in the network infrastructure. The application layer and the transport layer are the two end-to-end layers in the TCP/IP reference model [104]. In this chapter, we motivate why the transport layer is a better place for supporting host mobility across heterogeneous wireless networks.

3.1 Availability of Multiple Pipes

As we discussed in Section 2.1, several key functionalities that need to be supported for providing multi-homed mobile hosts with transparent mobility across heterogeneous wireless networks include seamless (access network) handoffs, server migration, and bandwidth aggregation. While these functionalities apply to different network scenarios, a common theme behind them is the existence of multiple pipes[1] between the server and the mobile host. In Figure 1, for example, P_1 and P_2 co-exist during network handoffs, P_2 and P_3 co-exist during server migration, and finally P_1 and P_3 (or P_4) co-exist during bandwidth aggregation. Much as the origin and the nature of these pipes vary with different scenarios, to provide mobile hosts with transparent mobility support, an invariant is for the connection to operate seamlessly over multiple pipes for as long as they exist.

TCP is the transport layer protocol predominantly used in the Internet [36, 106]. It provides reliable and sequential data delivery from the source to the destination, and uses a window-based mechanism to perform congestion control and flow control [7, 86]. TCP is designed for operating over a single path between the source and the destination, and hence it assumes the first-in-first-out (FIFO) style of packet delivery. Out-of-order arrivals are considered as an indication of packet

[1] A pipe is an end-to-end path terminated by individual network interfaces between the end hosts. However, we use the terms *pipe* and *path* interchangeably in this work.

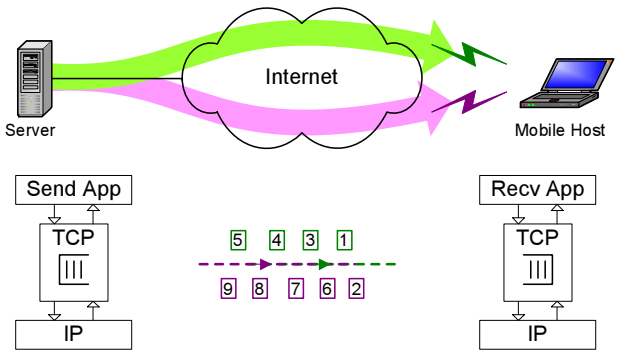


Figure 2: TCP over Multiple Pipes

losses, causing TCP to cut down its window and perform congestion control. While such a single-path design allows TCP to fast detect and recover from losses, it prevents TCP to operate effectively over multiple paths. Consider a mobile host that maintains a TCP connection with the server for Internet access. When the mobile host moves across networks and is exposed to multiple pipes (due to, say, network handoffs or bandwidth aggregation), its TCP connection will react adversely and suffer from performance degradation. As we show in Figure 2, the fact that packets between the server and the mobile host traverse multiple paths can cause significant out-of-order arrivals at the mobile host and trigger TCP's window cutdown.

There are several approaches proposed in different contexts that allow TCP to operate over multiple paths through packet scheduling and reordering at the network or link layers [2, 84, 99]. However, these approaches are applicable only to homogeneous network environments due to their assumptions on the characteristics of paths involved. More importantly, as we mentioned before, lower layer approaches suffer from the susceptibility to the wireless technology and network infrastructure involved. A different class of approaches, on the other hand, targets end-to-end solutions by making TCP aware of the handoffs or robust to reordering. For example, [39] freezes the operation of TCP during handoffs to avoid the unnecessary window cutdown, [16] removes the fast retransmit mechanism in TCP such that it relies exclusively on timeouts for loss detection, and [118] adaptively changes the threshold used for inferring packet losses. These approaches, however, cannot provide the optimal solution for the environment considered in this work since they either introduce connection stalls or sacrifice performance for robustness. While one option is to have a fundamental rethinking of the transport layer approaches, in the next section, we present application layer

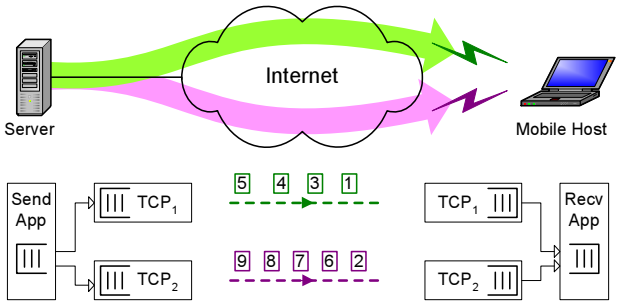


Figure 3: Striping with Multiple TCP Sockets

approaches that aim to address the problem without making any changes to TCP.

3.2 Application Layer Striping

Since TCP is designed to operate over a single path, an approach that does not change TCP is using multiple TCP connections for operating over multiple paths. In this section, we consider such an application layer approach that opens multiple TCP sockets, one each for every pipe, for striping from the source to the destination. Recall that the target problem is for a mobile host with a TCP connection to move transparently across heterogeneous wireless networks. Hence, unlike conventional *parallel sockets* approaches that can perform offline resequencing after all portions of data sent through different sockets have been received [6, 91, 98], the application layer striping approach considered needs to provide the same in-sequence semantics that TCP provides. In other words, the application layer striping approach is expected to deliver in-sequence data to the mobile host as data is striped from the server. The sending application stripes across multiple sockets, and the receiving application uses a finite buffer for resequencing. Figure 3 illustrates such an application layer striping approach.

We consider both an *unaware* application that has no knowledge of the data rate of each pipe, and a *smart* application that has some knowledge of the available data rates (which will consequently enable it to stripe more intelligently). In the former case, the sending application writes to each socket until blocked in a round-robin fashion. In the latter case, the sending application stripes data based on a ratio determined by estimation of the available rates on the different pipes. The receiving application in both cases will read packets from each socket as long as its resequencing buffer has available space. When the resequencing buffer is full, the application stops reading from sockets

that have already delivered packets with sequence numbers larger than the next expected application level sequence number.[2] It then enters a *peek mode* where it peeks into the next available packet in each of the other sockets, and reads a packet only when it is the next in-sequence packet. Note that the *recv*() socket call and its variants support a peek flag that when set allows the receive operation to retrieve data from the beginning of the receive buffer without removing that data from the buffer [112]. In such an application striping approach, it is clear that the size of the resequencing buffer will have an effect on when the sockets are blocked. If the application buffer size is zero, the application will always read in-sequence packets from the sockets. On the other hand, increasing the size of the application buffer has the effect of reducing the chances of the faster pipe being stalled by the slower one. We elaborate on this phenomenon later in Section 3.2.1 when we discuss the impact of data rate differential among the multiple pipes.

In the following, we identify the key constraints of such an application layer striping approach when operated over multiple pipes with heterogeneous characteristics.

3.2.1 Rate Differential

When the data rates of the pipes used by the unaware application are different, the aggregate bandwidth achieved by the simple approach remains a tight function of the data rate of the slowest pipe. This can intuitively be explained as follows: Consider two pipes with data rates of 10Mbps and 2Mbps respectively. Since the application stripes data by keeping the send buffer of each pipe filled, a send-buffer's worth of application data will be injected to the first (10Mbps) pipe (let this block of data be B_1). Blocked by the first pipe, the application will then proceed to inject data into the second (2Mbps) pipe (let this block of data be B_2). Because the first pipe will drain data faster, the application will, after filling the second pipe, inject more data into the first pipe (let this block of data be B_3). Assume that because of the data rate difference, the first pipe delivers B_3 before B_2 is drained out by the second pipe.

Since the additional data (block B_3) will be out-of-order, it will be queued up in the resequencing buffer of the receiving application pending the arrival of the entire block of data B_2 through the

[2] For simplicity, we assume application level sequence numbers to facilitate the resequencing process. We assume that data is packetized in the unit of the maximum segment size (MSS), and hence the sequence number refers to the packet sequence number.

second pipe. Because the first pipe will continue to transfer data at a faster rate, this will eventually result in the application's resequencing buffer overflowing. The receiving application will thereupon stop reading data from the first pipe, which in turn will cause the first pipe's TCP receiver buffer to fill up. The TCP receiver will then advertise a window size of zero, completely stalling the first pipe. Once the in-sequence data (block B_2), sent originally through the second pipe, reaches the receiver and hence releases space in the resequencing buffer, the first pipe will become active again. Note that such head-of-line blocking is indeed an artifact of the unaware striping mechanism used by the application. One way of reducing the above coupling between the faster and slower pipes is to increase the resequencing buffer size at the application layer. The larger the buffer size, the more the time for which the faster pipe can remain active without being inhibited through flow control. Specifically, if the two pipes have bandwidths of R_1 and R_2 ($R_1 < R_2$) respectively and equal delays, the application buffer required in steady state to effectively aggregate bandwidths is $\frac{R_2}{R_1} * W$, where W is the default socket buffer size. However, even assuming that the above buffer requirements can be accommodated, such buffering still cannot handle stalls that occur due to losses in the slower pipe. Also, even if the application does smart striping, such a problem will exist as long as the striping ratio does not exactly match the data rate ratio of the different pipes. We elaborate on this issue in Section 3.2.2.

The performance degradation for the simple approach could be even severer in TCP because of another phenomenon: persist timers. When the sender of the faster pipe receives a window advertisement of zero, it enters persist mode. If the single *window update* from the receiver happens to be lost (either due to congestion or random wireless losses), the sender probes the receiver only after the persist timer expires next (5 seconds). The persist timer value doubles after every unsuccessful probe and is capped only at 60 seconds [112]. While this effectively brings down the progress of the faster pipe to a crawl, the impact is more serious as the slower pipe can potentially enter persist mode because of the persist-timer induced stalling of the faster pipe! Hence, in TCP the effect of the data rate differential among the different pipes can potentially be catastrophic to the application, resulting in the aggregate throughput being *lower* than the data rate of the slowest pipe.

3.2.2 Rate Fluctuations

Although the problem due to data rate differential can be overcome by employing an intelligent striping scheme, performing such intelligent striping is inherently a difficult problem because of two reasons: (i) The pipes are end-to-end pipes that traverse multiple hops between the sender and the receiver, and the available bandwidth is likely to fluctuate dynamically; and (ii) Given the dynamic nature of wireless link characteristics, it is very likely that the pipes will exhibit highly varying data rates. When the application stripes based on the estimated data rates of the pipes, and the data rates change, the very purpose of intelligent striping is defeated resulting in degraded performance. Note that the dynamic characteristics of the wireless link, and the consequent difficulty in performing accurate rate estimation are only part of the reason for the degraded performance. The coupling of congestion control and loss recovery (for the aggregate connection) that exists because of the individual TCP pipes functioning independent of each other is also a contributing factor. For example, packets assigned to a TCP pipe by an application cannot be "withdrawn" from that pipe, notwithstanding any bandwidth reduction the pipe may experience. Thus, if bandwidth reduction occurs, packets assigned to the pipe that have not yet been transmitted due to lack of space in the reduced congestion window will be stalled and potentially cause blocking on other pipes that are still active.

3.2.3 Blackouts

Blackouts are extreme cases of rate fluctuations where the available data rate falls to zero and remains at zero for an extended period of time. Causes for such phenomena include temporary loss in connectivity (e.g. when the user is passing through a tunnel), fading, interference from a moving source, etc. Observations on the frequent and prolonged occurrence of such phenomena have been made in related work [97]. In the application layer striping approach, such blackouts on one or a subset of the pipes will stall the entire aggregate connection because of buffer overflow at the receiving application. This is obviously an undesirable phenomenon. While the only solution to this problem is to have some feedback mechanism at the application layer (for the application to realize that a particular pipe has stalled), this will substantially increase the overhead and complexity in the application as we discuss in Section 3.2.4.

3.2.4 Application Complexity

Although the above application layer approaches are simple in the sense that they do not require any protocol changes at the transport layer, the complexity and overheads at the application layer are considerable. Essentially the application has to implement a resequencing mechanism over the reordering already performed within each pipe by TCP. Sequence numbers that facilitate the resequencing have to be included in application defined headers, and the application has to explicitly ensure that the application layer "segments" (that have unique application layer sequence numbers) do not get fragmented. One conceivable way the application can ensure that application layer segments are not fragmented is to write exactly one MSS worth of data during every write – if nagling is enabled [78]. Similarly, in order to stripe intelligently, the application will have to redundantly implement a bandwidth estimation mechanism in spite of the bandwidth estimation already performed by TCP through its congestion control mechanism. Furthermore, in order to solve the problems identified as consequences of blackouts, the application will have to implement a feedback mechanism to recover from pipes that are stalled, and in effect duplicate both the reordering and loss recovery mechanisms already implemented by TCP for the individual pipes. It is clearly undesirable to overload applications in such a manner when all applications on the mobile host would require similar functionality. Note that the above arguments would also hold for session layer approaches in the absence of appropriate interfaces between the session layer and the transport layer.

3.3 A Queueing-Theoretic Perspective

We have discussed in Section 3.2 that application layer approaches without making any changes to TCP can suffer not only from performance suboptimality due to multiple pipes exhibiting vastly different characteristics, but also from protocol complexity and repetitive implementations of transport layer functionalities. In this section, we show from a queueing-theoretic perspective the limitations of application layer approaches when operated over heterogeneous pipes. In particular, we focus on the scenario with two pipes where the service rates (data rates) provided by individual pipes are mismatched. We compare application layer striping against transport layer striping in terms of the delay incurred for resequencing out-of-order packets. We first use two simple queueing models to capture the difference between the two approaches, and then derive the closed form solutions of the

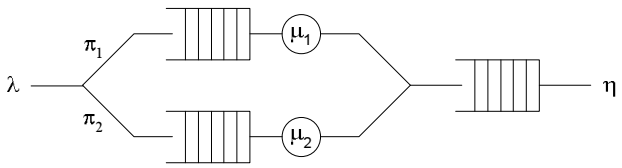


Figure 4: Application Striping Model

resequencing delay for the two striping models. Finally, we compare the performance of the two models in addressing service rate mismatches.

3.3.1 Application Striping Model

The application layer striping approach interfaces with each TCP socket using standard socket functions. While the sending application can control how data is distributed across multiple sockets (e.g. the *unaware* or *smart* application as we discussed in Section 3.2), once the data is written to the socket, it is up to the concerned TCP to control when and how the data will be sent. Packets traversing different pipes may arrive out-of-order at the receiver, but the receiving application will perform resequencing to restore the sequence of data.

A simple queueing model for the application layer striping approach operated over two pipes can be plotted as shown in Figure 4. The application data modeled as an arrival process is split to enter two transport layer queues through the striping algorithm used by the application. The queue is maintained inside each socket which the application has no control of. The service process and the service discipline are results of the algorithms (e.g. congestion control) used by TCP and the characteristics (e.g. bandwidth) of the pipe in consideration. A packet that arrives at the receiver from any pipe will wait at the resequencing buffer until all other packets with smaller sequence numbers sent through another pipe arrive. The resequencing buffer can be considered as a simplification of the resequencing process used at the TCP and the receiving application.

While a sophisticated modeling of application layer striping including the retransmission process and acknowledgment feedback in TCP is possible, the goal of this section is to *compare* application and transport striping, and gain insights into their performance tradeoffs using tractable queueing models. Therefore, we start with a queueing model with Poisson arrival process and exponential service time distribution. We assume that the arrival process is Poisson with mean arrival rate λ, and the service time distributions are exponential with mean $\frac{1}{\mu_1}$ and $\frac{1}{\mu_2}$ respectively. Arrivals

are dispatched to one of the queues following the Bernoulli routing process with probability π_1 to the first queue, and probability $\pi_2 = 1 - \pi_1$ to the second queue. In such a model, the two transport queues can be considered as two *independent* M/M/1 queues with mean arrival rates $\lambda_1 = \lambda\pi_1$ and $\lambda_2 = \lambda\pi_2$ respectively.

Consider the resequencing delay experienced by a typical packet (customer) called the tagged packet that just arrives at the system. Assume that the tagged packet upon arrival finds m packets in Q_1 (including those waiting in the queue and the one currently in service), and n packets in Q_2. With probability π_1 the tagged packet will be assigned to Q_1. Note that since we assume the server to operate in a first-come-first-served (FCFS) manner, the resequencing delay the tagged packet (that joins Q_1) will experience depends only on the number of packets it bypasses in Q_2. In other words, the resequencing delay is a function of m and n, and does not depend on the dynamics of the packets that arrive later than the tagged packet. Define $R_1(m,n)$ and $R_2(m,n)$ as the conditional resequencing delay that the tagged packet experiences when it joins Q_1 and Q_2 respectively. Then, the resequencing delay $R(m,n)$ of the tagged packet equals

$$R(m,n) = \pi_1 R_1(m,n) + \pi_2 R_2(m,n) \tag{1}$$

By removing the condition on the number of packets in the system, the expected resequencing delay R of the system can be expressed as

$$R = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} p(m,n) R(m,n) \tag{2}$$

where $p(m,n)$ is the probability that the system is in state (m,n) with m packets in Q_1 and n packets in Q_2, and $R(m,n)$ is the resequencing delay when the system is in state (m,n).

Let us proceed to evaluate $R_1(m,n)$. When the tagged packet finishes the service in Q_1 and enters the resequencing buffer, the resequencing delay it will experience depends on how many out of the n packets (present in Q_2 when it arrives) that it bypasses. Assume that upon departure from Q_1, the tagged packet finds that i packets in Q_2 have completed service. The mean resequencing delay it will experience thus can be expressed as

$$R_1(m,n) = \sum_{i=0}^{n} q(m+1,i) \frac{n-i}{\mu_2} \tag{3}$$

where $q(m+1,i)$ is the probability that at the time of the $(m+1)^{th}$ service completion in Q_1 (which corresponds to the service completion of the tagged packet), i packets in Q_2 have completed service with the $(i+1)^{th}$ packet currently in service [40]. Since only i packets in Q_2 have completed service, the tagged customer needs to wait for service completions of the remaining $(n-i)$ packets as shown in (3). It does not experience any resequencing delay if $i = n$. Note that the expected time to completion for the packet *currently in service* when the tagged packet finishes its service is $\frac{1}{\mu_2}$ as other packets waiting in the queue due to the memoryless property of the exponential distribution.

To calculate $q(m+1,i)$, assume the service completion time of the i^{th} packet in Q_j to be $T_{i,j}$. Since the sum of i independent identical exponential random variables is an Erlang type i random variable [43], the probability density function for $T_{i,j}$ is

$$f(x;i,\mu_j) = \frac{\mu_j^i}{(i-1)!} x^{i-1} \mathrm{e}^{-\mu_j x} \tag{4}$$

for $x > 0$. Hence,

$$\begin{aligned}
q_{m+1,i}(t) &\equiv q(m+1,i \mid T_{m+1,1} = t) \\
&= \mathrm{Prob}\{T_{i,2} < t,\, T_{i+1,2} > t\} \\
&= \int_0^t f(x;i,\mu_2) \int_{t-x}^{\infty} \mu_2 \mathrm{e}^{-\mu_2 y}\, \mathrm{d}y\, \mathrm{d}x \\
&= \int_0^t \left[\frac{\mu_2^i}{(i-1)!} x^{i-1} \mathrm{e}^{-\mu_2 x} \right] \mathrm{e}^{-\mu_2 (t-x)}\, \mathrm{d}x \\
&= \frac{\mu_2^i}{i!} t^i \mathrm{e}^{-\mu_2 t}
\end{aligned} \tag{5}$$

We can then find $q(m+1,i)$ as

$$\begin{aligned}
q(m+1,i) &= \int_0^{\infty} q_{m+1,i}(t)\, f(t;m+1,\mu_1)\, dt \\
&= \int_0^{\infty} \left[\frac{\mu_2^i}{i!} t^i \mathrm{e}^{-\mu_2 t} \right] \left[\frac{\mu_1^{m+1}}{m!} t^m \mathrm{e}^{-\mu_1 t} \right] \mathrm{d}t \\
&= \frac{\mu_1^{m+1} \mu_2^i}{m! i!} \int_0^{\infty} t^{m+i} \mathrm{e}^{-(\mu_1+\mu_2)t}\, \mathrm{d}t \\
&= \left(\frac{\mu_1^{m+1} \mu_2^i}{m! i!} \right) \left[\frac{(m+i)!}{(\mu_1+\mu_2)^{m+i+1}} \right] \\
&= \binom{m+i}{i} \left(\frac{\mu_1}{\mu_1+\mu_2} \right)^{m+1} \left(\frac{\mu_2}{\mu_1+\mu_2} \right)^i \\
&= \binom{m+i}{i} \delta_1^{m+1} \delta_2^i
\end{aligned} \tag{6}$$

with $\delta_1 = \mu_1/(\mu_1+\mu_2)$ and $\delta_2 = \mu_2/(\mu_1+\mu_2)$. Now (3) can be rewritten as

$$R_1(m,n) = \frac{\delta_1^{m+1}}{\mu_2}\sum_{i=0}^{n}\binom{m+i}{i}(n-i)\delta_2^i \tag{7}$$

By the same token, the mean resequencing delay $R_2(m,n)$ the tagged packet that joins Q_2 will experience when there are m packets in Q_1 and n packets in Q_2 is

$$R_2(m,n) = \frac{\delta_2^{n+1}}{\mu_1}\sum_{i=0}^{m}\binom{n+i}{i}(m-i)\delta_1^i \tag{8}$$

Finally, the expected resequencing delay of the system R can be evaluated to be

$$\begin{aligned} R &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} p(m,n)\left[\pi_1 R_1(m,n)+\pi_2 R_2(m,n)\right] \\ &= \pi_1(1-\rho_1)(1-\rho_2)\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} R_1(m,n)\rho_1^m\rho_2^n \\ &+ \pi_2(1-\rho_1)(1-\rho_2)\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} R_2(m,n)\rho_1^m\rho_2^n \end{aligned} \tag{9}$$

where $\rho_1 = \lambda\pi_1/\mu_1$ and $\rho_2 = \lambda\pi_2/\mu_2$. Note that

$$\begin{aligned} \tilde{R}_1 &\equiv \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} R_1(m,n)\rho_1^m\rho_2^n \\ &= \frac{\delta_1}{\mu_2}\sum_{m=0}^{\infty}(\rho_1\delta_1)^m\sum_{n=0}^{\infty}\rho_2^n\sum_{i=0}^{n}\binom{m+i}{i}(n-i)\delta_2^i \\ &= \frac{\delta_1}{\mu_2}\sum_{m=0}^{\infty}(\rho_1\delta_1)^m\left[\sum_{k=0}^{\infty}k\rho_2^k\right]\left[\sum_{i=0}^{\infty}\binom{m+i}{i}(\rho_2\delta_2)^i\right] \\ &= \frac{\delta_1}{\mu_2}\sum_{m=0}^{\infty}(\rho_1\delta_1)^m\frac{\rho_2}{(1-\rho_2)^2(1-\rho_2\delta_2)^{m+1}} \\ &= \frac{\delta_1\rho_2}{\mu_2(1-\rho_2)^2(1-\rho_1\delta_1-\rho_2\delta_2)} \end{aligned} \tag{10}$$

and

$$\begin{aligned} \tilde{R}_2 &\equiv \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} R_2(m,n)\rho_1^m\rho_2^n \\ &= \frac{\delta_2}{\mu_1}\sum_{n=0}^{\infty}(\rho_2\delta_2)^n\sum_{m=0}^{\infty}\rho_1^m\sum_{i=0}^{m}\binom{n+i}{i}(m-i)\delta_1^i \\ &= \frac{\delta_2\rho_1}{\mu_1(1-\rho_1)^2(1-\rho_1\delta_1-\rho_2\delta_2)} \end{aligned} \tag{11}$$

Hence, by substituting (10) and (11) back into (9), R can be simplified as

$$R = (1-\rho_1)(1-\rho_2)(\pi_1\tilde{R}_1+\pi_2\tilde{R}_2)$$

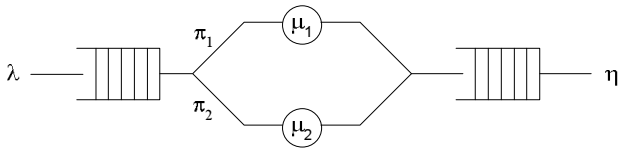


Figure 5: Transport Striping Model

$$
\begin{aligned}
&= \frac{\pi_1(1-\rho_1)^2\delta_1\rho_2\mu_1 + \pi_2(1-\rho_2)^2\delta_2\rho_1\mu_2}{\mu_1\mu_2(1-\rho_1)(1-\rho_2)(1-\rho_1\delta_1-\rho_2\delta_2)} \\
&= \frac{\lambda\pi_1\pi_2\left[\mu_1(\mu_1-\lambda\pi_1)^2 + \mu_2(\mu_2-\lambda\pi_2)^2\right]}{\mu_1\mu_2(\mu_1-\lambda\pi_1)(\mu_2-\lambda\pi_2)(\mu_1+\mu_2-\lambda)}
\end{aligned} \tag{12}
$$

which is the same as the result shown in [59] using the probability generating function approach. Note that since the expected queueing delay (including the service time) D equals

$$D = \frac{\pi_1}{\mu_1 - \lambda\pi_1} + \frac{\pi_2}{\mu_2 - \lambda\pi_2} \tag{13}$$

we can obtain the expected total delay T for the application striping model by summing (12) and (13). It represents the expected time lapsed since the server writes a new packet to a TCP socket until the packet is ready for use by the mobile host.

3.3.2 Transport Striping Model

The application layer striping approach maintains multiple TCP sockets in the connection, and hence it is modeled as parallel server queues in Section 3.3.1. The transport layer striping approach, however, performs striping at the transport layer, and hence it does not need to maintain multiple queues. In this section, we use a single transport queue with multiple servers to model transport layer striping as shown in Figure 5. A resequencing buffer is used at the receiver to model the resequencing process as is the case for application layer striping.

In the transport striping model, arrivals are enqueued in the transport queue and wait to be served until reaching the head of the queue. A packet at the head of the queue is served by the server that becomes idle. In case both servers are idle, the packet chooses the first server S_1 with probability π_1 and the second server S_2 with probability π_2. Note that if both servers are idle, the tagged packet will not experience any resequencing delay irrespective of the server it chooses – since all its predecessors have completed service before it starts service. The tagged packet will need to wait in the resequencing buffer only if it finds one of the server busy right before its service,

and finishes its service before the packet in the busy server.

Let us consider a tagged packet at the head of the queue that is about to be served. Assume that the tagged packet finds S_1 idle and S_2 busy. It then goes into the service of S_1, and enters the resequencing buffer after service completion. The tagged packet will need to wait in the resequencing buffer if its service time is less than the residual service time of the packet served by S_2. The mean resequencing delay W_1 after departing from S_1 can be calculated as

$$\begin{aligned} W_1 &= \int_0^\infty \int_t^\infty (s-t)\mu_2 e^{-\mu_2 s}\mu_1 e^{\mu_1 t} ds\, dt \\ &= \frac{\mu_1}{\mu_2(\mu_1+\mu_2)} \end{aligned} \tag{14}$$

due to the memoryless property of the exponential distribution. On the other hand, if the tagged packet finds S_1 busy and S_2 idle, it will be served by S_2. Its mean resequencing delay W_2 after departing from S_2 thus equals

$$W_2 = \frac{\mu_2}{\mu_1(\mu_1+\mu_2)} \tag{15}$$

Denote $\tilde{P}_{ij}$ as the probability that the tagged packet finds right before its service the state of S_1 and S_2 to be i and j respectively, assuming 0 as the idle state and 1 as the busy state. The expected resequencing delay R of the system thus can be expressed as

$$R = \tilde{P}_{00}\cdot 0 + \tilde{P}_{01}W_1 + \tilde{P}_{10}W_2 \tag{16}$$

where $\tilde{P}_{00} = 1-\tilde{P}_{01}-\tilde{P}_{10}$. We can obtain $\tilde{P}_{01}$ and $\tilde{P}_{10}$ using the equilibrium state probabilities of the system. That is,

$$\tilde{P}_{01} = p(0,1,0) + \frac{\mu_1}{\mu_1+\mu_2}\left[\sum_{k=0}^{\infty} p(1,1,k)\right] \tag{17}$$

$$\tilde{P}_{10} = p(1,0,0) + \frac{\mu_2}{\mu_1+\mu_2}\left[\sum_{k=0}^{\infty} p(1,1,k)\right] \tag{18}$$

where $p(i,j,k)$ is the probability that the first server is in state i, the second server is in state j, and the number of packets in the queue (excluding the servers) is k. Note that given both servers are currently in state 1, the probability that the packet served by S_1 will depart before that served by S_2 is $\mu_1/(\mu_1+\mu_2)$.

To calculate the state probabilities, we can start with the balance equations of the system as [66]

$$\lambda p(0,0,0) = \mu_1 p(1,0,0) + \mu_2 p(0,1,0)$$

$$
\begin{aligned}
(\lambda+\mu_2)p(0,1,0) &= \mu_1 p(1,1,0)+\lambda\pi_2 p(0,0,0)\\
(\lambda+\mu_1)p(1,0,0) &= \mu_2 p(1,1,0)+\lambda\pi_1 p(0,0,0)\\
(\lambda+\mu_1+\mu_2)p(1,1,0) &= (\mu_1+\mu_2)p(1,1,1)+\lambda[p(0,1,0)+p(1,0,0)]\\
(\lambda+\mu_1+\mu_2)p(1,1,k) &= (\mu_1+\mu_2)p(1,1,k+1)+\lambda p(1,1,k-1), \qquad k\geq 1
\end{aligned}
$$

and arrive at the following solutions:

$$
\begin{aligned}
p(0,1,0) &= \frac{\lambda(\pi_2+\rho)}{\mu_2(1+2\rho)}p(0,0,0) && (19)\\
p(1,0,0) &= \frac{\lambda(\pi_1+\rho)}{\mu_1(1+2\rho)}p(0,0,0) && (20)\\
p(1,1,0) &= \frac{\lambda(\lambda+\mu_1\pi_2+\mu_2\pi_1)\rho}{\mu_1\mu_2(1+2\rho)}p(0,0,0) && (21)\\
p(1,1,k) &= \rho^k p(1,1,0), \qquad k\geq 1 && (22)
\end{aligned}
$$

where $\rho=\lambda/(\mu_1+\mu_2)$ is the traffic intensity, and $p(0,0,0)$ equals

$$
p(0,0,0)=\left[1+\frac{\lambda(\lambda+\mu_1\pi_2+\mu_2\pi_1)}{\mu_1\mu_2(1+2\rho)(1-\rho)}\right]^{-1} \qquad (23)
$$

Therefore, we can obtain the expected resequencing delay of the system as

$$
\begin{aligned}
R &= \tilde{P}_{01}W_1+\tilde{P}_{10}W_2\\
&= \left[p(0,1,0)+\frac{\mu_1}{(\mu_1+\mu_2-\lambda)}p(1,1,0)\right]\frac{\mu_1}{\mu_2(\mu_1+\mu_2)}\\
&+ \left[p(1,0,0)+\frac{\mu_2}{(\mu_1+\mu_2-\lambda)}p(1,1,0)\right]\frac{\mu_2}{\mu_1(\mu_1+\mu_2)} \qquad (24)
\end{aligned}
$$

where $p(i,j,k)$ can be readily obtained using (19)–(23). The queueing delay D (including the service time) can be evaluated using the expected queue length L and Little's formula:

$$
\begin{aligned}
D &= \frac{L}{\lambda}\\
&= \frac{1}{\lambda}\left[p(0,1,0)+p(1,0,0)+\sum_{k=0}^{\infty}(k+2)p(1,1,k)\right]\\
&= \left[\frac{1}{\rho}+\frac{2-\rho}{(1-\rho)^2}\right]\frac{p(1,1,0)}{\lambda}\\
&= \frac{\lambda+\mu_1\pi_2+\mu_2\pi_1}{\mu_1\mu_2(1+2\rho)(1-\rho)^2}p(0,0,0) \qquad (25)
\end{aligned}
$$

where $p(0,0,0)$ is defined in (23). The total delay in the system T can be easily obtained by summing (24) and (25).

3.3.3 Performance Comparisons

In this section, we compare the performance of the application striping model (ASM) and the transport striping model (TSM) using the closed form solutions developed in Section 3.3.1 and Section 3.3.2 respectively. The objective is to study the value of the Bernoulli routing probability π_1 (and hence π_2) on the performance of the two systems. Since the choice of different π_1 values can be considered as the result of using different striping algorithms at the sender (e.g. through estimating the bandwidths of the two pipes to decide how data should be striped), the comparisons can provide insights regarding where (application or transport layer) a striping algorithm should ideally be implemented for achieving better performance. We focus on the delay performance of the two systems in terms of the expected resequencing delay R and total system delay T (resequencing delay plus the queueing delay D). Note that in these two queueing models, the mean waiting delay in the buffer is a direct indication of the mean buffer occupancy (i.e. $L_D = \lambda D$ and $L_R = \lambda R$), and hence the expected resequencing delay can be translated to the size of the mean resequencing buffer. We first consider the scenario when both servers have the same service rate (homogeneous servers), and then we consider the scenario with mismatched service rates (heterogeneous servers).

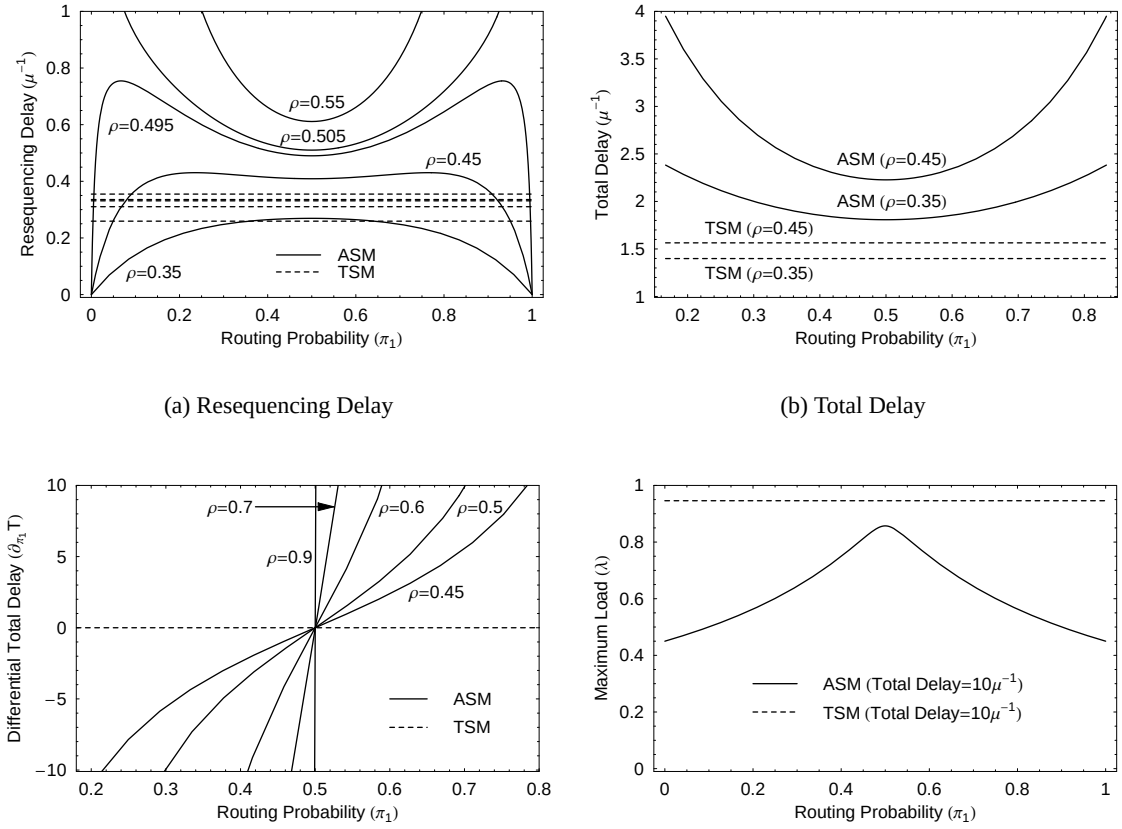
3.3.3.1 Homogeneous Servers

In case $\mu_1 = \mu_2 = \mu$, we can reduce (24) and (25) for the transport striping model to

$$R_{\text{TSM}} = \frac{\lambda}{2\mu^2 + \mu\lambda} = \frac{1}{\mu}\left(\frac{\rho}{1+\rho}\right) \tag{26}$$

$$D_{\text{TSM}} = \frac{4\mu}{4\mu^2 - \lambda^2} = \frac{1}{\mu}\left(\frac{1}{1-\rho^2}\right) \tag{27}$$

with $\rho = \frac{\lambda}{2\mu}$. Hence, the expected resequencing delay and queueing delay in the transport striping model are *independent* of the routing probability used. This can be explained as follows: when the tagged packet arrives to find both servers idle, it chooses the first server with probability π_1 and the second server with probability $1 - \pi_1$. Irrespective of which server it chooses, its resequencing delay will be zero in both cases, and its mean service time will also be the same in both cases since the two servers have the same service rate. For the application striping model, however, the routing probability has a great impact on the system performance [59]. Figure 6(a) shows the resequencing delay (normalized to the service time) of the two systems with respect to the routing probability.



(a) Resequencing Delay

(b) Total Delay

(c) Sensitivity to the Routing Probability

(d) Maximum Traffic Load

Figure 6: Homogeneous Servers ($\mu_1 = \mu_2$)

The figure is obtained for different traffic intensities that range from 0.35 to 0.55. For comparison, we also show the resequencing delay in the transport striping model with the same traffic intensity. The five dashed lines for the transport striping model are not labeled in the figure for sake of clarity, but it should be evident from (26) that $\rho = 0.35$ corresponds to the bottom line while $\rho = 0.55$ corresponds to the topmost line. We can observe that the resequencing delay of the application striping model is sensitive to the value of the routing probability. The figure is symmetric around $\pi_1 = 0.5$ since the two servers are identical. Consider now the region where $0 < \pi_1 \leq 0.5$. If $\rho < 0.5$ (i.e. $\lambda < \mu$), it is possible to achieve zero resequencing delay by directing all traffic to only one queue ($\pi_1 = 0$). For nonzero routing probability, however, the resequencing delay of the system is the result of the tradeoff between *the number of packets* that have to wait in the resequencing buffer, and *the amount of time* each packet has to wait. When the load is very low, diverting traffic

from the second queue to the first queue (increasing π_1 from zero) has a slight impact on reducing the queue length of the second queue, but increases the number of packets in the first queue that experience resequencing. Hence, the resequencing delay increases with increasing π_1 and reaches the maximum value at $\pi_1 = 0.5$, as illustrated by the $\rho = 0.35$ curve. On the other hand, when the load is close to 0.5, diverting traffic from the second queue can significantly reduces its queue length (which equals $\frac{\rho}{0.5-\rho}$). Hence, after an initial increase, the resequencing delay decreases with increasing values of π_1 as illustrated by the $\rho = 0.495$ curve. In fact, from (12) we can show that $\pi_1 = 0.5$ is a global maximum if $0 < \rho \leq \sqrt{2}-1$, and a local minimum if $\sqrt{2}-1 < \rho < 1$. The mean resequencing delay and queueing delay at $\pi_1 = 0.5$ are

$$\hat{R}_{\text{ASM}} = \frac{\lambda}{4\mu^2 - 2\mu\lambda} = \frac{1}{\mu}\left[\frac{\rho}{2(1-\rho)}\right] \tag{28}$$

$$\hat{D}_{\text{ASM}} = \frac{2}{2\mu - \lambda} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \tag{29}$$

If $\rho > 0.5$, the system is defined only for the range $1 - \frac{1}{2\rho} < \pi_1 < \frac{1}{2\rho}$ to ensure stability. The resequencing delay has a global minimum at $\pi_1 = 0.5$ as shown in the figure. Note that as ρ approaches 1, R_{TSM} converges to $\frac{1}{2\mu}$ while $\hat{R}_{\text{ASM}}$ becomes unbounded. Specifically, it can be shown that $R_{\text{TSM}} \leq \hat{R}_{\text{ASM}}$ as long as $\rho \geq \frac{1}{3}$.

As we showed in (26) and (27), the total delay T_{TSM} in the transport striping model is independent of the routing probability. For the application striping model, however, we can show using (12) and (13) that T_{ASM} is convex with the global minimum at $\pi_1 = 0.5$. The minimum value of T_{ASM} is equal to the sum of (28) and (29). Obviously, the optimal operating point for the application striping model to reduce the total delay is to split the traffic evenly over the two queues when the two servers are identical. Define Δ as the minimum difference in the total delay of the two systems. Then

$$\begin{aligned} \Delta &= \hat{T}_{\text{ASM}} - T_{\text{TSM}} = \frac{2+\rho}{2\mu(1-\rho)} - \frac{1+\rho-\rho^2}{\mu(1-\rho^2)} \\ &= \frac{\rho(1+3\rho)}{2\mu(1-\rho^2)} > 0, \qquad \forall \rho \in (0,1) \end{aligned}$$

In other words, *the total delay of the application striping model is always larger than that of the transport striping model*, as illustrated in Figure 6(b). Note that this is slightly different from the case of the resequencing delay when the load is low ($\rho < \frac{1}{3}$). Therefore, when the offered load is low, while it is possible for the application striping model to reduce the resequencing delay by

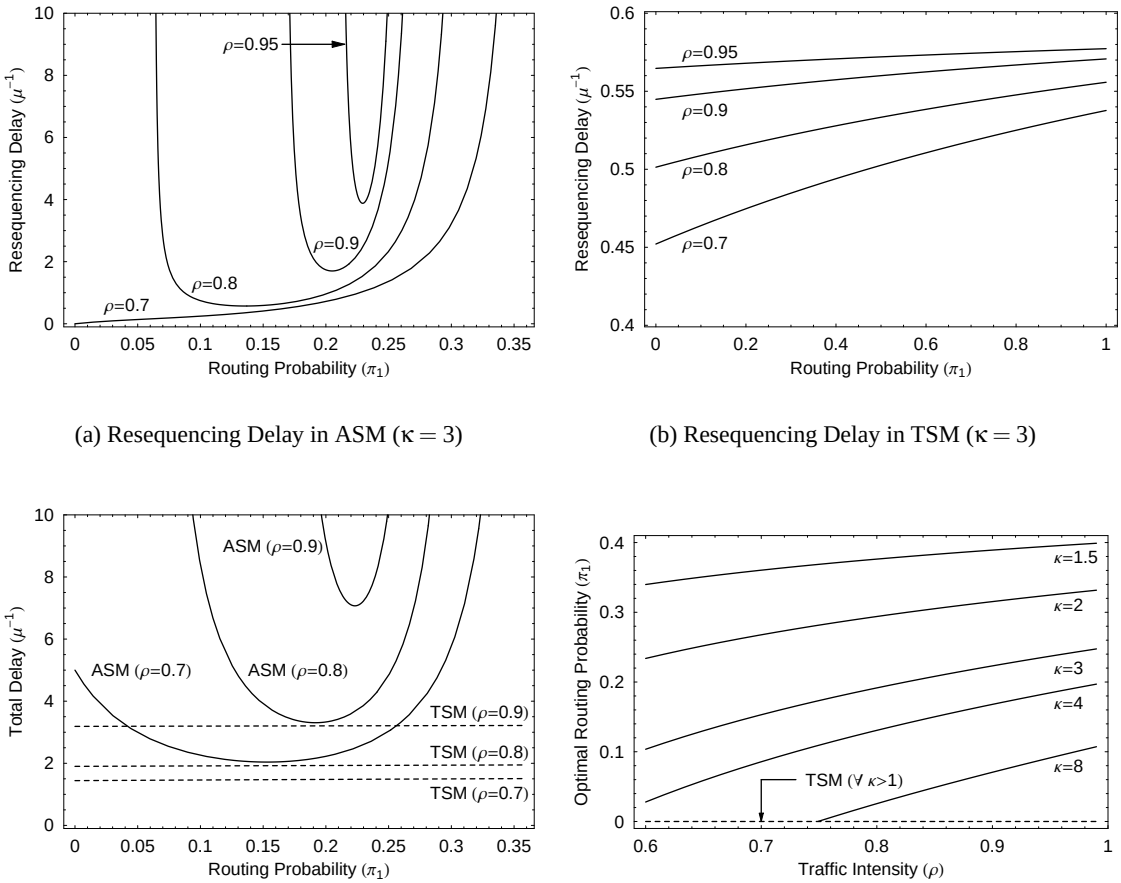
directing arrivals to only one of the queue predominantly, the resulting increase in the queueing delay due to the under-utilization of the other server always *outweigh* such a decrease, making its performance worse than the transport striping model.

We use Figure 6(c) and Figure 6(d) to depict the sensitivity of the application striping model to the value of the routing probability. In Figure 6(c) we plot the "rate of change" (derivative) of the total delay with respect to the routing probability at the optimal operating point $\pi_1 = 0.5$. It is clear from the figure that as the load increases, the slope for the application striping model becomes much steeper. Therefore, when the system is operated at a relatively high load, *a slight suboptimality in the choice of the routing probability used can cause drastic increase in the total delay.* Figure 6(d) provides a different view for the same phenomenon. Consider the scenario where admission control is used in the two systems to bound the mean latency to be within an allowable range (e.g. for providing certain quality of service for time-sensitive packets). Figure 6(d) shows that the maximum load that the application striping model can sustain is lower than that the transport striping model can sustain – even when the former is operated under the optimal condition. More importantly, the application striping model suffers from significant performance loss (the maximum sustainable load decreases) when the routing probability is not optimally chosen. In other words, the application striping model is likely to operate in an non-ideal condition if the service rates of the two servers are not accurately known or unknown to the striping algorithm.

To summarize, for non-trivial scenarios with moderate to high offered load ($\rho > 0.5$ to benefit from the provision of the two pipes), the transport striping model always achieves a much better performance than the application striping model in terms of both the resequencing delay and the total system delay. The delay performance can be translated to the requirement on the size of the buffer as we mentioned before. Moreover, the application striping model is vulnerable to the value of the routing probability used, which can be a problem if the precise bandwidths of the pipes are not known to the sending application.

3.3.3.2 Heterogeneous Servers

We now consider the scenario where the two servers have mismatched service rates. Assume $\mu_1 = \mu$, $\mu_2 = \kappa\mu$, and $\lambda = \rho(1+\kappa)\mu$, where $\kappa > 0$ is the service rate ratio of the two servers, and $0 < \rho < 1$



(a) Resequencing Delay in ASM ($\kappa = 3$)

(b) Resequencing Delay in TSM ($\kappa = 3$)

(c) Total Delay ($\kappa = 3$)

(d) Optimal Routing Probability vs. Traffic Intensity

Figure 7: Heterogeneous Servers ($\mu_2 = \kappa\mu_1$)

is the traffic intensity. We focus on non-trivial scenarios where the mean arrival rate is larger than the mean service rate of the faster server (i.e. it is necessary to use both servers).

Figure 7(a) shows the resequencing delay for the application striping model when the routing probability varies from 0 to 1. The figure is obtained for $\kappa = 3$, and ρ ranging from 0.7 to 0.95. We observe from the figure that the resequencing delay of the application striping model depends on the routing probability used – as is the case for homogeneous servers. For $0 < \rho < 0.75$, the minimum resequencing delay occurs at $\pi_1 = 0$, which means only the faster server should be used for minimizing the resequencing delay (since $\lambda < \mu_2$). For $0.75 < \rho < 1$, the curves are convex with global minima occurring at $\pi_1 \neq 0$. However, unlike in Figure 6(a) for homogeneous servers where the optimal probability $\hat{\pi}_1$ is always equal to 0.5 irrespective of the load, $\hat{\pi}_1$ for heterogeneous

servers depends on the traffic intensity. It can be seen from Figure 7(a) that $\hat{\pi}_1$ increases with ρ. For example, $\hat{\pi}_1 \approx 0.14$ for $\rho = 0.8$, and $\hat{\pi}_1 \approx 0.23$ for $\rho = 0.95$. In fact, from (12) we can show

$$\lim_{\rho \to 1} \hat{\pi}_1 = \frac{1}{1+\kappa} \tag{30}$$

and hence $\hat{\pi}_1$ in Figure 7(a) will converge to 0.25 as ρ approaches 1. Intuitively, the optimal strategy for reducing the resequencing delay is to split the traffic in accordance with the ratio of the service rates. Note, however, that the optimal resequencing delay in the application striping model also increases with ρ. Specifically, the resequencing delay and queueing delay at $\hat{\pi}_1$ for the application striping model can be evaluated using (12) and (13):

$$\hat{R}_{\mathrm{ASM}} = \left(\frac{1-\kappa+\kappa^2}{\kappa+\kappa^2}\right)\frac{\rho}{\mu(1-\rho)} \tag{31}$$

$$\hat{D}_{\mathrm{ASM}} = \left(\frac{2}{1+\kappa}\right)\frac{1}{\mu(1-\rho)} \tag{32}$$

which reduce to (28) and (29) respectively when $\kappa = 1$. We observe from Figure 7(a) that the "penalty" of the suboptimality in the routing probability chosen rockets as ρ approaches 1 – as indicated by the steep increase of the resequencing delay near the global minimum point. An interesting observation from (31) and (32) is that, if we keep the capacity of the system fixed ($\mu_1 + \mu_2 = 1$), but increases the disparity of the two servers ($\kappa \to \infty$), then even with the optimal traffic splitting strategy ($\pi_1 = \frac{1}{1+\kappa}$) the mean resequencing delay becomes unbounded, while the mean queueing delay stays constant at $\frac{2}{1-\rho}$.

Figure 7(b) shows the resequencing delay for the transport striping model using the same scenarios as in Figure 7(a). We observe from the figure that, unlike in the homogeneous servers case, the resequencing delay depends on the choice of the routing probability. Compared to the application striping model, however, the dependency is much less significant (note the scale of the y-axis). Moreover, the optimal routing probability always occurs at $\pi_1 = 0$ irrespective of the load of the system. Intuitively, if the tagged packet finds both servers idle right before its service, it is always favorable to choose the faster server to minimize the resequencing delay that *the next packet* might experience (in addition to minimizing the service time of the tagged packet). We note from Figure 7(b) that the curve flattens out as the load increases. It can be shown that

$$\lim_{\rho \to 1} R_{\mathrm{TSM}} = \frac{1-\kappa+\kappa^2}{\mu(\kappa+\kappa^2)} \tag{33}$$

which is independent of the routing probability. Hence, for a loaded transport striping model ($\rho \rightarrow 1$), packets can randomly choose one idle server without suffering from perceivable performance degradation.

Finally, in Figure 7(c) we compare the total delay of the two systems. It can be observed that the transport striping model achieves better performance than the application striping model for all routing probabilities. In addition, the performance gains increase as the load of the system increases, as is the case for homogeneous servers. Figure 7(d) shows the dependency of the optimal routing probability $\hat{\pi}_1$ on the traffic intensity for different values of κ. As we showed in Figure 7(b), the optimal routing probability for the transport striping model is independent of the traffic intensity. However, for the application striping model, the optimal routing probability is a function of the traffic intensity offered. Moreover, the optimal routing probability for the application striping model is a function of the service mismatch ratio (κ). The rightmost point for each of the ASM curves will reach $\frac{1}{1+\kappa}$ when ρ approaches 1, as indicated in (30). We note that as the service mismatch ratio increases, the optimal routing probability for the application striping model becomes more sensitive to the traffic intensity. This can intuitively be interpreted as the "penalty" of the suboptimality in striping incoming packets to the right pipe increases as the server heterogeneity increases.

In summary, for non-trivial scenarios where the load is greater than the capacity of the faster server, the transport striping model achieves better delay performance than the application striping model. Moreover, the application striping model exhibits much higher sensitivity to the routing probability than the transport striping model. Therefore, any suboptimality of the striping algorithm in estimating the exact bandwidths of all the pipes will result in significant performance degradation in the application striping model. The sensitivity to the routing probability can pose a very high demand on the accuracy of the striping algorithm used at the sender for achieving optimal performance. We corroborate the analysis in this section through packet simulation using working network protocols and a practical network topology in Section 4.3.

3.4 Summary

In this chapter, we motivated a solution based on the transport layer protocol to address the problem of host mobility across heterogeneous wireless networks. We started with end-to-end solutions due

to their ability to support mobility without relying on the support from any entity that is specific to the network in consideration. We then identified the key functionality that needs to be supported for achieving seamless network handoffs, server migration, and bandwidth aggregation as the ability to operate (stripe) over multiple pipes. We discussed why application layer striping, an end-to-end approach that does not require changes to the transport layer, can suffer from performance degradation and implementation complexity when operated over multiple pipes exhibiting rate differential, rate fluctuations, and blackouts. We also used queueing-theoretic analysis to model application layer striping and transport layer striping, and derived the closed form solutions of the two models regarding the delay characteristics and buffer requirement. We showed that not only does the application striping model achieve lower performance than the transport layer striping model, but it also exhibit higher sensitivity to the accuracy of the striping algorithm used. We thus concluded that a transport layer solution can achieve the ideal performance in addressing the problem. Existing transport layer protocols, however, cannot effectively operate over multiple pipes and handle network heterogeneity. In the rest of the work, we propose two fundamental principles called *parallelism* and *transpositionality* that should be incorporated in designing new transport layer protocols to address network heterogeneity. The principle of parallelism allows a transport protocol to leverage resource multiplicity, while the principle of transpositionality allows a transport protocol to address resource disparity. We show that a transport layer protocol designed with the two principles can address the drawbacks in existing transport protocols and support transparent host mobility across heterogeneous wireless networks.

CHAPTER 4

PARALLELISM

We have so far motivated a transport layer solution to address the problem of network heterogeneity in future wireless networks. As we explained in Section 3.1 that a common theme behind the key functionalities that need to be provided for supporting transparent host mobility is the existence of multiple pipes between the server and the mobile host. Existing transport layer protocols, however, cannot seamlessly operate over multiple pipes. In this chapter, we propose *parallelism* as the first fundamental principle that needs to be incorporated in transport layer design for supporting transparent host mobility. In particular, we show how TCP, the prevailing transport layer protocol designed for a single path, can be extended to be a *parallel* protocol called **pTCP** (parallel TCP) that can effectively operate over multiple paths. We first discuss the key tenets for designing pTCP the parallel protocol, and then present the detailed protocol operations including the state diagram and protocol handshake for such a parallel protocol. Finally, we evaluate the performance of pTCP and compare it with an application layer striping approach. We conclude that the principle of parallelism incorporated in the transport layer design can effectively enable seamless operations over multiple pipes.

4.1 Protocol Design

We discuss in this section several key design elements for building the parallel TCP protocol. Whenever appropriate, we explain how the design elements can help address the problems of inefficient striping at the application layer that we identified in Section 3.2.

4.1.1 Maintaining Multiple States

Since TCP is designed for operating over a single path between the source and the destination, it captures the characteristics of the path it traverses such as bandwidth and latency in the form of TCB (Transmission Control Block) state variables [86]. TCB includes variables such as congestion

window and round-trip time for TCP to determine the available data rate of the path. Since multiple pipes in a connection can exhibit a very high degree of heterogeneity in terms of the data rates, round-trip times, and loss rates, maintaining only one set of TCB variables (single state) can render the achieved throughput suboptimal. Therefore, the first element in designing a parallel transport protocol for operating over multiple pipes is to maintain multiple states in accordance with the number of pipes used in the connection. In the context of parallel TCP, multi-state design allows it to maintain one TCB state for each pipe that becomes active in the connection. Since each state is associated with only one pipe, the single-state design of TCP can be reused with minimal changes – without suffering from performance degradation as we identified in Section 3.1.

4.1.2 Decoupling of Functionalities

Application layer striping approaches as we discussed in Section 3.2.4 suffer from considerable complexities and overheads due to repetitive implementations of functionalities across multiple sockets. Therefore, to incur minimum overheads resulting from the multi-state design, pTCP should decouple the transport layer functionalities associated with per-pipe characteristics from those pertaining to the aggregate connection. Toward this end, pTCP is designed as a *wrapper* around a slightly modified TCP that we refer to as *TCP-v* (TCP-virtual). The TCP-v opened for each pipe handles the per-pipe state (TCB), while the pTCP engine (which for simplicity we also refer to as pTCP) handles the aggregate connection. Specifically, pTCP maintains and controls a single send buffer across all the TCP-v pipes for the aggregate connection. The individual TCP-v pipes perform congestion control and loss recovery just like regular TCP. However, any segment transmission by a TCP-v is preceded by an explicit call to pTCP requesting for application data. Since pTCP has control over the buffer, a retransmission at the TCP-v level does not need to be a retransmission at the pTCP level. On the other hand, the amount of data that can be sent out through each TCP-v pipe is strictly determined by the TCP congestion control algorithm employed by each respective pipe. Therefore, TCP-v controls the *amount* of data that can be sent while pTCP controls *which* data to send. In this fashion, pTCP decouples congestion control and reliability. We describe as we go along how the decoupling contributes to improved performance and functionality in pTCP.

4.1.3 Delay Binding

When a TCP-v pipe has space in its congestion window for transmissions, it requests pTCP for data. If there exists no unsent data, pTCP registers the concerned TCP-v pipe as an *active* pipe and returns a *freeze* value. The TCP-v pipe then waits for a subsequent *resume* call from pTCP before requesting for data again. When pTCP receives new data from the application, it issues the resume call only to those TCP-v pipes that are registered as active. Note that such striping is different from striping that is conditional on buffer availability (as seen in the unaware application layer approach). In pTCP, *data will be given to a TCP-v pipe only when there is space in its congestion window for the data to be sent.* Note that this inherently assumes the congestion window to be a true representative of the bandwidth-delay product (BDP) of the pipe. While the TCP congestion window *is* an approximation of the BDP, it is possible that it is an incorrect estimation (say, for example, due to deep buffers in the network). The striping of data based on the congestion window of the individual pipes removes the problem that arises due to differences in the rates of the pipes, *provided there is no fluctuation in the available bandwidth*.

4.1.4 Dynamic Reassignment

Recall that it is possible for the congestion window to be an over-estimate especially just before congestion occurs. This can result in an undesirable hold up of data in pipes where the congestion window was reduced recently. For example, consider a scenario in which the congestion window of pipe p_i is $cwnd_i$. If $cwnd_i$ worth of data is assigned to p_i, and the window is cut down to $\frac{cwnd_i}{2}$ due to bandwidth fluctuations, the $\frac{cwnd_i}{2}$ worth of data that falls outside the congestion window of p_i will be blocked from transmission till the $cwnd_i$ opens up. In the meantime, this is equivalent to a static scenario in which the application undesirably assigned more data than what a pipe can carry and in the process slows down other faster pipes. pTCP solves this problem by leveraging the decoupling that exists between congestion control and reliability. When a pipe experiences congestion, irrespective of whether the detection is through duplicate acknowledgments or a timeout, the window is reduced (by half in the former and to one in the latter). If the congestion window of a pipe is thus reduced, pTCP immediately *unbinds* the data that was bound to the sequence numbers of the concerned pipe that fall outside the current congestion window. Thus, if another pipe has space in

its congestion window and requests for data, the unbound data is now available for reassignment to that pipe. When the original pipe requests for data corresponding to the same sequence number that was unbound, new application data is bound by pTCP and returned to the pipe. Such a reassignment strategy can greatly improve the performance of pTCP under dynamic conditions.

4.1.5 Redundant Striping

While the strategy described above reassigns data that falls out of a pipe's congestion window, it does not deal with the one MSS (maximum segment size) worth of data (the first MSS in the congestion window) that will never fall out of the congestion window irrespective of the state of the pipe. Failure to deliver that one MSS worth of data can potentially stall the entire aggregate connection if the concerned pipe undergoes multiple timeouts or suffers a blackout. Hence, pTCP *redundantly stripes the first MSS of data in a congestion window that has suffered a timeout, onto another pipe.* In doing so, the binding of the data is changed to the new pipe, although the old pipe has access to a copy of the same data. The reason for leaving a copy behind instead of a regular reassignment is that the old pipe will require at least one MSS worth of data to send in order to recover. At the same time, providing it with a new MSS worth of data is a potential pitfall because of the chances of blocking, given that the pipe is experiencing severe conditions.

4.2 Protocol Operations

We now present the detailed operations of pTCP, including its software architecture, state diagram, protocol handshake, and protocol operations.

4.2.1 Architectural Overview

Figure 8 provides an architectural overview of the pTCP protocol. pTCP acts as the central engine that interacts with the application, IP, and TCP-v respectively. pTCP creates and maintains one TCP-v for each active pipe used by the application. The figure also illustrates the key data structures maintained for every aggregate connection. pTCP controls and maintains the send and receive socket buffers for the connection. Application data writes are served by pTCP, and the data is copied onto the `send_buffer`. A list of active TCP-v pipes (that have space in the congestion window to transmit) called `active_pipes` is maintained by pTCP. A TCP-v pipe is placed in `active_pipes`

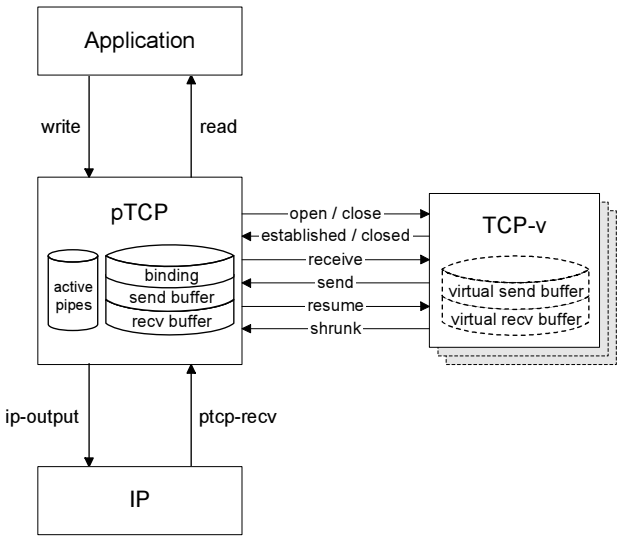


Figure 8: pTCP Architecture

initially when it is created by pTCP. Upon the availability of data that needs to be transmitted, pTCP sends a *resume*() command to the active TCP-v pipes. Once a resume is issued to a pipe, the corresponding pipe is removed from `active_pipes`. A TCP-v pipe that receives the command builds a regular TCP header based on its state variables, and gives the segment (sans the data) to pTCP through the *send*() interface. pTCP binds an unbound data segment in the `send_buffer` to the header of the "virtual" segment TCP-v has built, maintains the binding in the data structure called `binding`, inserts its own header, and sends it to the IP layer. A *resumed* TCP-v continues to issue *send*() calls till there is no more space left in the congestion window, or pTCP responds back with a *freeze* return value to "freeze" the concerned pipe. When pTCP receives a *send*() call, and has no unbound data left, it returns a *freeze* value, and adds the corresponding pipe to `active_pipes`.

When pTCP receives an *ACK*, it strips the pTCP header, and hands over the packet to the appropriate TCP-v pipe through the *receive*() interface. The correct TCP-v pipe is recognizable from the TCP 4-tuple. The TCP-v pipe processes the *ACK* in the regular fashion, and updates its state variables including the virtual send buffer. The virtual buffer can be thought of as a list of segments that have only appropriate header information. The virtual send and receive buffers are required to ensure regular TCP semantics for congestion control and connection management within each TCP-v pipe. The pTCP header carries cumulative pTCP level ACK information that pTCP uses to purge its receive buffer if required. When pTCP receives an incoming data segment, it strips both

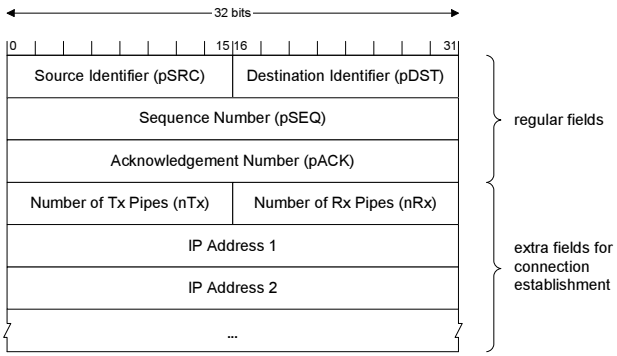


Figure 9: pTCP Header Format

the pTCP header and the data, enqueues the data in the `recv_buffer`, and provides the appropriate TCP-v with only the skeletal segment that does not contain any data. TCP-v treats the segment as a regular segment except that no application data is queued in the virtual receive buffer.

4.2.2 TCP-v Interface

As seen in Figure 8, the following eight functions act as the interface between pTCP and TCP-v: *open*(), *close*(), *established*(), *closed*(), *receive*(), *send*(), *resume*(), and *shrunk*(). pTCP uses the *open*() and *close*() calls as inputs to the TCP-v state machine for opening and closing a TCP-v pipe respectively. TCP-v uses the *established*() and *closed*() interfaces to inform pTCP when its state machine reaches the **ESTABLISHED** and **CLOSED** states respectively [112]. The *send*() call is used by TCP-v to send "virtual" segments to pTCP which will bind the segments to real data. The *receive*() interface on the other hand is used by pTCP to deliver "virtual" segments to TCP-v. pTCP uses *resume*() to inform TCP-v that additional unbound data is available. TCP-v, upon receiving the call, attempts to send as much data as possible till it gets a *freeze* return value on its *send*() call and freezes. Finally, TCP-v uses the *shrunk*() interface to inform pTCP of any change in its congestion window so that pTCP can perform reassignment as described in Section 4.1.

4.2.3 Header Formats

Figure 9 presents the header formats for the pTCP protocol. Note that the header is in addition to the regular TCP header that will be used by TCP-v. The regular pTCP header consists of the following four fields: (i) source connection identifier (*pSRC*), (ii) destination connection identifier (*pDST*), (iii) pTCP sequence number (*pSEQ*), and (iv) pTCP acknowledgment number (*pACK*).

The connection identifiers are used to uniquely identify the aggregate pTCP connection at both ends. The *pSEQ* is the sequence number at the aggregate connection level and is independent of the TCP sequence number. The *pACK* is a cumulative acknowledgment similar to the TCP acknowledgment (ACK) field. Note that the individual TCP-v pipes will use the TCP ACK fields to perform congestion control (recall that congestion control and loss recovery are coupled in TCP), and hence they cannot be reused by pTCP. Because pTCP is responsible for performing flow control (given that it controls the buffer), it requires a field for window advertisement as in TCP. However, since TCP-v pipes do not have to perform flow control (they merely maintain virtual buffers), pTCP reuses and overrides the TCP window advertisement field for performing flow control. The reuse does not interfere with the progress of the individual TCP-v pipes due to the fact that the pTCP advertised window will always be greater than the actual window of an individual pipe (we elaborate on this in Section 4.2.5).

In addition to the regular pTCP header fields, the header format for the connection establishment phase is further augmented with the following fields: (i) number of transmitting pipes to be desirably used (nTx), (ii) number of receiving pipes that can be used (nRx), (iii) list of IP addresses corresponding to nTx ($ipTx$), and (iv) list of IP addresses corresponding to nRx ($ipRx$). The nTx field is the number of pipes the source would ideally like to use for its transmissions (which in effect will require nTx pipes to be maintained at the receiving ends), and the nRx field is the maximum number of pipes on which the source is willing to serve the reverse path. Note that if nTx or nRx equals one, the pTCP connection falls back to a regular TCP connection with single state.

4.2.4 Connection Management

We use the state machine of pTCP and the connection establishment handshake presented in Figure 10 and Figure 11 respectively for the following discussions on connection management. Note that the state machine for TCP-v is the same as that of default TCP, and the interface between pTCP and TCP-v is presented in Section 4.2.2. We assume the number of pipes to be nIF at both ends.

- *Establishment:*

  When an active open is issued by the client application, a pTCP socket with a transmission control block (TCB) similar to TCP's TCB, but with the additional state variables introduced

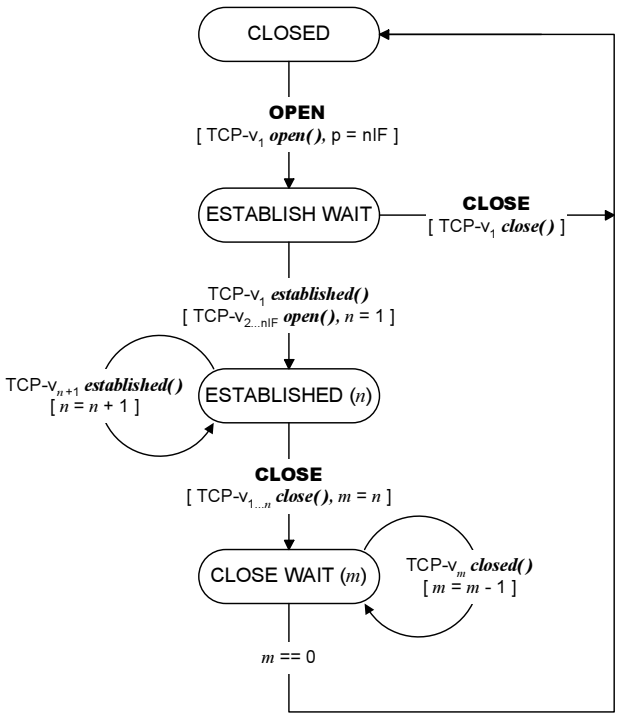


Figure 10: pTCP State Machine

earlier in Section 4.2.1, is created. After the pTCP socket is created, pTCP creates one TCP-v TCB and issues the $open()$ call to it. When the TCP-v *SYN* packet is sent out, pTCP sets *nIF* in the nTx field and the corresponding IP addresses in $ipTx$, and appends additional pTCP connection management header information to the packet. When the pTCP at the server end receives the *passive* open, it checks to see if it can support *nIF* TCP-v pipes.[1] Assuming that the receiver can support the required number of pipes, it creates the first TCP-v TCB, issues the passive open to it, and in the process takes it to the **SYN_RCVD** state [112]. When the *SYN+ACK* is sent out by the first TCP-v at the server end, the destination IP address is appropriately set based on the information received in the first *SYN*, and the source address reflects the local interface the first TCP-v pipe is bound to. The *SYN+ACK* message carries *nIF* in the *nRx* field that the server has agreed to support, and the corresponding IP addresses in $ipRx$.

When the client pTCP receives the *SYN+ACK*, it creates the remaining $nIF - 1$ TCP-v TCBs

[1]There might be several reasons including memory or processor limitations, security considerations, etc., because of which the receiving host might desire to limit the number of pipes in the case of bandwidth aggregation.

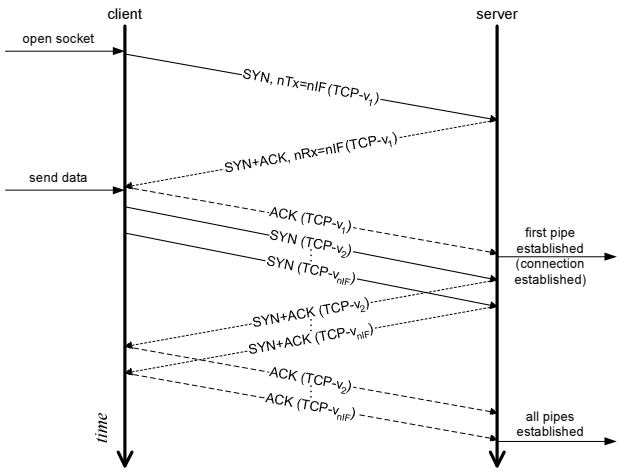


Figure 11: pTCP Connection Establishment Handshake

and issues $open()$ calls to each of them. Also, the first TCP-v pipe at this stage enters the **ESTABLISHED** state after sending back an *ACK* to the server. pTCP thus goes into the **ESTABLISHED (1)** state and can start accepting data from the application. *Hence, even if some of the pipes are experiencing connection setup problems, pTCP will still ensure data flow between the client and the server.*

The source IP address of each of the outgoing *SYN*s is set to the local interface the TCP-v pipe is bound to. The destination address is set to one of the addresses in $ipRx$ in the *SYN+ACK* sent from the server. When the first TCP-v pipe at the server receives the *ACK*, it enters the **ESTABLISHED** state and can thus participate in the data exchange with the client. The pTCP at the server end also enters the **ESTABLISHED (1)** state. When the server receives the *SYN* messages from each of the remaining $nIF - 1$ TCP-v pipes, it creates the corresponding TCP-v TCBs and assigns the respective *SYN*s to the TCBs, taking each of them to the **SYN_RCVD** state. From there on, the exchange of information between each server TCB and the corresponding client TCB is similar to that of TCP.

As and when each of the individual TCP-v pipes enter the **ESTABLISHED** state, they issue the $established()$ call to pTCP making pTCP move down the state machine shown in Figure 10. Finally, when all the individual pipes enter the **ESTABLISHED** state, pTCP enters the **ESTABLISHED (nIF)** state.

- *Termination:*

  The teardown of a pTCP connection is relatively simpler than the connection establishment. When an application closes the connection, pTCP uses the *close*() interface to make the individual TCP-v pipes close. Each pipe closes using TCP's regular closing handshake. When a TCP-v pipe enters the **CLOSED** state in its state machine, it invokes the *closed*() callback to pTCP. For every *closed*() message pTCP receives, it appropriately keeps track of the number of closed TCP-v pipes. Upon successful completion of all TCP-v pipes, pTCP enters the **CLOSED** state of Figure 10 and confirms the close to the application layer.

4.2.5 Congestion Control and Flow Control

pTCP by itself does not perform any congestion control. The individual TCP-v pipes are solely responsible for controlling the amount of data transferred through each pipe. On the other hand, flow control in pTCP is performed at the pTCP layer. While the primary reason is the fact that pTCP has control over the receive buffer, it also helps in better utilization of the buffer across the multiple pipes. For example, in the case of the unaware application approach, irrespective of the BDP of the individual TCP pipes, each pipe would have a constant buffer (of 64KB by default). This will result in wastage of buffer space for pipes with smaller BDPs and wastage of capacity for pipes with larger BDPs. However, in pTCP the buffer space will be shared by the individual pipes based on their respective BDPs. Note that this property can also be achieved using some approaches proposed in related work [94].

We assume the buffer space (both send buffer and receive buffer) available at the pTCP layer is equal to $n * B$, where n is the number of TCP-v pipes, and B is the default TCP buffer size. Every segment that belongs to a pTCP connection always carries the available space in the pTCP receive buffer, irrespective of which pipe it belongs to. The pTCP sender keeps track of the number of outstanding bytes for the connection, and ensures that the receive buffer never overflows. Although all individual TCP-v pipes see the same available buffer space and hence can contend simultaneously for that space (provided there is space in their congestion windows), since pTCP has control over all data transmissions, it prevents any excess data from being transmitted. For example, consider a scenario in which the receiver has advertised a window size of 1000 bytes. Assuming that there

exist three TCP-v pipes at the sender and each of them has 1000 bytes space left in the congestion window, each of the pipes will attempt to transmit 1000 bytes worth of data. However, except for the first pipe that succeeds in transmitting the 1000 bytes, the other pipes would have a *freeze* value returned for their *send*() calls since pTCP would be aware of the global situation.

4.2.6 Reliability

As we discuss in Section 4.2.1, pTCP maintains the bindings between the actual data segments and the TCP-v virtual segments. Once the application data is bound to a particular TCP-v pipe, it is the concerned TCP-v pipe's responsibility to reliably deliver the data to the receiver by using its own (essentially TCP's) reliability mechanism. Therefore, reliable transport of the application data is achieved in pTCP as long as every data segment in pTCP's send buffer is bound to a virtual segment in one of its TCP-v pipe's virtual send buffer. However, note that the binding between the actual data segment and TCP-v virtual segment can be altered when pTCP performs dynamic reassignment or redundant striping. We now discuss how pTCP can still ensure reliability under these two conditions:

- *Dynamic Reassignment:*

  Whenever pTCP performs dynamic reassignment for a particular TCP-v pipe, it unbinds a data segment from that pipe, and reassigns (binds) it to the next available pipe (which can be the same pipe). From then on, the new pipe will assume the responsibility of reliably delivering the reassigned data segment to the peer TCP-v. When the old pipe "retransmits" the previously bound virtual segment, it will be reassigned a different data segment.

- *Redundant Striping:*

  Redundant striping in pTCP is a special case of the dynamic reassignment where the old pipe still keeps a copy of the data segment. Since the data segment will be bound to a new pipe, its reliable delivery will be guaranteed by the new pipe. However, since the old pipe will also attempt to deliver the same data segment to its peer, there might be duplicates at the receiving pTCP. Such duplicates can be easily detected via the sequence number field ($pSEQ$) in the pTCP packet header.

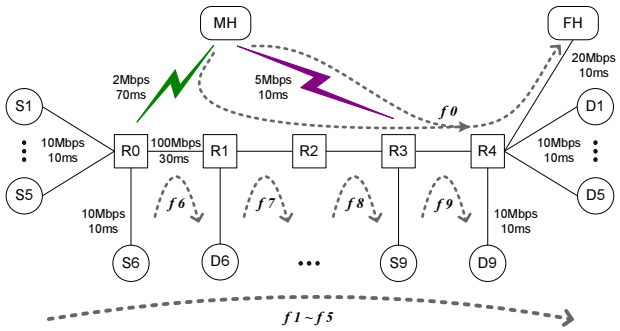


Figure 12: Network Topology for Evaluating pTCP

We have thus far described the key components of the pTCP protocol. In the next section, we present performance of pTCP and compare it with the performance of both the simple and sophisticated application layer techniques.

4.3 Protocol Evaluation

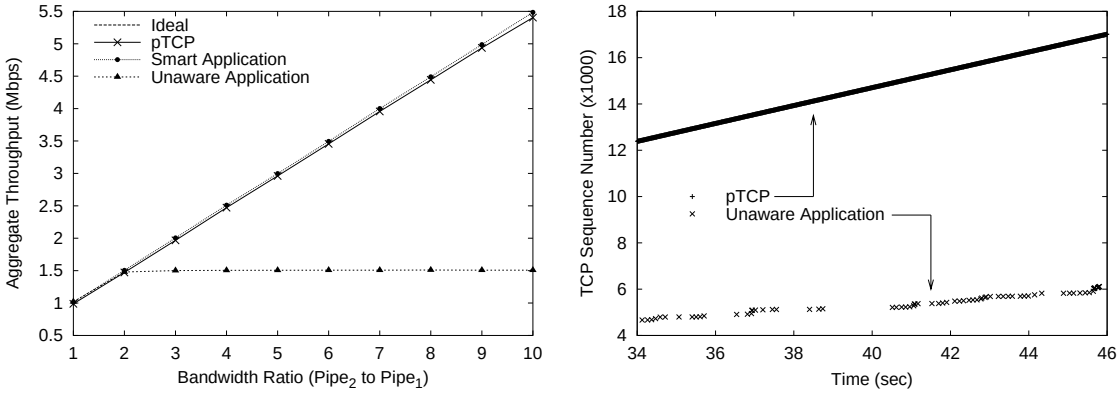
We evaluate the performance of pTCP using the *ns-2* [105] network simulator and the network topology shown in Figure 12. The network topology consists of 5 backbone routers (R0 to R4) and 20 access nodes. For simplicity the backbone routers also double up as wireless access points (or base stations) that the mobile host (MH) can use to communicate with a fixed host (FH) in the backbone network. The mobile host primarily uses two different types of wireless links: (i) link MH–R0 with bandwidth equal to 500Kbps or 2Mbps (depending upon the scenario) and 70ms access delay, and (ii) link MH–R3 with bandwidth ranging from 500Kbps to 5Mbps (depending upon the scenario) and 10ms access delay. Together with the delay experienced in the backbone network, these values simulate a WWAN connection (through a macro-cell or a pico-cell) with 400ms round-trip time, and a shared WLAN connection with 100ms round-trip time respectively. We introduce random bandwidth fluctuations (between 20% and 100% of the maximum link capacity), blackouts (as long as 25 seconds), and random losses (from 0.01% to 1% packet loss rate) in the wireless links to capture the link characteristics. Besides the concerned flow ($f0$) between MH and FH, we also include 9 other flows ($f1$ to $f9$) to simulate the background traffic in the backbone network: 5 long TCP flows with 280ms round-trip time (S1–D1 to S5–D5), and 4 short UDP flows with 10Mbps data rate and 100ms round-trip time (S6–D6 to S9–D9) as shown in the figure.

To observe the effect of bandwidth aggregation, we consider a backlogged application at the

mobile host that uses active network interfaces to perform data striping, and measure the throughput delivered to the peer application at the fixed host. We compare the performance of pTCP with two application layer striping techniques: an unaware application and a smart application. The *unaware* approach, as described in Section 3.2, represents the simplest form of the application layer striping approach where the sending application writes to each socket in turn until blocked, and the receiving application reads from each socket only in-sequence packets. On the other hand, the *smart* approach relies on a sophisticated application to perform bandwidth estimation of the end-to-end path, such that it can stripe according to the ratio of the available data rates of individual pipes. For reference, we also present the "ideal" performance of bandwidth aggregation where we add up the individual throughputs of independent TCP connections, one each for every interface, that do not experience any head-of-line blocking. We use TCP-SACK for all TCP flows including the TCP-v pipes in the simulation. Unless otherwise specified, all data points are averaged over 10 samples using random seeds, and each sample is run for a period of 600 seconds. We present the following results for different striping approaches in the rest of the section: (i) scalability with respect to rate differential, (ii) scalability with respect to the number of pipes, (iii) resilience to rate fluctuations, (iv) resilience to blackouts, and (v) co-existence of different congestion control schemes.

4.3.1 Rate Differential

We show in this section the impact of rate differential between the two pipes of the aggregate connection. We fix the bandwidth of one of the wireless links to 500Kbps (link MH–R0), and increase the bandwidth of the other link from 500Kbps to 5Mbps in increments of 500Kbps (link MH–R3). Note that since the wireless link is the bottleneck of the end-to-end path, the data rate at which the pipe can send is equal to the bandwidth of the wireless link. Figure 13(a) shows the ideal aggregate bandwidths and the performance of the three different striping approaches when the rate differential between the two pipes increases. The x-axis value represents the bandwidth ratio of the second pipe to that of the first pipe, while the y-axis value shows the aggregate throughput achieved at the receiving application. It is clear from the figure that both pTCP and the smart application achieve near ideal performance, but the unaware approach performs significantly worse and exhibits a non-increasing aggregate throughput beyond a ratio of 2.



(a) Aggregate Throughput

(b) Sequence Number Progression

Figure 13: Scalability with Rate Differential

The non-performance of the unaware application, while explained in Section 3.2, can be further illustrated by the results presented in Figure 13(b), where we compare the sequence number progression of the faster pipe (the bandwidth ratio of the two pipes is set to 6) in the unaware application and pTCP during a small time window. We observe that in the unaware application, the head-of-line blocking at the receiver due to the slower pipe holding up packets stalls the faster pipe. The faster pipe thus exhibits distinct idle periods (e.g. 36s, 38s, and 40s in Figure 13(b)), resulting in the degraded performance of the unaware application. One way to avoid such head-of-line blocking is to perform bandwidth estimation at the application and stripe data according to the bandwidth ratio of the two pipes, as demonstrated by the performance of the smart application. In contrast, pTCP by virtue of its congestion window based striping, frees itself from re-implementing the bandwidth estimation already performed by TCP's congestion control, and achieves the same ideal performance.

As we discuss in Section 3.2, another way to improve the performance of the unaware application is to reduce the coupling due to head-of-line blocking between the faster and slower pipes by increasing the size of the resequencing buffer at the application. Equivalently, we can increase the size of the socket resequencing buffer in the faster pipe while keeping that of the slower pipe fixed. We show in Figure 14 the effect of using a larger buffer size in the faster pipe (an x-axis value of 1 indicates the default buffer size) when the bandwidth ratio of the two pipes is 10. It can be observed

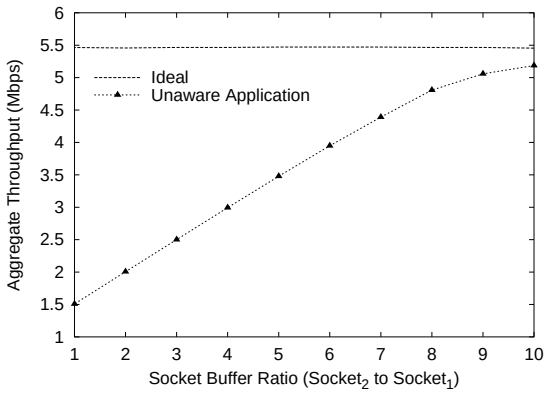


Figure 14: Buffer Requirement of the Unaware Application

that the unaware application will be able to achieve close to the desired aggregate bandwidth only when the buffer over-allocation ratio is at least the ratio of rate differential. Such a requirement on the buffer size can significantly limit the scalability of this approach.

4.3.2 Number of Pipes

We now show that the performance observations made in the previous section still hold true when the number of pipes in the aggregate connection increases beyond 2. We use the same network topology as shown in Figure 12, but increase the number of network interfaces at the mobile host from 2 to 5. Since the mobile host opens one pipe for each active network interface, it incrementally adds the following pipes to the aggregate connection: (i) $pipe_1$ with 2Mbps bandwidth and 400ms round-trip time (via MH–R0 link), (ii) $pipe_2$ with 5Mbps bandwidth and 100ms round-trip time (via MH–R3 link), (iii) $pipe_3$ with 500Kbps bandwidth and 300ms round-trip time (via MH–R1 link), (iv) $pipe_4$ with 1Mbps bandwidth and 200ms round-trip time (via MH–R2 link), and (v) $pipe_5$ with 10Mbps bandwidth and 50ms round-trip time (via MH–R4 link).

Figure 15 presents the aggregate throughput enjoyed by the application at the fixed host as the number of pipes increases. We compare the performance of ideal bandwidth aggregation and that of the three striping techniques and conclude that the performance of pTCP scales well with increasing number of pipes. Note that although the smart application is also able to achieve the desired bandwidth aggregation, the overheads incurred at the application due to bandwidth estimation also increase with the number of pipes (bandwidth estimation is performed on a per-pipe basis). While it is clear the unaware application cannot achieve the desired performance, we observe that when the

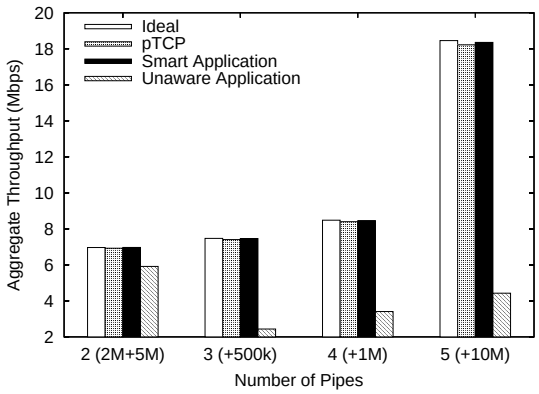
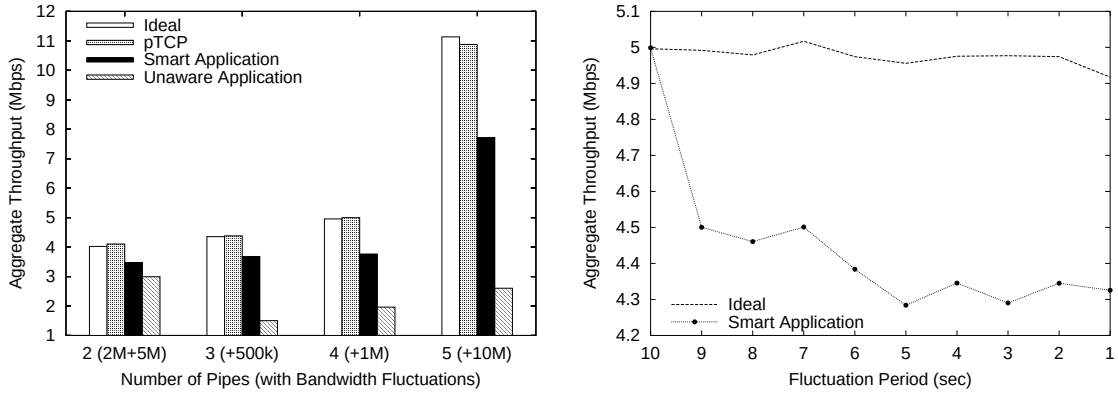


Figure 15: Scalability with Number of Pipes

third pipe (500Kbps) is added to the striped connection, the aggregate throughput of the unaware application even "degrades" to a value lower than that of using only two pipes! This observation further substantiates our argument in Section 3.2 that the performance of the unaware application will be throttled by the slowest pipe of the aggregate connection. Hence it is not always advantageous for the unaware application to use as *many* pipes as possible. For the scenario shown in Figure 15, the unaware application can make use the bandwidths of only two pipes.

4.3.3 Rate Fluctuations

Since the characteristics of wireless links exhibit large variances, in this section we investigate the performance of pTCP in the presence of fluctuations in the available data rate of individual pipes. We emulate the bandwidth fluctuations of a wireless link by periodically changing its link bandwidth to a random value. Specifically, for a link k with a maximum capacity of C_k, its bandwidth will change every T_k seconds to a value randomly chosen from $[0.2C_k, C_k]$ and rounded to the nearest tenth of C_k. We consider the same topology as the one used in Section 4.3.2 where the mobile host is equipped with multiple network interfaces ranging from 2 to 5. Each pipe has the same round-trip time as before, but now with random data rate fluctuations. The fluctuation period of each pipe is to 3, 10, 6, 20, and 5 seconds respectively. Figure 16(a) shows the ideal performance of bandwidth aggregation and the three striping techniques as the number of pipes with rate fluctuations increases. We note that Figure 16(a) mirrors Figure 15 with the ideal aggregate throughput scaling down to 60% due to bandwidth fluctuations (the average bandwidth for a link with capacity C_k is $0.6C_k$ in our link fluctuation model). It is clear that even under such a dynamic environment, the performance of



(a) Throughput vs. Number of Pipes

(b) Coarse-Grained Bandwidth Estimation

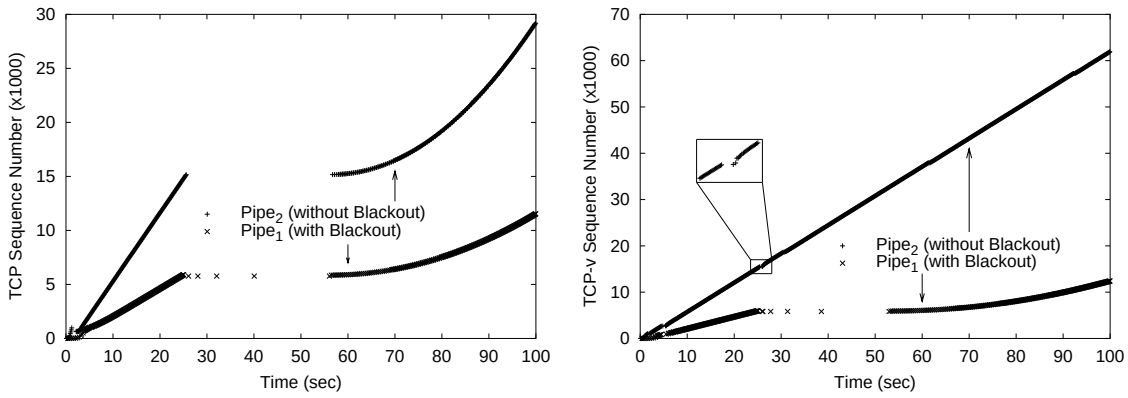
Figure 16: Impact of Rate Fluctuations

pTCP still closely follows that of the ideal performance. This is however not the case for the smart and unaware applications. Specifically, compared with Figure 15 we find that the performance of the smart application drops significantly, demonstrating the inefficiency when its bandwidth estimation mechanism cannot accurately track the actual fluctuations of available data rates.

We use Figure 16(b) to further substantiate this argument, where we show the performance of the ideal bandwidth aggregation and smart application in a two-pipe scenario. The first pipe has 2Mbps bandwidth and 400ms round-trip time, while the second pipe has 5Mbps bandwidth and 100ms round-trip time. We fix the bandwidth of the first pipe, but introduce bandwidth fluctuations in the second one. We assume the smart application performs a coarse-grained bandwidth estimation such that it is able to obtain correct bandwidth estimates of the second pipe every 10 seconds. Figure 16(b) shows that as the fluctuation period of the second pipe decreases from 10 seconds to 1 second, the performance of the smart application degrades drastically, even when there is only a slight mismatch between the bandwidth estimation interval and the actual bandwidth fluctuation period. This simulation result corroborates the theoretic analysis we made in Section 3.3.3 regarding the sensitivity of the application striping model to the accuracy of the striping algorithm.

4.3.4 Blackouts

In this section, we show the impact of blackouts on the performance of pTCP and application striping techniques. We use a two-pipe scenario, where the first pipe has 2Mbps bandwidth and 400ms



(a) Smart Application during Blackouts

(b) pTCP during Blackouts

Figure 17: Impact of Blackouts

round-trip time, and the second one has 5Mbps bandwidth and 100ms round-trip time. We introduce a long blackout in the first pipe between 25s and 50s when the available bandwidth decreases to zero, and all packets transmitted during such period are dropped. The second pipe however does not experience any blackout. Figure 17(a) shows the TCP sequence number progression (at the mobile host) of both pipes in the smart application. It is obvious that during the blackout period, the first pipe stops sending data after repeated failures and starts bandwidth probing through exponential backoffs. However, because of the head-of-line blocking problem described in Section 3.2, even the second pipe which does not experience blackouts stalls almost for the entire duration of the blackout period, resulting in the aggregate connection coming to a standstill. On the other hand, as seen in Figure 17(b), in pTCP, although the first pipe stalls during the blackout period as in the smart application, the second pipe continues to progress after a minor stall. This is possible because of the redundant striping policy in pTCP that reassigns the first segment within the congestion window of the first pipe to the second pipe and prevents the latter from experiencing head-of-line blocking. pTCP thus shows the advantages of using the aggregate connection to improve reliability in the face of link failures, which is not observed in the smart application. While the smart application can potentially use some mechanisms to handle blackouts, this will impose added complexity at the application. Not only will the application need to detect the presence of blackouts through appropriate feedback mechanisms, but it will also have to “pull” back segments that are already in

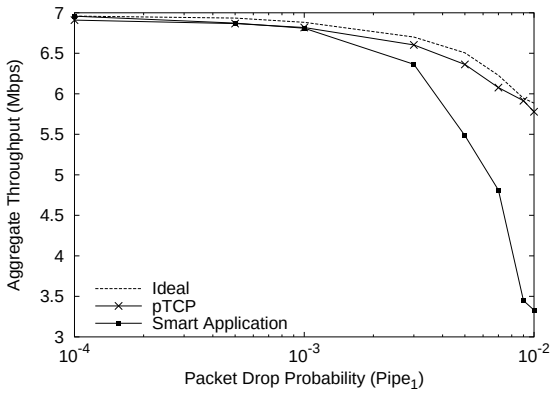


Figure 18: Multiple Congestion Control Schemes

the pipe experiencing a blackout for reassigning to the other pipe.

4.3.5 Different Congestion Control Schemes

So far we have only considered a pTCP connection when its TCP-v pipes use the same congestion control scheme. In this section, we demonstrate the ability of the pTCP protocol to use two different congestion control schemes within the same aggregate connection. The need for different congestion control schemes across multiple pipes is because approaches to improve the performance of TCP over wireless networks have been tailored to the characteristics of the wireless network in consideration for optimal performance. For example, WTCP [97] has been proposed for wireless WANs with very low bandwidths and reverse path congestion, while STP [49] has been proposed for satellite networks with highly asymmetric links and long propagation delays. Therefore, when a mobile host has multiple interfaces, a conceivable scenario is to use interface-specific congestion control schemes for achieving optimal performance.

We consider the same two-pipe scenario as the one used in Section 4.3.4. We introduce random losses to the first pipe by inserting a loss module in the MH–R0 wireless link (refer to Figure 12). The module inserts losses at packet error rates ranging from 0.01% to 1%. We consider the performance of pTCP when the first pipe uses TCP-ELN [10] and the second pipe uses regular TCP (we do not explicitly introduce losses in the second pipe). TCP-ELN receives explicit loss notification from the underlying link layer when a packet is dropped due to random wireless losses, and does not react to such losses. Note that we use TCP-ELN to emulate an intelligent congestion control

scheme that can differentiate the wireless random losses from congestion losses. It is not the sophistication of the congestion control scheme used that is of key importance, but the ability of the pTCP protocol to accommodate two different congestion control schemes within the same framework. Figure 18 shows the performance of pTCP when the loss rate in the first pipe increases from 0.01% to 1%. It is clear that pTCP is able to achieve the ideal aggregate bandwidth (sum of the throughputs of a TCP-ELN connection over the first pipe and an independent TCP connection over the second pipe) for all cases, illustrating the seamless nature in which pTCP allows the two congestion control schemes to co-exist. For comparison we also show the performance of the smart application in the same scenario. We observe that when the packet loss rate is low, the smart application is able to achieve the ideal performance. However, as the loss rate increases, the fact that TCP-ELN becomes ineffective in recovering from the random losses in the first pipe (the ideal curve goes down as loss rate increases) introduces head-of-line blocking at the receiver, and stalls the second pipe in the smart application. (For example, the aggregate throughput of the smart application when the loss rate is 1% is lower than the bandwidth of the second pipe.)

While we have shown the ability of pTCP to accommodate multiple congestion control schemes in one connection, the need to support different congestion control schemes due to the heterogeneity of wireless links in fact has a more significant impact on the scalability of the protocol design. We elaborate on the problem and solution in Chapter 5.

CHAPTER 5

TRANSPOSITIONALITY

The principle of parallelism allows existing transport layer protocols designed for a single path to operate seamlessly over multiple pipes for facilitating network handoffs, server migration, and bandwidth aggregation. A transport layer solution coupled with the principle of parallelism thus can enable transparent host mobility in heterogeneous wireless without relying on the support from the underlying network infrastructure. While such a solution is immune to the heterogeneity of the network infrastructure by appropriately conferring the task of mobility support on the end points (server and mobile host), it still requires support from the server that is specific to the wireless access technology used. In particular, various transport layer protocols tailored to the characteristics of different wireless access technologies have been proposed that show significant performance benefits in the target environment [10, 49, 97]. However, in order for the mobile host to enjoy the performance gains of using network-specific protocols, the server needs to be changed to participate in the protocol operations. In other words, the server needs to be made aware of the wireless technology used at the mobile host – which can be undesirable since such heterogeneity is local to the mobile host. Therefore, an ideal solution that can scale with the increasing heterogeneity of wireless networks is to tackle the heterogeneity *locally* at the mobile host without exposing it to the backbone server. In this chapter, we propose *transpositionality* as the second fundamental principle that needs to be incorporated in transport layer design to address network heterogeneity. We first explain transpositionality and and motivate the need for a transpositional transport protocol. We then apply the principle of transpositionality to TCP and arrive at a new protocol called **RCP** (Reception Control Protocol) that effectively addresses the server scalability issue mentioned before. Finally, we apply the principle of transpositionality in tandem with the principle of parallelism to propose a new transport protocol called **R^{2}CP** (Radial RCP). We show how such a transport protocol can provide a truly transparent mobility support in heterogeneous wireless networks, without relying on any support from the network infrastructure and the server specific to the wireless technology used.

5.1 Motivation

A transpositional transport protocol is one that can dynamically redistribute the intelligence of the protocol functionalities to the sender or the receiver for achieving the most effective operation. In the context of heterogeneous wireless networks, for example, a transpositional transport protocol should ideally transpose its protocol functionalities tailored to the characteristics of the wireless environments (e.g. *wireless-aware* congestion control and loss recovery) to the end point that is adjacent to the wireless link, namely the mobile host. TCP (and its numerous variants) is a *sender-centric* transport protocol where the (data) sender performs all important tasks including congestion control and reliability. The receiver participates in the operation of the protocol, but contributes only by sending feedback in the form of acknowledgments. When the mobile host acts as the source uploading data to the server, it is in charge of congestion control and reliability and hence can use the lower layer feedback about the wireless link (e.g. link bandwidth or link loss) for intelligent, wireless-aware operations. However, in the more prevalent scenario when the server acts as the source sending data to the mobile host, the sender-centric nature of TCP may make it suffer from suboptimal protocol operations due to the server having no direct access to the feedback information. In other words, an ideal protocol should allow the mobile host to drive the operations of the protocol through transposition of protocol functionalities, irrespective of the mobile host being the sender or the receiver.

In this section, we motivate why TCP should be made transpositional to exhibit the *receiver-centric* behaviors when mobile hosts act as receivers for data sent from servers in the backbone network. While we explain in Section 5.2 various protocol functionalities that can be moved from the sender to the receiver, for purposes of discussions in this section, we assume that a receiver-centric transport protocol controls *how much data can be sent*, and *which data should be sent*, by the sender. The sender merely acts based on the requests from the receiver. We first discuss the *performance gains* for the mobile host by dealing with the characteristics of the wireless last-hop, and then discuss the *functionality gains* when the mobile host is equipped with multiple heterogeneous wireless interfaces. For simplicity, we use the terms “receiver” and “mobile host” interchangeably in the following discussions.

5.1.1 Tackling the Wireless Last-Hop

Numerous TCP variants and alternatives have been proposed for mobile hosts operating in a wireless environment [9, 10, 11, 49, 97]. A common theme between these transport protocols is the notion of addressing the problems induced by the wireless last-hop. Despite the wireless-aware behavior of these transport protocols, the congestion control and reliability mechanisms of the connection are still predominantly controlled by the sender. In the following, we discuss why placing the transport protocol's intelligence at the mobile host can enable fundamentally smarter mechanisms for congestion control, loss recovery, and power management when compared to sender-centric approaches.

5.1.1.1 Loss Recovery

TCP assumes that all losses are due to congestion, and hence it invokes its congestion control mechanisms when recovering from losses. In the presence of non-congestion-related losses introduced by wireless links such as channel errors, delay variations, blackouts, and handoffs, TCP suffers from performance degradation due to unnecessary window cutdowns. Hence, many approaches proposed to improve the performance of TCP in wireless environments have focused on providing TCP with information about the characteristics of the wireless link for it to distinguish the causes of losses and take appropriate actions. The information can be in the form of loss classification (whether a loss is due to congestion or corruption), RTT sample filtering (excluding RTT samples adversely inflated due to link retransmissions), channel states or potential link outages (handoffs or blackouts), etc [9, 11, 13, 39].

Since the mobile host is adjacent to the wireless last-hop, it is obviously better equipped to obtain first-hand knowledge of the above pieces of information. In TCP, since the loss recovery (including loss detection) is performed at the sender, the mobile host needs to convey the requisite information to the server for it to take "wireless-aware" actions. While this model of operation has predominantly been adopted in related work, it has some key limitations: (i) Providing feedback to the sender incurs a finite overhead in terms of the throughput consumed on the reverse path. This can translate into degraded performance for connections, especially when the forward and reverse traffic shares the same bottleneck channel (as is the case for the wireless last-hop), or when the

feedback is lost. (ii) Providing all available information as feedback within a limited transport protocol framework can be unwieldy to achieve. For example, some mobile hosts might use a reliable link layer that affects the round-trip time of the connection, and hence might choose to feedback information to filter specific RTT samples. Other mobile hosts might have an unreliable link layer, but can provide feedback information about the reasons for losses (random or congestion-related). If transport protocol headers need to be changed to accommodate such information, how can the changes be made generic enough to accommodate any possible feedback information? (iii) Along the same vein, how can a sender be designed generically to operate with potentially a wide variety of such types of feedback coming from mobile hosts that use any arbitrary link layer protocol?

A receiver-centric transport protocol that performs loss recovery at the receiver, however, can *avoid the feedback overheads and latency*, and be responsive to the dynamics of the wireless link using the information obtained locally. Moreover, while any intelligence added to sender-centric approaches requires changing both the backbone server (for reaction) and the mobile host (for feedback), a receiver-centric approach involves *changing only the mobile host*. The backbone server that is not in charge of loss recovery does not need to be aware of the characteristics of the wireless link.

5.1.1.2 Congestion Control

The congestion control mechanism that TCP uses is designed for wired environments, without taking into consideration the characteristics of wireless environments. Related work that aims to achieve optimal performance in various wireless environments has proposed different congestion control mechanisms tailored to the characteristics of the specific target environment [49, 74, 97, 108]. For example, WTCP [97] has been proposed for wireless WANs with very low bandwidths and reverse path congestion, while STP [49] has been proposed for satellite networks with highly asymmetric links and long propagation delays.

To achieve optimal performance, a mobile host should ideally use the congestion control mechanism (or transport protocol) designed for the specific wireless network it has access to. However, in sender-centric approaches, these network specific congestion control mechanisms need to be implemented at the backbone server. While it is conceivable that a mobile host has access to only a very limited number of wireless networks, a backbone server may need to support a significantly large

amount of connections from mobile hosts belonging to any arbitrary wireless network. Given the increased heterogeneity of the wireless networks, the disadvantages of sender-centric approaches is pronounced in terms of its lack of deployability. *Not only is it infeasible for the server to implement all possible congestion control mechanisms designed for various wireless environments, but it is unscalable to require the server to change its protocol stack whenever a new congestion control mechanism optimized to a new wireless access technology is introduced.*

A receiver-centric protocol where the receiver is responsible for congestion control thus has unique advantages over a sender-centric one. Since the sender is not tasked with implementing the congestion control mechanism of the connection, its functionality can be significantly simplified and made transparent to the specific congestion control mechanisms used at the receiver.

5.1.1.3 Power Management

While a majority of work on the performance of TCP has focused on the throughput achievable, recently the energy efficiency of TCP has also gained attention [96, 107, 120]. It is shown in [107, 120] that since channel errors tend to be bursty (correlated), it is energy-conserving to cut down the window size (and hence reduce the number of packets in flight) when wireless losses are detected. This is because packets retransmitted immediately after wireless losses are likely to be lost again, thus wasting the energy. While TCP-SACK achieves better throughput performance compared to other TCP variants, in fact it is the least energy-conserving protocol of all when the channel error rate is high [96].

Therefore, an energy-efficient transport protocol should avoid persistently accessing the channel when the channel condition is hostile, as energy consumed during this period for attempting to transmit or receive packets is likely to be wasted. Instead, it should adjust the retransmission policy according to the channel dynamics. While it is possible to implement such power management in a sender-centric transport protocol like TCP, there are several limitations to this approach: (i) While the receiver is more aware of the channel condition than the sender, any power-saving decision cannot be made locally at the receiver. This is because the sender is responsible for congestion control and loss recovery, and hence any “unexpected” prolonged delay incurred at the receiver (that decides to refrain from accessing the channel until the channel condition is more favorable)

in receiving data packets or transmitting *ACKs* can easily cause the sender to timeout or wrongly inflate its RTT estimation. (ii) Even if the receiver decides to inform the sender of the power-saving decision, the feedback information will suffer from the same problems that we discussed in Section 5.1.1.1. More importantly, packets transmitted for conveying such feedback information incur extra energy consumption – especially if the channel condition is bad such that multiple retransmissions are required. The overheads incurred in sending the feedback information hence limit the granularity and effectiveness of any sender-centric power management scheme.

On the other hand, in a receiver-centric protocol the receiver decides which and how much data it needs to receive, and the sender merely responds based on the receiver's direction. Efficient power-conserving decisions can be made at the receiver without triggering any adverse reaction at the sender. Hence, the receiver has a higher degree of flexibility to control the transmission or retransmission decisions, without involving the sender.

5.1.2 Supporting Heterogeneous Interfaces

The primary reason for a mobile host to be equipped with heterogeneous wireless interfaces, as we discussed in Section 2.1, is the performance tradeoffs that different access technologies exhibit, in terms of mobility support, coverage area, network capacity, and transmission power. The availability of heterogeneous interfaces, however, has given rise to new challenges to existing transport protocols in terms of the functionalities they provide. In the following, we discuss the *functionality gains* that a receiver-centric transport protocol can achieve to leverage the existence of multiple interfaces at the mobile host.

5.1.2.1 Seamless Handoffs

When the coverage areas of different access technologies overlap, it is possible to achieve seamless handoffs at the link layer. However, such link layer handoffs do not necessarily translate into seamless handoffs at the transport layer. Specifically, when a mobile host handoffs from one interface to another with an IP address change handled by Mobile IP, the prolonged delay for registration with the home agent [93] can potentially introduce packet losses after the link layer handoff has completed. To prevent TCP from having adverse reactions due to packet losses during handoffs, the mobile host needs to inform the sender of the handoff decision. As we discussed in Section 5.1.1.1,

whenever feedback information is required, a receiver-centric protocol has advantages over a sender-centric one due to the locality of information needed.

However, while it is possible to *freeze* TCP during handoffs [39], such a stall causes connection disruption and prevents users from enjoying seamless handoffs. One solution to avoid the handoff latency without relying on infrastructure support [31], is to use a mobility-enabled transport protocol for achieving end-to-end host mobility [90]. When the mobile host decides to perform a vertical handoff [102], it can create a new "data stream" for data transfer through the new address, as soon as the new interface becomes active. With an approach like [55], the mobile host can use multiple TCP pipes (streams) simultaneously without experiencing any connection stall as long as the link layer supports seamless handoffs.

A receiver-centric transport protocol thus has advantages over a sender-centric one in such a scenario, since the receiver can accurately control which and how much data to send through each pipe based on the status (say, signal strength) of each interface. Moreover, as we discussed in Section 5.1.1.2, when the receiver decides to switch to another interface specific congestion control mechanism after handoffs, such decision does not need to involve the sender, which otherwise would be tasked with, in addition to supporting a plethora of congestion control mechanisms, the seamless transition from one congestion control mechanism to another for a live connection.

5.1.2.2 Server Migration

Server migration is necessary for achieving service continuity when a mobile host handoffs from one network to another, but fails to connect to the original server using the new network address. For example, consider a mobile host with both WWAN (Access Network A) and WLAN (Access Network B) interfaces as shown in Figure 19. When it initially uses the WWAN interface to connect to the *E! Online* server, the mobile host is provided access to the proxy server inside the WWAN that mirrors the same content (e.g. consider a WWAN service provider such as EarthLink that teams with Akamai for improving the network service it provides [5]). When the mobile host moves to the WLAN and undergoes a vertical handoff, it cannot connect to the original proxy server in the WWAN (due to, say, firewalls). However, it may have a connection to a different server through the WLAN interface, and hence can initiate server migration to enjoy service continuity.

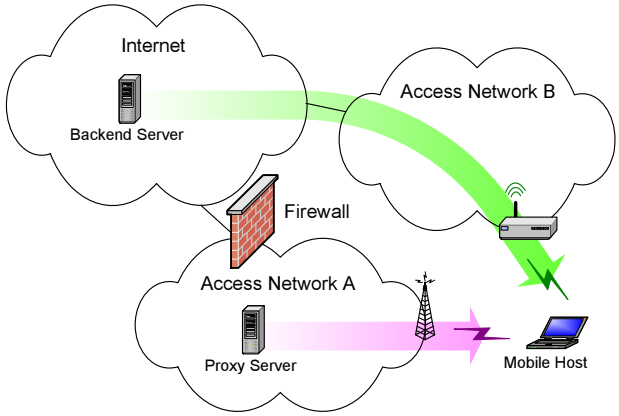


Figure 19: Scenario for Server Migration

Server migration may require support from the application [103] or the transport protocol [100] to synchronize the states between servers. A well-designed transport layer protocol can facilitate such a synchronization process. If a sender-centric transport protocol is used for server migration, states maintained at the server for performing congestion control and loss recovery need to be transferred from one server to another. Moreover, for TCP, after the new server assumes control of the connection, the mobile host has to flush any out-of-order data from its resequencing buffer, to avoid confounding the server by acknowledging data that the sender has not yet sent [100]. As much as a window's worth of data buffered at the receiver needs to be flushed after server migration.

On the other hand, in a receiver-centric transport protocol, since the states maintained for protocol operations are biased toward the receiver, overheads incurred in transferring protocol states from one sender to another are minimized. Moreover, since the receiver has access to the receive buffer and has control over which data to receive from the sender, there is no need to flush the buffer after migration. The receiver can simply request every "hole" in the receive buffer from the new sender.

5.1.2.3 Bandwidth Aggregation

When the coverage areas of different wireless networks a mobile host has access to, overlap, the mobile host can use multiple interfaces simultaneously, with the goal of enjoying the aggregate bandwidth available.

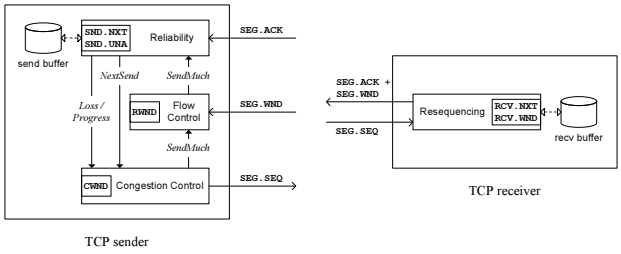
While approaches for achieving bandwidth aggregation on multi-homed mobile hosts using sender-centric transport protocols have been proposed [55, 73], they are limited to using only one

server. Such point-to-point bandwidth aggregation, however, might not be possible or desirable in some cases. For example, as we discussed in Section 5.1.2.2, some proxy servers are accessible only through the designated wireless interface, and hence it is not possible to achieve bandwidth aggregation using additional wireless interfaces that have no access to the existing server. In such a scenario, a sender-centric approach would not be desirable since it would otherwise require explicit coordination between these geographically-spaced servers. Moreover, it is possible that mobile hosts want to leverage the existence of multiple active interfaces in an *opportunistic* fashion, based on the tradeoffs between different interfaces in terms of achieved throughput, power consumption, and the cost incurred. The decision to use or shut down an active interface thus can be dynamic, based on the channel conditions (e.g. loss rates and delays) and the policy of bandwidth aggregation that the mobile host desires.

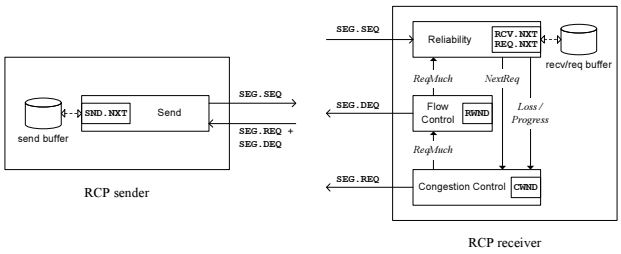
It is advantageous to use a receiver-centric transport protocol for achieving different instantiations of bandwidth aggregation. For multipoint-to-point bandwidth aggregation, since the receiver is the center of control, it can easily coordinate the transmission of multiple senders internally, without any explicit coordination between senders themselves. For policy-based bandwidth aggregation, any policy can be easily implemented and updated at the receiver based on the characteristics of the last-hop and the preference of the user.

5.2 RCP: Reception Control Protocol

We now present details of how TCP, a sender-centric protocol, can be transposed to RCP, a receiver-centric protocol. Briefly, RCP moves the responsibility for performing reliability and congestion control from the sender to the receiver. We first give a short review of the sender-receiver interaction in TCP, and how it is transposed in RCP. We then give an overview of the protocol operation in RCP, and present different protocol functionalities including connection management, congestion control, flow control, and reliability. Finally, we use simulation results to show that while RCP is indeed TCP-friendly, it achieves better performance in wireless environments in terms of intelligent loss recovery, scalable congestion control, and efficient power management.



(a) TCP (Sender-Centric)



(b) RCP (Receiver-Centric)

Figure 20: Sender–Receiver Interactions

5.2.1 Transposition of Functionalities

TCP is a connection-oriented transport layer protocol that provides reliable in-sequence data delivery to the application. Its protocol operation mainly consists of the following four functionalities: connection management, flow control, congestion control, and reliability. Figure 20(a) shows a schematic view of the sender–receiver interaction in TCP, along with several state variables using the notation introduced in [86]. The connection management is required by any connection-oriented protocol to synchronize connection states between the communicating peers. After the connection is established, the sender in TCP controls the progress of data transfer. The sender drains data from its buffer based on the amount of data that the receiver can accept (flow control), and the amount of data that the network can sustain (congestion control). The receiver performs resequencing and acknowledges data received. Reliable data transfer is achieved through loss detection and loss recovery performed at the sender.

It is clear that the connection management cannot be implemented only at one side of the connection, but needs participation of both the sender and the receiver. For the other functionalities, while TCP uses a sender-centric approach, RCP delegates the responsibility to the receiver as shown

in Figure 20(b). Briefly, while the receiver in TCP merely sends back *ACKs* with no control over which and in what sequence data is transmitted by the sender, in RCP the receiver explicitly controls these factors and the reliable delivery of data. Moreover, the RCP receiver also assumes total control over the bandwidth the connection can consume, using the same window based algorithm employed by the TCP sender. Finally, although flow control in TCP involves the sender, it is performed solely by the receiver in RCP. Therefore, the receiver in RCP determines *how much data* the sender can send (via congestion control and flow control), and *which data* the sender should send (via reliability).

5.2.2 Protocol Overview

In RCP, since the control of data transfer is shifted from the sender to the receiver, the *DATA–ACK* style of handshaking in TCP is no longer applicable. Instead, to mimic the self-clocking characteristics of TCP, RCP uses the *REQ–DATA* handshake for data transfer, where any data transferred from the sender is preceded with an explicit request (*REQ*) from the receiver. Equivalently, RCP uses the incoming data to *clock* the request for new data. The sender simply maintains the send buffer with one pointer (`SND.NXT`) indicating the maximum sequence number sent thus far.

After the connection is established, the receiver requests data from the sender based on the size of the initial congestion window. The progression of its congestion window follows the slow start, congestion avoidance, fast retransmit, and fast recovery phases just like in TCP. The key difference in the operation is that any trigger for performing congestion control is inferred based on the arrival (or non-arrival) of *data segments*. For example, a loss is inferred upon the arrivals of three out-of-order data segments – instead of *ACKs*. Upon detection of a segment loss, RCP cuts down its congestion window, and retransmits the corresponding *REQ* asking for the lost segment. Finally, the receiver performs data resequencing, and gives in-sequence data to the application.

5.2.3 Protocol Operations

In the following, we present details of the RCP protocol in terms of the *REQ–DATA* handshake, and different functionalities including connection management, congestion control, flow control, and reliability.

5.2.3.1 REQ–DATA Handshake

In the *DATA–ACK* handshake, TCP uses the cumulative acknowledgment for achieving robustness to losses. To emulate this behavior and tolerate loss in the reverse path, RCP allows the receiver to send request either in a *cumulative* mode or in a *pull* mode, by appropriately setting the pull flag (`PUL`) in the packet header. The receiver by default uses the cumulative mode to requests for new data, and uses the pull mode only for retransmission of requests. When the sender receives a request with the pull flag set, it sends only the data segment indicated in the packet header. Otherwise, the sender cumulatively transmits data from `SND.NXT` that has not been sent yet. Hence, the loss of *REQ* in cumulative mode has similar impact to that of *ACK* loss in TCP. To protect *REQ* in the pull mode from losses, RCP also uses a similar mechanism used by TCP for protecting SACK from losses. The receiver puts the most recent blocks of sequence numbers (we use three blocks as proposed in the SACK option [76]) it requested in the *REQ* header. The sender, in addition to maintaining the send buffer, also maintains a cyclic buffer consisting of the most recent blocks of sequence numbers (three blocks) it sent out. Upon receiving the request from the receiver, the sender checks the consistency between the blocks in *REQ* and its cyclic buffer. Any mismatch is an indication of *REQ* losses, and will be recovered by the sender. Note that a request in the pull mode for a specific data segment will be carried in at least four *REQ*s. We revisit the robustness of *REQ* in Section 5.2.4.1.

5.2.3.2 Connection Management

Just like in TCP, either the RCP sender or the receiver can initiate the connection setup. The setup process consists of the same *SYN – SYN+ACK – ACK* handshake as in TCP. However, once the connection is established, instead of the sender sending the first data segment, the RCP receiver transmits the first *REQ* with the initial sequence number. The sender then transmits the first data segment upon receiving the *REQ*. The connection teardown in RCP also follows that in TCP.

5.2.3.3 Congestion Control

In RCP, the receiver performs congestion control and maintains the congestion control parameters including the congestion window `CWND` and round-trip time information. Since RCP is a TCP clone,

it adopts the window based congestion control used in TCP. The slow start, congestion avoidance, fast retransmit, and fast recovery phases are triggered and exited in the same fashion as in TCP. Note that while the same window adaptation algorithm (additive increase, multiplicative decrease) can be implemented either at the sender or at the receiver for performing congestion control, the semantics of the congestion window and the trigger for window increase or cutdown are different. In TCP, the size of the congestion window limits the amount of unacknowledged *DATA* in the network, and the sender uses the return of *ACKs* to trigger the progression of the congestion window. In RCP, the size of the congestion window limits the amount of outstanding *REQs* in the network, and the receiver uses the return of *DATA* to trigger the progression of the congestion window.

5.2.3.4 Flow Control

Flow control allows the receiver to limit the amount of in-transit data to the available receive buffer space – when waiting for the application to read (and purge) in-sequence data, or waiting for the arrivals of out-of-order data. In RCP, a request is sent out only if the corresponding data, when received, will not cause an overflow of the buffer at the receiver. This can be achieved by creating a "dummy" `sk_buff` [17] (that does not contain any data) in the receive buffer for each data segment requested. New requests are issued as long as new space is created in the buffer. Note that the sender in TCP relies on the window advertisement from the receiver to perform such flow control. However, in RCP since the receiver maintains the receive buffer, and has total control over how much data the sender can send, flow control is internal to the receiver. Interestingly, in RCP the sender may need to perform *reverse* flow control against the receiver to control the amount of data requested by the receiver. This could happen due to data-limited applications (e.g. telnet), or the sender is limited by its processing power to serve future requests.[1] In these cases, the sender can use a window field to advertise the maximum sequence number that the receiver can request. When the RCP receiver hits the limit imposed by the sender, it will enter the persist mode as in TCP [112]. Afterward, it can periodically probe the sender for request permissions, or wait for the explicit window update from the sender.

[1] However, note that an RCP sender is simpler than a TCP sender. A server running as the RCP sender thus can service more users than a server running as the TCP sender, if the bottleneck is the processing power.

5.2.3.5 Reliability

As Figure 20(b) shows, in RCP the resequencing and reliability functionalities are collocated at the receiver. Upon receiving a data segment from the sender, the receiver enqueues the data in the corresponding `sk_buff` (created when its request was transmitted), and updates `RCV.NXT` after the resequencing process. In TCP, since reliability is performed at the sender while resequencing is performed at the receiver, `RCV.NXT` is conveyed as the cumulative ACK to the sender for it to perform loss detection. However, `RCV.NXT` conveys limited information about the state of the receive buffer, and hence early implementations of TCP that rely on the cumulative ACK for performing loss detection, suffer from recovering at most one loss per round-trip time, in addition to incurring frequent timeouts [34]. The SACK option is proposed to address this limitation, using which the TCP sender aims to construct the bitmap of the receive buffer in the "scoreboard" data structure [76]. However, in RCP the receiver has direct access to the receive buffer, and hence it can timely and accurately perform loss detection and loss recovery without relying on the use of SACK. Note that `RCV.NXT` in TCP allows the sender to purge its socket buffer. RCP uses a window field `SEG.DEQ` in the packet header to inform the sender of the highest in-sequence data received so far (which can be calculated at the sender using `SEG.REQ - SEG.DEQ`), thus allowing the sender to purge such data from its send buffer. The window scale option [61] used in TCP can also be applied to RCP in the same fashion.

While RCP can use any loss recovery algorithm optimized to the wireless environment, in the current implementation of RCP we adopt the algorithms proposed in [14, 75]. Briefly, (i) the same threshold in terms of the number of out-of-order arrivals is used for detecting all holes (not just the first one) in the receive buffer; (ii) RCP does not incur a timeout when a retransmitted segment is lost; and (iii) there is no need to clear the receive buffer upon a timeout (whereas a TCP sender using SACK should clear its scoreboard due to the possibility of receiver reneging [76]).

5.2.4 Performance Gains

In this section, we show through simulations the performance gains of a receiver-centric transport protocol over a sender-centric one. We first show that RCP is indeed a TCP clone and is friendly to TCP in the wired environment. We then show that in the wireless environment, RCP achieves better

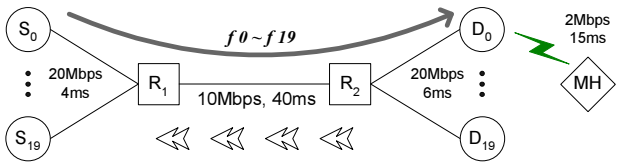


Figure 21: Network Topology for Evaluating RCP

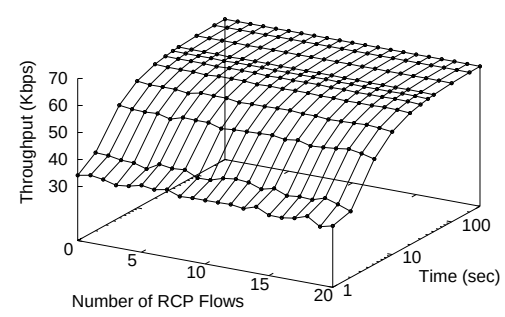
performance than TCP in terms of loss recovery, congestion control, and power management.

We use the *ns-2* network simulator [105] and a dumb-bell network topology shown in Figure 21 for performance evaluation. For fair comparisons between sender-centric and receiver-centric protocols, we modify the implementation of TCP-SACK in *ns-2* to include better loss recovery mechanism such as better estimation of the pipe size [75], and prevention of timeouts due to lost retransmissions (refer to Section 5.2.3.5). Unless otherwise specified, each data point in the figures is an average of 10 samples using random seeds, and each sample is run for 300s.

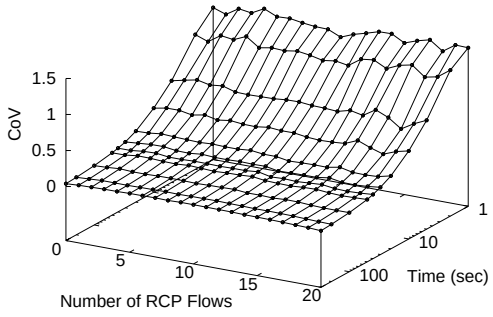
5.2.4.1 TCP Friendliness

We introduce 20 flows between $S_{0...19}$ and $D_{0...19}$ with different proportions of TCP and RCP flows, to study the impact of RCP on existing TCP flows. We vary the number of RCP flows from 0 to 20 (the others are TCP flows), and observe for each scenario the short-term and long-term behaviors of all flows in the network. As shown in Figure 22(a) and Figure 22(b), we plot the mean and coefficient of variation of the per-flow throughput at different time instants after the simulation starts. The coefficient of variation (CoV) is obtained by dividing the standard deviation of the throughput by the mean throughput [47]. Note that for clarity of presentation, the direction of the time-axis in Figure 22(b) is reversed. We can find from both figures that the impact of introducing RCP flows on existing TCP flows in terms of the throughput re-distribution is minimal, as evident from the "flat" curve across different proportions of RCP flows. Specifically, since CoV is an index of unfairness in the network, it is clear from Figure 22(b) that TCP flows do not suffer from unfairness in the presence of RCP flows. (Otherwise, CoV would have increased with increasing number of RCP flows.)

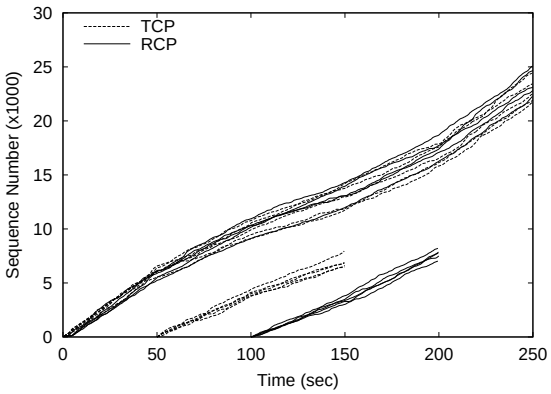
In Figure 22(c) we show the microscopic behaviors of TCP and RCP flows in response to network dynamics. We initially use 5 TCP and 5 RCP flows when the simulation starts. At $t = 50s$ we introduce another 5 TCP flows (terminated at $t = 150s$), and at $t = 100s$ we introduce another



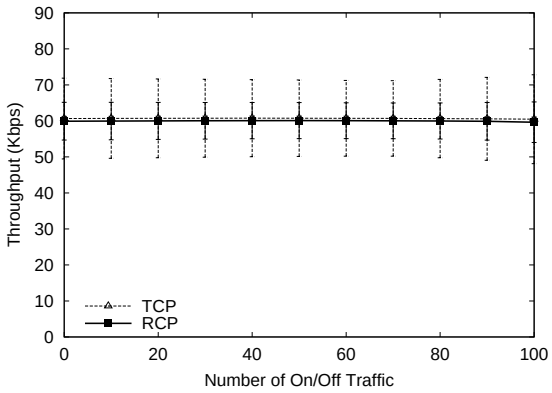
(a) Mean Throughput



(b) CoV



(c) Sequence Number Progression



(d) Reverse Path Loss

Figure 22: RCP is Friendly to TCP

5 RCP flows (terminated at $t = 200s$). We plot the sequence number progression for each of the 20 flows in Figure 22(c), and make the following observations: (i) Throughout the simulation, the sequence number progression plots of the first 5 TCP and 5 RCP flows are always intertwined. (ii) Comparing the 5 TCP flows introduced at $t = 50s$ and the 5 RCP flows introduced $t = 100s$, we find that their sequence number progression plots are similar in terms of the height of their final points, and the width of dispersion (note that while these flows start at different times, they all last for 100 seconds and experience similar network conditions). The microscopic behavior of RCP in response to network dynamics thus is TCP-friendly.

To profile the robustness of the request mechanism used in RCP, we consider a scenario where there is significant loss in the reverse path. As indicated in Figure 21, we introduce 0 to 100 on/off

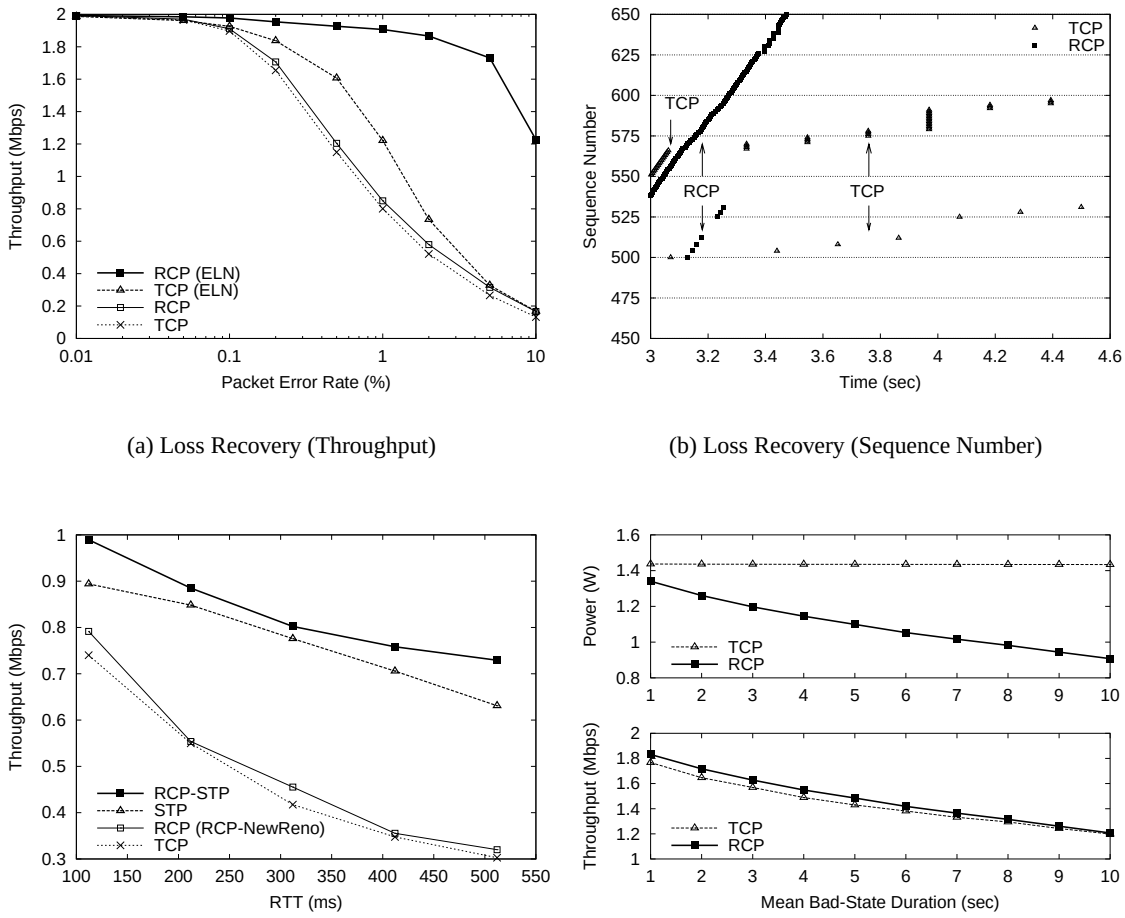
UDP flows in the reverse direction of the bottleneck link to emulate the flash crowds (e.g. WWW-like traffic) in the Internet [47]. These on/off flows generate traffic based on the Pareto distribution, where the shape parameter is set to 1s, the mean idle time is set to 2s, the mean burst time is set to 1s, and the data rate during the burst period is set to 500Kbps. Such flash crowds introduce significant packet drops (to *ACK* or *REQ*) in the reverse path. For example, with 100 on/off traffic sources, each of the TCP (or RCP) flows experiences a packet drop rate of approximately 40% in the bottleneck link. We introduce 20 RCP flows in the forward path, and compare the per-flow throughput achieved against that of using 20 TCP flows. As Figure 22(d) shows, despite the heavy losses in the reverse path, the performance of RCP closely tracks that of TCP. This substantiates our argument made in Section 5.2.3.1 that the design of the cumulative request and the use of cyclic buffer help RCP tolerate losses in the reverse path.

Thus far we have compared the performance of TCP and RCP only in a wired environment. In the following, we consider a mobile host with wireless access to the backbone network. As shown in Figure 21, the wireless link between the mobile host *MH* and the access point D_0 has a bandwidth of 2Mbps, and access delay of 15ms. It allows the mobile host to connect (using either TCP or RCP) to the backlogged traffic source S_0 for, say, file download. We introduce random packet losses from 0.01% to 10% in the wireless link (in both directions), and use the achieved throughput and power consumption at the mobile host to compare the performance of the two protocols.

5.2.4.2 Intelligent Loss Recovery

Information from lower layers about the characteristics of the wireless link allows the transport layer protocol to identify the cause of losses, and hence to intelligently perform loss recovery. As we discussed in Section 5.1.1.1, a key motivation for using receiver-centric protocols at the mobile host is to avoid the feedback overheads and latency seen in sender-centric protocols, and to allow more flexible layer coordination without being limited by the format of the packet header. While it is not the focus of this work to provide assorted instantiations of wireless-aware transport protocols leveraging such benefits, we use the following example to show the performance gain achievable when the information used for loss recovery is locally available.

Explicit loss notification (ELN) [11] has been proposed as a TCP option that allows TCP to



(a) Loss Recovery (Throughput)

(b) Loss Recovery (Sequence Number)

(c) Congestion Control

(d) Power Management

Figure 23: RCP Performance Gains

distinguish wireless random losses from congestion losses. Mobile hosts, with or without the assistance from the base station [11, 13], keep track of packet drops due to wireless errors. When cumulative acknowledgments indicating the packet lost due to wireless errors are generated, the `ELN` flag in the packet header is set by the mobile host. Upon detecting a hole with the `ELN` flag set, the TCP sender retransmits the lost segment without cutting down its congestion window. As we can see in Figure 23(a), the performance of TCP improves substantially for loss rates between 0.2% and 2% when ELN is used. However, when the loss rate increases beyond 2%, the performance gain decreases rapidly. This is because the `ELN` bit allows the sender to identify only one wireless error, and hence when multiple wireless losses occur in one round-trip time, ELN fails to provide the sender with the necessary loss classification information, making the performance of TCP-ELN

degrade to that of vanilla TCP. As we show in Figure 23(b), when packets with sequence numbers 500, 504, 508, 512, 525, 528, and 531 are dropped in the wireless link, it takes TCP-ELN 1.43 seconds to recover from these losses, at most one loss per round-trip time.

RCP with ELN, on the other hand, achieves a much better performance in loss recovery. As shown in Figure 23(b), it takes RCP only 0.12 seconds to recover the same amount of losses. The primary reason is that the wireless loss information maintained at the mobile host is directly accessible to RCP. Hence RCP has accurate information about the cause of losses for all holes in the receive buffer, which allows it to recover from each loss intelligently. While it is possible to couple SACK with ELN, and *redesign* the TCP packet header such that each *un-SACKed* segment has its own ELN flag, this approach in fact exposes the limitations of sender-centric protocols that we mentioned earlier in this section. We note from Figure 23(a) that even in the absence of ELN, RCP constantly achieves better performance than TCP, with the performance gain increasing as the packet error rate increases. The reason, as we discussed in Section 5.2.3.5, is because loss recovery and resequencing in RCP are collocated at the receiver. The effectiveness of the SACK blocks in helping the TCP sender construct the bitmap of the receive buffer, is impaired when both the data segments and *ACKs* suffer from high loss rates (recall that we introduce random losses in both directions of the wireless link).

5.2.4.3 Scalable Congestion Control

As we mentioned in Section 5.1.1.2, various congestion control mechanisms have been proposed for use with different wireless environments. Until a unified congestion control framework is available, to achieve optimal performance, a mobile host needs to use the congestion control mechanism designed for the specific wireless network it has access to. In sender-centric protocols, since congestion control is implemented at the sender, the backbone server is overloaded with supporting a plethora of congestion control mechanisms for all possible wireless networks mobile hosts might connect from. In this section, we present how a receiver-centric transport protocol like RCP can address this problem in a scalable way.

To start with, we consider a satellite environment with long propagation delay and highly asymmetric links, compared to the terrestrial wireless networks. The authors in [49] show that TCP

(SACK) fares badly in such an environment. They propose a new transport protocol called STP (Satellite Transport Protocol) with improved performance. However, STP is a sender-centric protocol like TCP, and hence a mobile host using the satellite network to access the backbone server, cannot use STP unless it is implemented in the protocol stack of the concerned server. Now, by using the algorithm presented in [49], and the technique of functionality transposition discussed in Section 5.2.3, we transform STP into a receiver-centric protocol called RCP-STP. By virtue of the simple sender design in receiver-centric protocols, while STP uses a fundamentally different congestion control algorithm from TCP, the RCP-STP sender uses the same algorithm as the RCP sender. Hence, the backbone server as an RCP sender can communicate with any mobile host acting either as an RCP receiver, or as an RCP-STP receiver – depending on the access network the mobile host uses.

We use the same network topology and scenario used in [49] for evaluating the performance of STP, RCP-STP, TCP and RCP (RCP-NewReno). Briefly, the network topology resembles the dumb-bell topology in Figure 21 with the source (S_0) and destination (D_0) separated by a satellite link (link $R_1 - R_2$). The satellite link is the bottleneck link with a bandwidth of 1.5Mbps, and propagation delay ranging from 50ms to 250ms. Link asymmetry is emulated using backlogged TCP flows in the reverse direction. Four HTTP traffic sources are introduced in each direction to emulate the background traffic. As evident from Figure 23(c), the NewReno style of congestion control used in TCP (and RCP) does not perform well in the satellite environment, while STP and RCP-STP achieve a much better performance. We can make the following observations from the results: (i) the performance difference between TCP and STP substantiates the need to use network specific congestion control for achieving optimal performance; and (ii) the performance difference between STP and RCP-STP reinforces the benefits of receiver-centric protocols over sender-centric ones.

5.2.4.4 Efficient Power Management

We now show the performance of RCP in terms of facilitating power management at the mobile host. As we described in Section 5.1.1.3, when the channel condition is severe, it is not energy-efficient for a mobile host to persevere with persistent retransmissions. Since the mobile host is

an end-point of the wireless last-hop, it is aware of the channel condition (via, say, measuring the signal strength in the received packets or beacons from the access point). Upon detecting a hostile channel state, the mobile host can save the battery power by reducing the amount of data in transit or refraining from transmissions. However, note that while significant energy savings can be achieved by operating the wireless interface card in the sleep mode, doing so without the sender being aware of such energy-conserving tactics may cause adverse reactions at the sender and cause performance degradation [65]. A receiver-centric protocol such as RCP does not have this problem since the mobile host has full control over *how much* data the sender should send.

To evaluate the performance benefits in power consumption, we use the IEEE 802.11b wireless card as a case study. The IEEE 802.11b card consumes 1.65W, 1.4W, and 0.045W when operated in the transmit, receive, and sleep modes respectively [65, 95]. We consider a two-state Markov error model for varying the channel condition of the wireless link [9]. The packet error rate in the good state is set to 0.01% and that in the bad state is set to 10% (for deep fades). The mean duration of the good state is set to 10s, while that of the bad state varies from 1s to 10s depending on the scenario. We assume an energy-frugal mobile host that enters the sleep mode whenever it detects the channel is in the bad state. Once in the sleep mode, all data transmissions and receptions are suspended (hence all packets in transit that arrive during this period are lost). The mobile host wakes up periodically every 100ms to listen to the beacons from the access point [56], during which it also measures the channel state using the received signal strength. The duration of the beacon is 2ms, and the power consumed for receiving the beacon is based on the value assumed for the receive mode. Once the mobile host decides that the channel is in the good state, it *de-freezes* and resumes data transmission as usual.

Figure 23(d) compares the performance of RCP and TCP in terms of power consumption and achieved throughput when the mean duration of the bad state varies from 1s to 10s. We assume that the sender is unaware of the channel state, and hence when TCP is used, the mobile host receives data and transmits *ACKs* irrespective of the channel state. On the other hand, when RCP is used, the mobile host enters and leaves the sleep mode as mentioned before. The mobile host freezes the RCP timer when it enters the sleep mode. When it wakes up, RCP resumes data request based on the state (holes) of the receive buffer. As expected, the longer the mobile host stays in the sleep

mode, the more energy savings it can achieve using RCP. While the energy savings are obvious, Figure 23(d) also shows an interesting result that compares the achieved throughput between TCP and RCP. Since the mobile host suspends all packet transmissions and receptions in the sleep mode, it obviously suffers from throughput loss in terms of giving up the data in transit and giving up the time to use the channel. However, for various conditions of the channel state, the throughput achieved using RCP is in fact no less than that when using TCP. The reason that TCP suffers from a more pronounced performance degradation is due to the adverse reaction of the congestion control mechanism in the presence of severe packet losses.

5.3 R^{2}CP: Radial RCP

We have shown in Section 5.2 that the principle of transpositionality allows TCP to be transposed to RCP for achieving performance gains in terms of intelligent loss recovery, scalable congestion control, and efficient power management. In this section, we combine the principles of transpositionality and parallelism (discussed in Chapter 4) to show how these two fundamental principles can allow transport layer protocols to effectively support transparent mobility in heterogeneous wireless networks. Specifically, we focus on the following functionalities that need to be supported when a mobile host handoffs from one network to another during a live connection: (i) seamless handoffs without relying on infrastructure support, (ii) server migration for achieving service continuity, and (iii) bandwidth aggregation using multiple active interfaces.

In the following, we present how a multi-state extension of RCP at the receiver called R^{2}CP can achieve the desired functionalities, without the requirement of changing the senders as shown in Figure 24. We first discuss the design elements of the R^{2}CP protocol, and then present the architectural overview and protocol details. Finally, we demonstrate the functionality gains achievable in R^{2}CP using network simulation and testbed emulation.

5.3.1 Protocol Design

We discuss in this section the design motivation of R^{2}CP that combines the principles of parallelism and transpositionality in one transport layer framework.

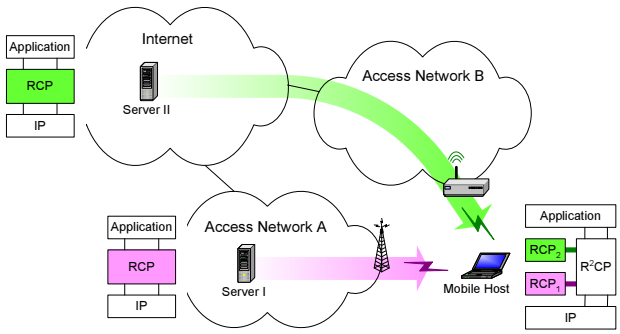


Figure 24: R^{2}CP Design

5.3.1.1 Receiver-Centric Operation

To achieve optimal performance, a mobile host may need to use network (or interface) specific congestion control. When the mobile host is equipped with heterogeneous wireless interfaces, a receiver-centric protocol allows it to freely use the desired congestion control mechanism depending on the interface it chooses, or the access network it migrates to, without involving the remote server. In addition, during periods of mobility, the mobile host may need to handoff from one server to another (for service continuity), or change the number of servers it connects to (for bandwidth aggregation). It is thus advantageous for the mobile host to use a receiver-centric protocol with a simple sender design, allowing the mobile host to have control over the reliable delivery of data from the sender(s). RCP, being a receiver-centric protocol that allows the mobile host to drive the protocol operation such as congestion control and reliability, hence turns out to be an ideal protocol for the target environment.

5.3.1.2 Maintaining Multiple States

Existing transport protocols suffer from performance degradation during handoffs across heterogeneous networks due to the prolonged handoff latency Mobile IP introduces. While end-to-end host mobility without relying on the support from the infrastructure has been proposed [101], it does not fully address this problem due to the single-state design in TCP that maintains only one TCB [86] per connection. When link layer handoffs invalidate the state maintained at the transport layer (e.g. due to the change in IP addresses), the transport layer protocol needs to modify its state accordingly for achieving transport layer mobility. Although [101] intelligently performs connection migration,

it introduces packet losses by "overwriting" the old state right after the new one is created. An ideal solution for achieving state migration, however, should allow the two states to co-exist in the connection for as long as it takes to handoff the states (considering packets in transit). Therefore, to support transparent host mobility without infrastructure support, a transport layer protocol should be able to handle multiple states. We hence build R^{2}CP as a multi-state extension of RCP. R^{2}CP dynamically creates and deletes RCP states according to the number of active interfaces in use. It effectively maintains multiple states at the mobile host without requiring explicit support from the remote server. *No change is necessary at the RCP sender to support the multi-state operation at the receiver.* R^{2}CP thus is different from related approaches [55, 90] that require changing both ends to support the multi-state operation. Since R^{2}CP is a receiver-only extension of RCP, it allows the mobile host to establish a multipoint-to-point connection to communicate with multiple servers, while in related work multiple states are confined to within a unicast connection.

5.3.1.3 Decoupling of Functionalities

An R^{2}CP connection with *k* active interfaces consists of *k* states at the receiver. Effectively, R^{2}CP maintains one RCP *pipe* per end-to-end path that exists between the receiver and the sender(s). R^{2}CP minimizes the overheads due to maintaining multiple states in a connection, by decoupling the transport layer functionalities associated with the per-pipe characteristics from those that pertain to the aggregate connection. For example, congestion control, being a per-pipe functionality, is handled by individual RCP pipes. On the other hand, reliability and socket buffer management pertain to the aggregate connection, and hence are handled by R^{2}CP itself. Therefore, the R^{2}CP engine controls what data to request from each sender, and individual RCP pipes control how much data it can request along its path. The overheads due to repetitive implementations of transport layer functionalities are minimized.

5.3.1.4 Effective Packet Scheduling

A key challenge in maintaining multiple states in a connection is the effective multiplexing of pipes with mismatched characteristics in terms of bandwidths, delays, and loss rates. Specifically, since R^{2}CP uses multiple RCP pipes across heterogeneous interfaces to request data from one or multiple senders, data segments with smaller sequence numbers traversing the slower pipes may arrive later

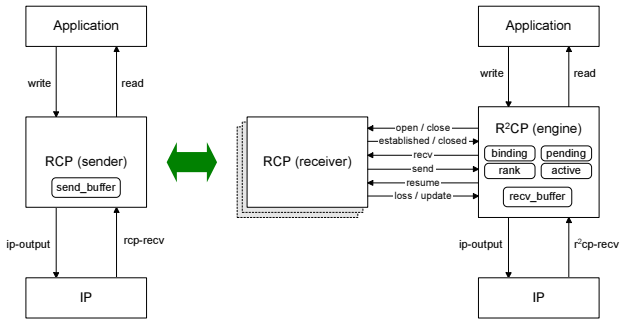


Figure 25: R^{2}CP Architecture

than those with larger sequence numbers traversing the faster pipes. Out-of-order arrivals at the receive buffer thus may cause head-of-line blocking and make the aggregate connection stall. R^{2}CP achieves effective multiplexing and bandwidth aggregation by scheduling transmissions (requests) based on the congestion window and the round-trip time of each RCP pipe. Briefly, R^{2}CP assigns the sequence of requests to each RCP pipe based on the (estimated) time the requested segment will arrive through the concerned pipe. Moreover, a request is assigned to an RCP pipe only when there is space in its congestion window. Any loss detected by individual RCP pipes is reported to R^{2}CP such that the corresponding request is reassigned to another pipe that has space in its window, to prevent the aggregate connection from stalling. Hence, head-of-line blocking due to segment losses, and bandwidth or delay mismatches of individual pipes is minimized.

5.3.2 Protocol Overview

Figure 25 presents an architectural overview of R^{2}CP and its key data structures. An R^{2}CP connection consists of one receiver, and one or multiple senders. Different senders of an R^{2}CP connection can be located at one or multiple hosts. While a unicast R^{2}CP connection is in fact equivalent to an RCP connection, a multipoint-to-point R^{2}CP connection can be considered as an aggregation of multiple RCP connections whose receiving ends are coordinated by an R^{2}CP engine at the receiver using the interface functions shown in the figure. We refer to the *virtual* connections that exist between the R^{2}CP receiver and individual senders as RCP pipes, and focus on the receiver for the following discussions.

When the application at the mobile host opens an R^{2}CP connection, initially one RCP pipe is created between the active interface and the remote server. When the mobile host handoffs from one

interface to another, a new RCP pipe between the newly active interface and the server is created, after which the old RCP pipe is deleted. However, if bandwidth aggregation is possible (the old interface remains active after handoffs) and desirable (instructed by the application through a socket option), the old pipe is not deleted. If server migration is required when the mobile host handoffs to the new interface, the new RCP pipe is created between the newly active interface and the new server. The application can use a socket option to convey the address of the new server to R^{2}CP.

Whenever multiple RCP pipes co-exist in an R^{2}CP connection, the R^{2}CP engine performs transmission scheduling using the data structures shown in Figure 25, to minimize out-of-order arrivals due to data requested through different RCP pipes. Since multiple RCP pipes collaboratively request data for the same connection, it is possible that data requested through individual pipes is non-contiguous, depending on the transmission schedule used by the R^{2}CP engine. Hence, in R^{2}CP the request is always transmitted in the pull mode (refer to Section 5.2.3.1), such that the sender can transmit *only* the data requested. However, to facilitate loss detection and loss recovery, at the receiver each RCP pipe internally maintains a local sequence number space. Since the R^{2}CP engine controls the packet I/O (to and from the IP layer), it converts the local sequence number used by each RCP pipe to the global sequence number used by the aggregate connection before sending out the packet, and vice versa. We discuss in Section 5.3.3.1 how the conversion is achieved.

5.3.3 Protocol Operations

In this section, we describe the key protocol functionalities in R^{2}CP including scheduling, connection management, congestion control, flow control, and reliability.

5.3.3.1 Scheduling

A key functionality in R^{2}CP is to perform packet scheduling across multiple RCP pipes. In R^{2}CP, the self-clocking in individual RCP pipes drives the transmissions of requests (for pulling data from the senders). Upon receiving a transmission request from any RCP pipe, R^{2}CP determines which data to retrieve and assigns (binds) the corresponding sequence number to the request. As we show in Figure 26, a FCFS (first-come-first-served) style of packet scheduling that assigns the next unbound data to a new request will result in undesirable out-of-order data arrivals. To avoid this problem, the assignment should be made based on the time the corresponding data will arrive, not

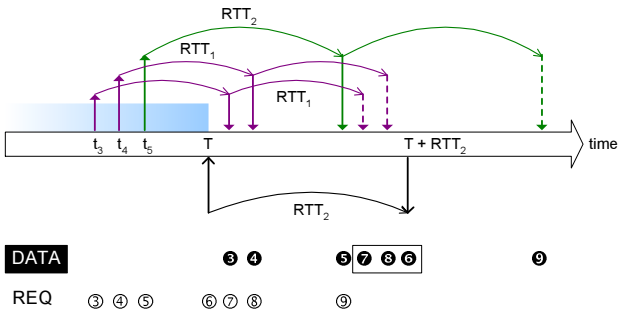


Figure 26: Motivation for R^{2}CP Scheduling

the time the request is sent. For example, in Figure 26 the request issued at $t = T$ should be assigned the *third* unbound data (which is 8), instead of the next unbound data (which is 6). We refer to the rank of the request as third.

R^{2}CP maintains the following four key data structures for performing effective packet scheduling:

- `binding`: For each request sent out by one of the RCP pipes, R^{2}CP maintains the mapping between the local sequence number of the concerned RCP pipe, and the global sequence number of the aggregate connection in the `binding` data structure. The pipe through which the data segment is requested is also recorded in the `binding` data structure.

- `pending`: The ranges of sequence numbers for data yet to be requested are maintained in the `pending` data structure. It consists of the sequence numbers of data segments that need to be retransmitted (requested again), and sequence numbers greater than the highest sequence number requested so far.

- `rank`: For every outstanding request for segment i (one with starting sequence number i) sent by pipe j, an element is inserted into the `rank` data structure with a timestamp of $T^i + 2 * RTT_j$, where T^i is the time at which the request was transmitted, and RTT_j is the round-trip time of pipe j. The timestamp is reflective of the time the data segment requested in response to the arrival of segment i, is expected to arrive (refer to Figure 26).

- `active`: When an RCP pipe issues a request to R^{2}CP for transmission, R^{2}CP can return with `FREEZE` to the corresponding RCP pipe due to unavailable space in the receive buffer. In such

an event, R^{2}CP adds the concerned RCP pipe to the `active` data structure. When any space is created in the receive buffer, R^{2}CP issues a *resume*() call to each of the pipes in the `active` data structure.

We now explain how R^{2}CP uses these data structures to perform transmission scheduling and interacts with individual RCP pipes. When pipe j uses the *send*() call with RCP sequence number s for transmission request at time T, R^{2}CP locates the rank k of the request by comparing $T + RTT_j$ with existing entries in the `rank` data structure. Then it finds segment i as the k^{th} segment to request in the `pending` data structure, updates the entry for segment i in the `binding` data structure with (j, s), and inserts an entry (i, $T + 2 * RTT_j$) in the `rank` data structure. Finally, it uses the sequence number i in the request header, and sends out the request. When a data segment m arrives, R^{2}CP deletes the corresponding entry in the `rank` data structure, enqueues the data in the receive buffer, finds the corresponding RCP pipe and its local sequence number q based on the `binding` data structure, and passes q to the corresponding RCP pipe using the *recv*() call. The concerned RCP pipe then updates its states (e.g. congestion control parameters and the next sequence number to send), and determines whether it can send more requests or not. In case it can generate more requests, it uses the *send*() interface with the next RCP sequence number for transmission request as before.

Upon receiving the transmission request from any RCP pipe, if there is no available space for more data in the receive buffer, R^{2}CP returns with `FREEZE` to freeze the concerned RCP pipe, and puts it in the `active` data structure. If later any buffer space opens up due to, say, the arrival of the head-of-line segment, R^{2}CP uses the *resume*() call to de-freeze all pipes in the `active` data structure. Whenever an RCP pipe detects a loss, it uses the *loss*() call to inform R^{2}CP. R^{2}CP then unbinds the lost segment in the `binding` data structure, inserts the sequence number in the `pending` data structure, and deletes the corresponding entry from the `rank` data structure. Whenever an RCP pipe updates its RTT estimate, it uses the *update*() call to inform R^{2}CP, which then updates the `rank` data structure for pending requests pertaining to the concerned pipe.

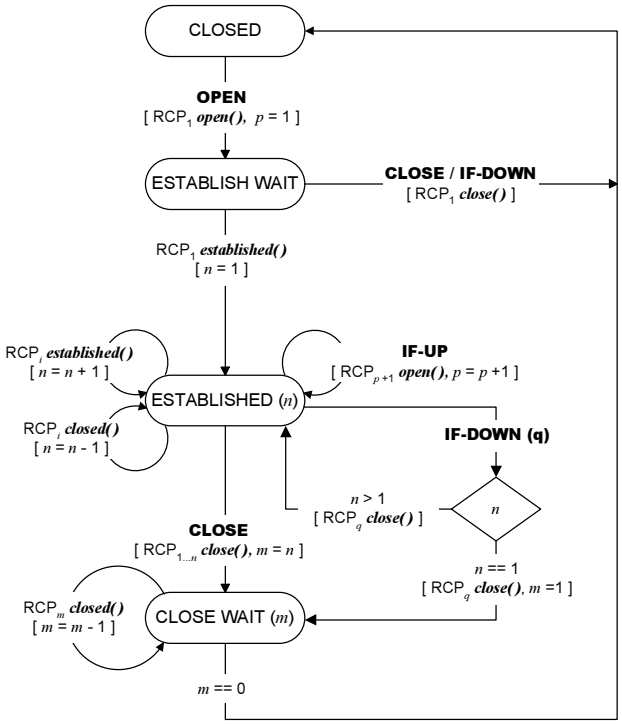


Figure 27: R^{2}CP State Diagram

5.3.3.2 Connection Management

Figure 27 illustrates the state diagram for the connection establishment and teardown phases of R^{2}CP. When the application opens an R^{2}CP socket, one RCP pipe is created. R^{2}CP creates an RCP pipe by using the *open*() call to make the RCP pipe start the connection setup procedure. The connection setup procedure for each RCP pipe is discussed in Section 5.2.3.2. When the RCP pipe is established, it uses the *established*() call to notify R^{2}CP, and then the R^{2}CP connection is established. During the lifetime of the connection, R^{2}CP may create more RCP pipes based on the callbacks from the lower layers, such that one RCP pipe is maintained for each path between an active interface and a remote sender. R^{2}CP enters the **ESTABLISHED (n)** state when the total number of RCP pipes opened is *n*. On the other hand, when an interface becomes inactive, or a sender is disconnected from the connection, R^{2}CP deletes the corresponding RCP pipe. R^{2}CP deletes an RCP pipe by using the *close*() call to make the RCP pipe enter the closing handshake which we discussed in Section 5.2.3.2. R^{2}CP appropriately moves down the state diagram when an RCP pipe returns with the *closed()* call. Finally, when the application closes the R^{2}CP socket, R^{2}CP

sends a *close()* message to each of the component RCP pipes. When all RCP pipes return with the *closed*() call, the R^{2}CP connection is closed.

5.3.3.3 Congestion Control

Congestion control in an R^{2}CP connection is performed on a per-pipe basis, where each RCP pipe is responsible for controlling the amount of data transferred through the respective path. R^{2}CP decides the congestion control mechanism to use for each wireless interface by opening an appropriate RCP pipe (e.g. RCP-NewReno, or RCP-STP as we discussed in Section 5.2.4.3). We assume the choice as to which congestion control scheme to use for each interface is an external decision, and is provided to R^{2}CP through a system configuration or a socket option.

5.3.3.4 Flow Control

Since R^{2}CP has control over the receive buffer, it is responsible for the flow control of the aggregate connection. R^{2}CP freezes a requesting RCP pipe if it finds that the number of outstanding data is equal to the available buffer space. It de-freezes concerned pipes through the *resume*() call when any space is created in the buffer. The flow control mechanism for individual RCP pipes that we discuss in Section 5.2.3.4 will not kick in since they do not deal with the actual data segments. Note that flow control not only allows R^{2}CP to avoid buffer overflow, but also allows it to achieve flexible bandwidth aggregation. Based on the "policy" supplied by the user or the application, R^{2}CP can freeze an RCP pipe even when there is space in the receive buffer, to explicitly control the amount of data transmitted through each pipe, thus achieving the policy-based bandwidth aggregation as we mentioned in Section 5.1.2.3.

5.3.3.5 Reliability

R^{2}CP is primarily responsible for the reliable data transfer of the aggregate connection. It achieves this goal by maintaining the binding information for all data segments. Once a segment is bound to a particular RCP pipe, the concerned pipe will take over the responsibility (since RCP is a reliable protocol). However, note that when an RCP pipe detects a segment loss and reports to R^{2}CP using the *loss*() call, R^{2}CP will unbind the corresponding data segment, and delegate the reliable transfer of the lost segment to the next available pipe (according to the rank). While the original RCP pipe

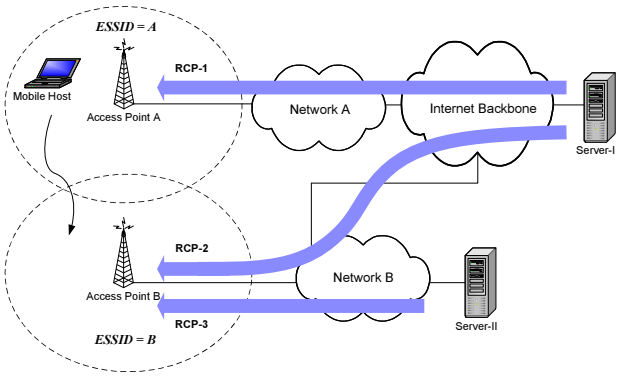
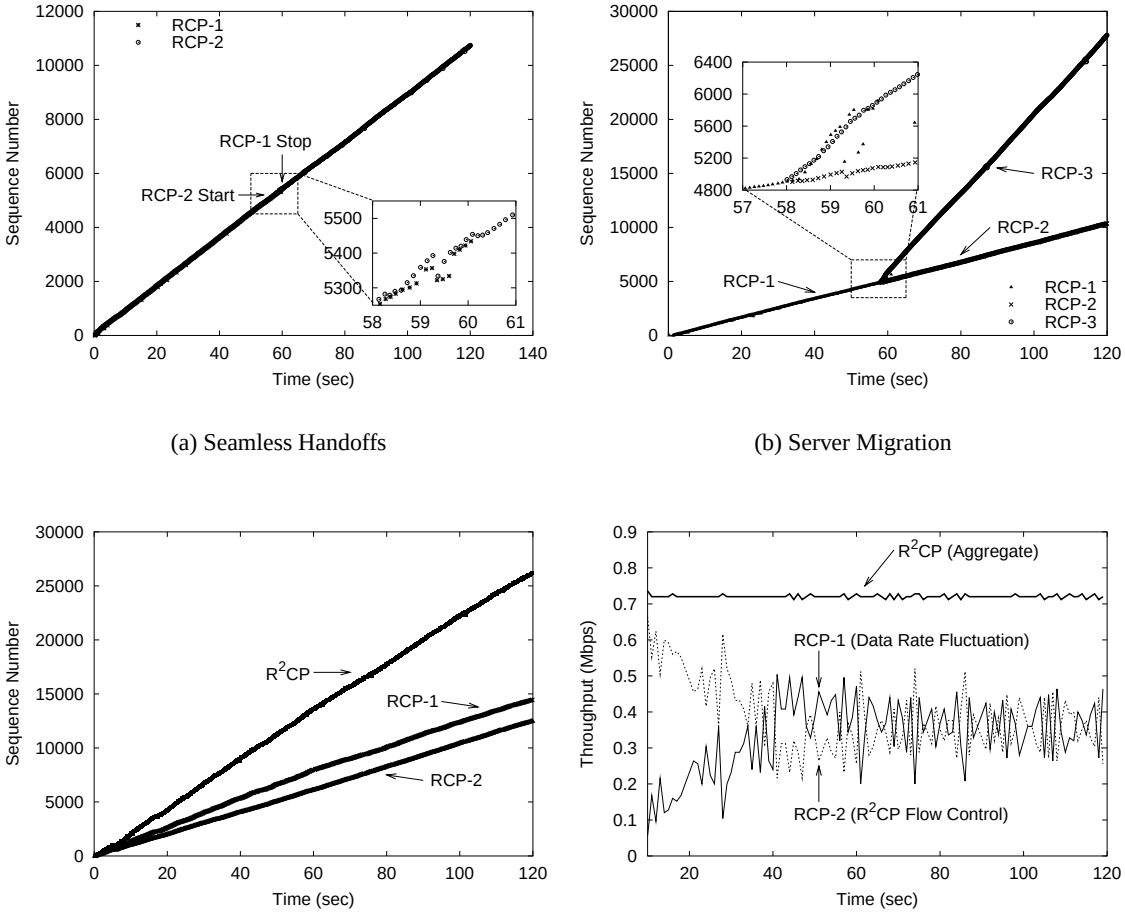


Figure 28: Testbed Scenario for Evaluating R^{2}CP

will still strive to deliver the same segment (in terms of the RCP sequence number) via retransmissions, it will be assigned a different data segment by R^{2}CP. Note that R^{2}CP is also responsible for appropriately informing the senders about what data to purge using the `SEG.DEQ` field in the RCP header that we discussed in Section 5.2.3.5.

5.3.4 Functionality Gains

In this section, we show the functionality gains when using R^{2}CP at a mobile host with heterogeneous wireless interfaces. We use both network simulation and testbed emulation to present the results. While *ns-2* has been popularly used for network simulation, it can also be used as an emulator to interact with a live network. The protocol object developed in *ns-2* can tap into the device driver of the interface card (of the host where *ns-2* is running) to inject real packets to the network. Packets received by the interface card can also be dispatched to the target protocol object in *ns-2*. The advantage of using emulation is that packets generated by the emulator experience the same bandwidth fluctuations, round-trip time variations, and losses as any other live traffic in a real network. This is especially useful for evaluating the performance of the protocol in an uncontrolled wireless environment. We use the testbed shown in Figure 28 for performing emulation. The mobile host is an IBM Thinkpad T-20 laptop, and the servers are Dell Optiplex GX110 desktops. The mobile host is equipped with two IEEE 802.11b interfaces that allow it to connect to two WLANs belonging to different administrative domains (the two cards are associated with different ESSIDs, and assigned different IP addresses). Server-I and Server-II are replicated file servers. We also use simulation with controlled parameters (e.g. bandwidth and round-trip time) to show the



(a) Seamless Handoffs

(b) Server Migration

(c) Bandwidth Aggregation (Best-Effort)

(d) Bandwidth Aggregation (Policy-Based)

Figure 29: R^{2}CP Functionality Gains

performance of R^{2}CP in various environments.

5.3.4.1 Seamless Handoffs

When mobile hosts handoff between heterogeneous wireless networks, a key challenge in supporting seamless handoffs is the problem associated with address change and prolonged registration delay. Conventional approaches for performing vertical handoffs suffer from connection disruptions due to this problem. As we explained in Section 5.3.1, the multi-state design in R^{2}CP allows it to open multiple connections (pipes) associated with the wireless interfaces that become active during handoffs. By retaining the old connection (for as long as the link layer supports) during the initial setup delay of the new connection, the application can continue transmitting and receiving

data from either or both interfaces without being disrupted during handoffs.

We show in Figure 29(a) the testbed results when the mobile host handoffs from one access network to another. The mobile host is initially connected to Server-I through network A, and hence one RCP pipe (RCP-1) is created in the R^{2}CP connection. At $t = 58s$, the mobile host decides to handoff to network B, so a second RCP pipe (RCP-2) is created (using the new network address). However, as the figure shows, RCP-1 is not closed until $t = 60s$ (a preset value), and hence during $t = 58s$ and $t = 60s$ two pipes co-exist in the connection to collaboratively deliver data for the application. Even if there is some setup or ramp-up (e.g. due to slow start) delay for the RCP-2 pipe, the existence of the RCP-1 pipe allows the aggregate connection to continue progressing without being disrupted. This is very different from related work that uses a single-state transport protocol for handoffs. Since R^{2}CP is a multi-state transport protocol, it is capable of maintaining multiple (interface specific) pipes effectively in a connection without suffering from problems due to packet reordering or duplicates. Note that the redundant striping technique proposed in [55] can also be used during handoffs for achieving better performance.

5.3.4.2 Server Migration

A key difference between R^{2}CP and other multi-state transport protocols is the ability to support end-point handoffs in R^{2}CP. By virtue of its receiver-centric design, the sender does not maintain any "hard" state (e.g. retransmission timers) of the connection. Since the mobile host controls which data to receive from the sender, handoffs from one server to another can be as simple as stop requesting data from the old server, and start from the new one. As we described in Section 5.1.2.2, server migration involves interaction between the transport layer and higher layer protocols. We focus in this section on the ability of R^{2}CP to facilitate server migration given sufficient support from the higher layers, and hence motivate its use as a valuable and effective building block for end-to-end mobility support frameworks.

As Figure 28 shows, when the mobile host moves to network B, it has access to a replicated server (Server-II). The end-to-end path from the mobile host (using interface B) to Server-II has a shorter round-trip time and a larger bandwidth, and hence the mobile host decides to perform server migration from Server-I to Server-II. Initially, the R^{2}CP connection creates an RCP pipe (RCP-1)

using network address A and the address of Server-I. When the mobile host moves to network B, R^{2}CP creates a new RCP pipe (RCP-3) using network address B and the address of Server-II. Note that in Figure 29(b) we also show a contrasting scenario where the mobile host does not perform server migration, and hence the second RCP pipe created (RCP-2) is between network address B and the address of Server-I. After the new RCP pipe is established, the mobile host requests data that has not been delivered by Server-I, instead of requesting from the first byte of the data.[2] The difference between the slopes of RCP-2 and RCP-3 indicates that RCP-3 provides a larger bandwidth than RCP-2. Approaches used for achieving seamless handoffs discussed in Section 5.3.4.1 can also be used for achieving seamless server migration.

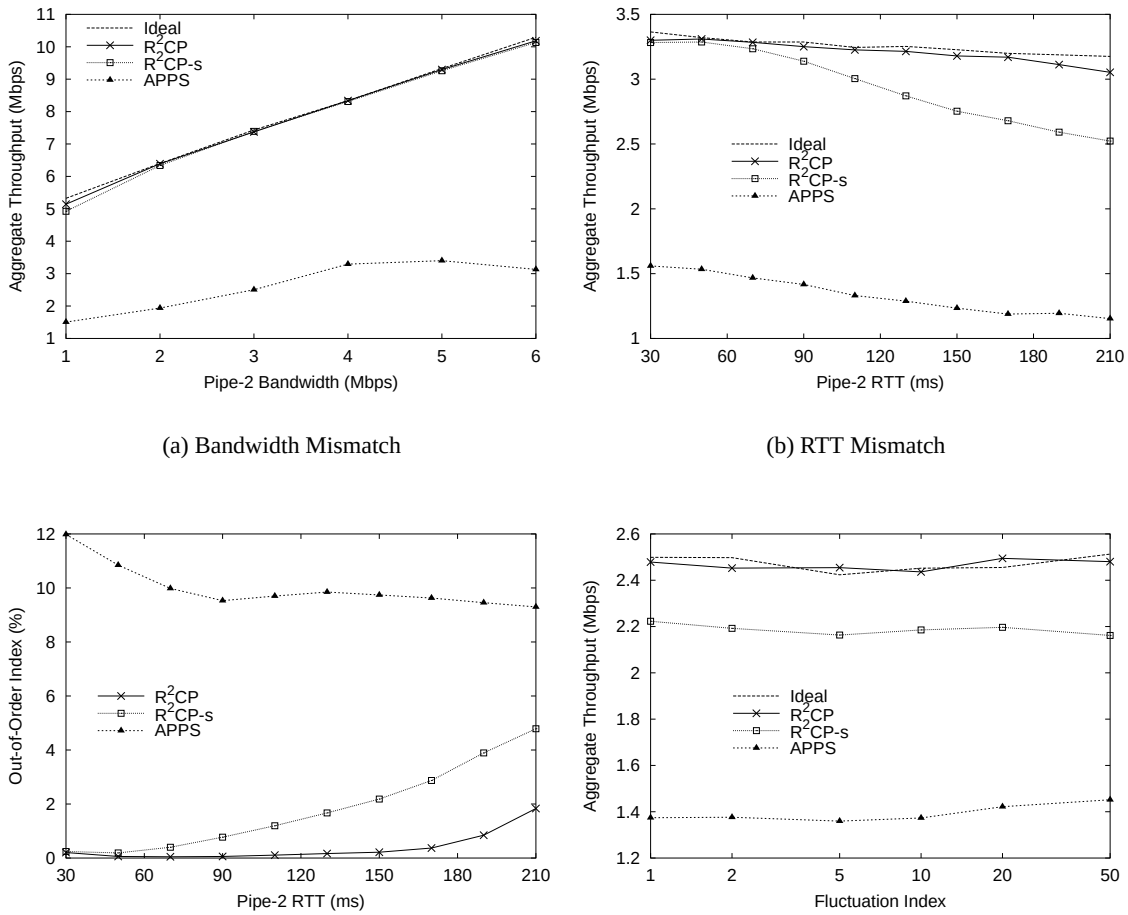
Based on the content of its receive buffer, R^{2}CP may request non-contiguous data from Server-II. Hence server migration using R^{2}CP does not not cause redundant transmissions compared to that using only TCP (the TCP sender delivers only in-sequence data stream). While support for selective pulling of data is provided by some applications (e.g. HTTP 1.1 Range Requests), it can be achieved in R^{2}CP with *no support from the server side application.*

5.3.4.3 Bandwidth Aggregation

When a mobile host handoffs between heterogeneous wireless networks, it is possible that the old connection remains active after the handoff is complete. In such a case, it would be advantageous for the mobile host to achieve aggregate bandwidths by simultaneously using both interfaces. Since R^{2}CP allows multiple RCP pipes to co-exist in one connection, and performs effective transmission scheduling for striping across multiple pipes, a mobile host using R^{2}CP can easily achieve bandwidth aggregation if desired.

We first consider the testbed scenario shown in Figure 28. While bandwidth aggregation can be achieved between the mobile host and one server (point-to-point), we consider a scenario where the two pipes connect to different servers (multipoint-to-point). The mobile host opens the RCP-1/RCP-2 pipe between network address A/B and the address of Server-I/Server-II respectively. However, instead of closing the RCP-1 pipe after RCP-2 is established, the mobile host keeps both pipes open

[2]Note that the RCP sender on Server-II may need to purge from its send buffer the data that is not required by the receiver, with or without the application's interaction.



(a) Bandwidth Mismatch

(b) RTT Mismatch

(c) Out-of-Order Index

(d) Bandwidth Fluctuations

Figure 30: Performance of R^{2}CP Scheduling

during the period it is within the coverage of both WLANs. As shown in Figure 29(c), R^{2}CP can achieve the aggregate bandwidth of the two pipes.

While Figure 29(c) shows the bandwidth aggregation result when the goal is merely to achieve the sum of individual throughputs, Figure 29(d) shows a result for policy-based bandwidth aggregation. We introduce background traffic that contends with the RCP-1 pipe in the first cell (with ESSID A). As we observe from Figure 29(d), the achieved throughput in RCP-1 suffers from large fluctuations (the instantaneous throughput is obtained using a 1-second bin). We consider a scenario where the goal of bandwidth aggregation is to achieve a steady throughput of 720Kbps (for, say, video streaming) by *opportunistically* using the RCP-2 pipe to compensate for the data rate fluctuations in RCP-1. R^{2}CP uses a simple token-bucket algorithm for performing the flow control

that we discussed in Section 5.3.3.4. The token is generated depending on the target data rate, and is consumed when data is bound to an RCP pipe. The policy used in this example is that the secondary pipe (RCP-2) is allowed to consume only tokens not used by the first pipe (RCP-1). R^2CP freezes an RCP pipe whenever there is no more token available. The figure shows that the aggregate throughput achieved by R^2CP using such a policy-based bandwidth aggregation is relatively constant, despite the fluctuations observed in RCP-1. The amount of data transferred using the RCP-2 pipe *mirrors* that of the RCP-1 pipe.

We now use simulation to evaluate the performance of R^2CP in achieving effective bandwidth aggregation under various network conditions. We use a network topology similar to the testbed topology shown in Figure 28. The mobile host opens two pipes to aggregate bandwidths from different servers. We vary the characteristics of the two paths, in terms of the bandwidth of the bottleneck link, and the round-trip time of the entire path, to introduce bandwidth mismatches and delay mismatches. We also introduce bandwidth fluctuations by using on/off traffic sources as we described in Section 5.2.4. We compare the performance of R^2CP against the following approaches: (i) Ideal: the ideal performance of bandwidth aggregation, where the aggregate bandwidth equals the sum of bandwidths along the two pipes; (ii) APPS: an application layer striping approach (similar to the one used in [55]), where the application stripes across multiple RCP connections without using R^2CP; and (iii) R^2CP-s: a simplified version of R^2CP, where the data request is assigned to individual pipes on a first-come-first-served basis without considering the round-trip times.

In Figure 30(a), we vary the bandwidth of the two pipes such that the bandwidth of the first pipe is fixed at 4Mbps, while that of the second pipe varies from 1Mbps to 6Mbps. We observe that both R^2CP and R^2CP-s achieve the ideal performance irrespective of the bandwidth mismatches. The application striping approach fails to achieve the desired performance for the same reason explained in [55]. In Figure 30(b), we vary the round-trip time of the two pipes such that the RTT of the first pipe is fixed at 30ms, while that of the second pipe varies from 30ms to 210ms. We find that while the performance of R^2CP still closely tracks the ideal performance, R^2CP-s fails to scale when the RTT mismatch increases beyond 3. The performance degradation of R^2CP-s is due to the scheduling used that does not take into consideration the round-trip times of different pipes. While an FCFS style of striping policy works well when the round-trip times of different paths are comparable, as

the RTT mismatches increase, it suffers from frequent out-of-order arrivals. Due to the limited space in the R^{2}CP receive buffer, head-of-line blocking eventually triggers the flow control of R^{2}CP and causes the progression of the aggregate connection to stall. We show in Figure 30(c) the percentage of packets that find the buffer 75% full upon arrivals, for three different striping approaches. The reason for the non-performance of the application striping approach is clear from the figure. While R^{2}CP-s manages to maintain a small queue size when the RTT mismatches are small, the queue builds up noticeably as the RTT mismatches increase. R^{2}CP, on the other hand, achieves better performance even with large RTT mismatches. Finally, Figure 30(d) shows the performance of R^{2}CP when there are significant data rate fluctuations in individual pipes. We introduce 10 Pareto on/off flows in each pipe as the background traffic. The burst time of the Pareto traffic is set to $0.1 * t$ seconds, and the idle time is set to $0.2 * t$ seconds. The rest of the parameters used are the same as those used in Section 5.2.4.1. We vary t from 1 to 50 and refer to it as the fluctuation index. Note that packets sent through each pipe will experience large delay variations (jitters) due to the on/off traffic. It is clear that despite such fluctuations R^{2}CP is still able to closely track the ideal performance.

PART II

Bandwidth Scarcity

CHAPTER 6

NETWORK SCALABILITY WITH USER POPULATION

Wireless links provide bandwidths that are significantly lower than those provided by their wired counterparts. The scarcity of wireless bandwidth has inspired a considerable amount of research toward improving the performance of wireless networks in different layers of the network protocol stack. However, the tremendous growth in the number of mobile users has significantly exposed the limitations of these approaches. In this chapter, we first describe the problem of network scalability with user population in existing wireless networks. We then discuss why this problem is not effectively addressed by related work. Finally, we present an outline of the solution proposed in this work.

6.1 The Problem

Existing wireless networks such as 3G WWANs and WiFi WLANs use a *cellular network model* where a base station (or an access point) covering a cellular service area coordinates the transmissions of all mobile hosts in the cell. Mobile hosts communicate directly with the base station and do not interact in any manner with other mobile hosts in the cell. The base station in turn is connected to the backbone Internet through a distribution network. If the source and the destination lie within the same base station's cell, the base station serves as a relay between the mobile hosts. Otherwise, the base station serves as a gateway between the wireless network and the wired backbone network.

In such a cellular network model, all mobile hosts in the cell share the bandwidth provided by the base station for network access. For a given number of users n, the bandwidth per user is limited to the order of $O(\frac{1}{n})$. While a considerable body of researches has proposed solutions in different layers of the network protocol stack for improving the performance of wireless networks, the performance attainable has been constrained by the fundamental limits posed by the underlying cellular network model. Therefore, the only avenue for improving data rate in cellular networks is to decrease the coverage area per base station, thus reducing the number of users served. Although such

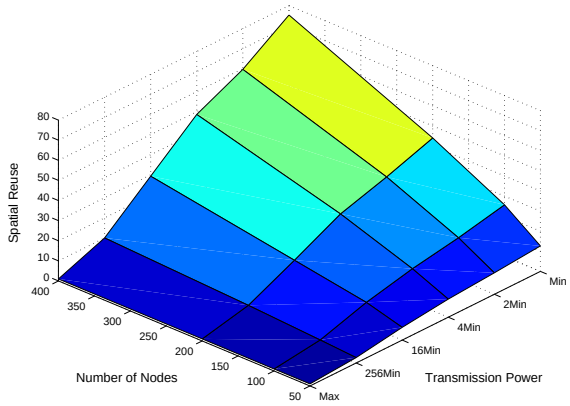


Figure 31: Spatial Reuse in Wireless Networks

an approach has been adopted in hierarchical cell structure (HCS) systems [22, 89], the drawback is the high infrastructure cost involved in deploying a large number of base stations and the associated distribution networks that eventually makes it unable to scale with the tremendous growth of mobile hosts.

The inability of the cellular network model to scale with user population stems from its poor performance in achieving the desired *spatial efficiency* (defined as bits per second per square meter) [35, 85] of the wireless bandwidth. While packing more base stations in the same area is an expensive way to increase the spatial efficiency, a more scalable approach is allowing mobile hosts to communicate only with their neighbors using minimal transmission powers (which decrease with the number of mobile hosts). If we define *spatial reuse* [64] as the number of simultaneous transmissions possible in the network, it can be seen in Figure 31 that spatial reuse increases not only with decreasing transmission ranges, but also with increasing number of nodes (mobile hosts) in the network.[1] If mobile hosts can be made to access the network (i.e. base station) through communication with their neighbors, the spatial reuse gain can be leveraged for increasing the network capacity.

In this context, the *peer-to-peer network model* used by a special class of networks called adhoc networks has gained attention. In the peer-to-peer network model, mobile hosts *cooperatively*

[1] In consideration of the various multiple access techniques available, the term *simultaneous transmissions* can be qualified with respect to a time slot, frequency band, or code sequence. Note that the *Min* transmission power labeled in the figure is the minimum power used to keep the network from partitioning, while the *Max* transmission power is the minimum power used to keep the network *fully* connected. For details on the how Figure 31 is obtained, please refer to [52].

act as routers to relay traffic for peer hosts in the network. Using transmission ranges that are just large enough to ensure network connectivity allows the peer-to-peer network model to potentially maximize the spatial reuse in the network. It has been shown in [46] that in a network with n uniformly distributed nodes and random source-destination pairs, the transport capacity of the network is of the order $O(\sqrt{n})$ using the peer-to-peer network model (while that of the cellular model is $O(1)$ irrespective of the number of nodes in the network, as shown in Figure 31 with the maximum transmission power). Another study in [44] shows that if nodes are mobile and can use as many nodes as possible in the network to relay traffic, the network capacity of the peer-to-peer network can increase as $O(n)$. In other words, the peer-to-peer network model can potentially allow the network to *scale with any number of users in the network.*

Despite its potential advantages, the peer-to-peer network model has hitherto been used only in stand-alone environments that lack the services of an established backbone infrastructure, with the goal of communicating with other mobile hosts in the network. In the context of Internet access, however, a dominant portion (if not all) of the traffic consists of accesses to servers (and hence the base station) in the backbone Internet, instead of communication between mobile hosts [21]. Moreover, while protocols designed for ad-hoc networks typically focus more on overall network performance rather than on providing service guarantee to individual mobile hosts, this is not the case for wireless access networks. For applications where connectivity and fairness are requirements, the peer-to-peer network model has not been considered as a viable option due to the performance degradation associated with host mobility and unfair resource allocation [18, 52]. In this work, we start with the peer-to-peer network model for leveraging its spatial reuse benefits that can scale with user population. We evaluate its performance benefits and drawbacks when considered as an alternate network model in conventional cellular wireless environments.

6.2 Related Work

As we mentioned earlier, existing work that focuses on solutions in different layers of the network protocol stack suffers from the performance limitations of the underlying network model [12, 62, 70, 97]. In terms of alternate network models, recently several approaches have been proposed to use the relaying capability of mobile hosts in a cellular network for load balancing [114, 116],

coverage extension [1, 3], throughput increase [48, 72], and transmission power reduction [52, 69]. In [3], the authors propose an approach called ad-hoc GSM (A-GSM) to improve the coverage of GSM networks over dead spots where direct communication with the base station is not possible. Mobile hosts in dead spots switch *dual-mode* terminals to the ad-hoc mode for relaying traffic to other hosts that have direct communication with the base station. Opportunity driven multiple access (ODMA) [1], a scheme proposed for use with the 3G systems, uses peer relays to address the problem of data rate degradation toward the edge (boundary) of the cell. Mobile hosts that suffer from low-bit-rate transmissions due to the large transmission distance involved, relay traffic to other hosts within the high-bit-rate coverage area for using link transmissions with higher data rates. In [114], the authors propose an integrated cellular and ad-hoc relay (iCAR) system to balance traffic loads between cells. Special mobile relays are placed strategically between cells to relay traffic from an overloaded cell to a relatively underloaded cell. The authors in [72] also propose a unified cellular and ad-hoc network architecture (UCAN) to provide fair service to mobile hosts without reducing the aggregate throughput in the cell. Mobile hosts that suffer from poor channel quality use peer-to-peer links to access proxy clients with better channel quality. Multi-hop cellular networks (MCN) proposed in [69] allows multi-hop communication between the mobile hosts and the base station to reduce the transmission power used. MCN is shown to improve throughput performance when sources and destinations co-exist in the same cell without mobility. Finally, a hybrid network model is proposed in [52], where mobile hosts by default use the peer-to-peer network model for communication. The base station dynamically coordinates the network topology by directing the power used at the mobile hosts, to maximize spatial reuse and reduce network partitions. The model is again evaluated only for the scenario where all sources and destinations are co-located within the same cell.

While these approaches are clear instances that show the potential performance benefits of incorporating peer-to-peer communication in cellular networks, they did not provide solutions to address the problem we consider since the cellular network model is still the dominant mode of operations. Moreover, in these approaches, the peer-to-peer network model is used “as-is” with the drawbacks such as mobility degradation still present. Mobile hosts hence can potentially suffer from performance degradation due to the use of the peer-to-peer network model in the network.

6.3 Solution Outline

Existing work that studies the performance of the peer-to-peer network model has so far focused on scenarios where mobile hosts randomly choose any other hosts in the network as destinations. In conventional cellular wireless networks, however, a dominant portion of the traffic consists of accesses to servers in the backbone network, and hence most mobile hosts use the base station as the destination. Therefore, we start in Chapter 7 by investigating the performance of the peer-to-peer network model when used in the cellular wireless environment for Internet access. We find that while the peer-to-peer network model has performance gains over the cellular network model in terms of lower power consumption, it suffers from lower throughput performance when all mobile hosts use the base station as the destination. We identify the reasons why the spatial reuse gain cannot be effectively translated into higher end-to-end throughput as: the bottleneck around the base station, inefficiencies of the peer-to-peer network protocols, and impact of host mobility. Using the peer-to-peer network model "as-is" in cellular wireless networks thus can result in degraded throughput despite the power savings.

Based on the insights gained through the performance analysis, we proceed to propose two fundamental principles that in tandem allow the peer-to-peer network model to leverage its spatial reuse gain, and achieve performance improvement over the cellular network model. In Chapter 8 we focus on the first principle called *base station assistance* that leverages assistance from the base station for reducing the protocol inefficiencies and avoiding the performance degradation due to the artifacts in the peer-to-peer network model such as vulnerability to mobility. We propose two instantiations of the principle called *assisted scheduling* and *dual-mode operation*, and show through network simulation their performance benefits. In Chapter 9 we focus on the second principle called *multi-homed peer relay* that leverages the relaying capability of multi-homed peer hosts for breaking the bottleneck at the base station. We propose two instantiations of the principle for relaying traffic between the *wireless and wired domains*, and between *heterogeneous wireless domains* respectively. We use network simulation to substantiate their effectiveness in alleviating the base station bottleneck. In the end, we conclude that using the peer-to-peer network model in cellular wireless networks is a promising approach when the network model is complemented with appropriate mechanisms.

CHAPTER 7

PEER-TO-PEER NETWORK MODEL

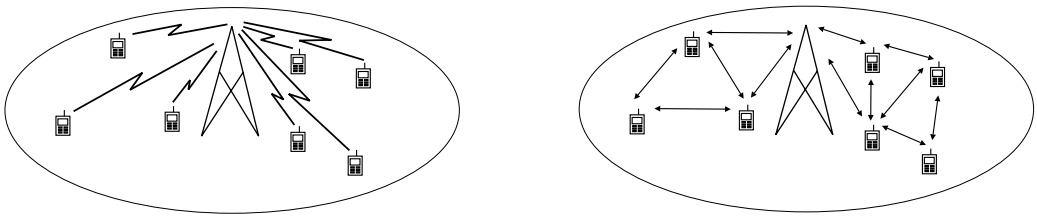
The peer-to-peer network model offers spatial reuse gains that can scale with the number of nodes in the network. Related work has shown through theoretic analysis the scalability of its transport capacity (end-to-end throughput) with the network size when nodes randomly choose destinations in the network [44, 46]. In this chapter, we study the performance of the peer-to-peer network model in an Internet access scenario where a dominant portion of the nodes chooses the base station as the destination. We first describe the simulation model used for the performance evaluation, and then present results motivating the benefits of the peer-to-peer network model over the conventional cellular network model. Finally, we investigate the performance of the peer-to-peer network model when used in the Internet access scenario.

7.1 Evaluation Model

We consider a packet-switched cellular wireless data network that provides services to mobile hosts through the base station in the cell. Existing cellular wireless networks use the cellular network model shown in Figure 32(a) as the mode of communication. However, we consider the use of the peer-to-peer network model in the cellular wireless environments[1] as shown in Figure 32(b). Section 7.2 considers a scenario where the source-destination pairs are randomly chosen in the cell, while Section 7.3 focuses on a scenario where all mobile hosts use the base station for Internet access. We use the *ns-2* network simulator with multi-hop wireless extensions [105] for evaluating the performance of the cellular and peer-to-peer network models. In the following, we describe the evaluation model used and assumptions made in this work:

- *Topology and Mobility Model:* We primarily use a network of 100 mobile hosts (nodes) for performance evaluation. However, to show the impact of network size, we also consider

[1] We restrict the use of the peer-to-peer network model to mobile hosts in the same cell, and hence we focus on a single cell in this work.



(a) Cellular Network Model (b) Peer-to-Peer Network Model

Figure 32: Wireless Network Models

different network sizes including 50, 200, and 400 nodes. For most scenarios nodes are randomly distributed in a 1500m $\times$ 1500m grid with the base station at the center, although we also consider non-uniform (skewed) distributions with introduced hot spots in Section 7.2.2. We use the waypoint mobility model with parameters *speed* and *pause* [18] when node mobility is considered in the scenario. Initially, a mobile host randomly picks a destination within the grid, and moves toward the destination with a speed randomly chosen from the range (0, *speed*]. Once the mobile host reaches the destination, it remains static at that position for *pause* amount of time, after which the whole cycle repeats again. We vary *speed* from 1m/s (pedestrian speed) to 20m/s (vehicular speed), and set *pause* to 0 seconds for creating the most dynamic scenario.

- *Network Model:* We characterize the cellular network model as one-hop communication coordinated by the base station. Mobile hosts use transmission powers required to communicate directly with the base station, which then uses a round-robin packet scheduling algorithm to arbitrate channel access among mobile hosts [52]. The IEEE 802.11 protocol in the PCF (Point Coordination Function) mode [56] is used for channel access in the cellular network model. On the other hand, in the peer-to-peer network model, mobile hosts reach the base station through multi-hop routes consisting of other peer hosts in the network. The transmission range of each mobile host is set to 250m by default, and the IEEE 802.11 protocol in the DCF (Distributed Coordination Function) mode (with RTS-CTS handshake) is used as the MAC protocol. We primarily use DSR (Dynamic Source Routing) [62] as the routing protocol, but

we also use DSDV (Destination-Sequenced Distance Vector) [83] to investigate the impact of different routing protocols. We choose DSDV since it is a proactive, table-driven hop-by-hop routing protocol, whereas DSR is an on-demand source routing protocol.

- *Traffic Model:* We use TCP as the transport layer protocol because of the target environment that we consider in this work. Specifically, more than 90% of the traffic in the Internet consists of TCP flows [36, 106], and it is reasonable to assume that traffic due to mobile Internet users will not deviate from this behavior. While the focus of this work is to consider a scenario where all mobile hosts use the base station for Internet access, we also consider scenarios with different traffic localities in Section 7.2.3. The base station is the end point for all *non-local* flows with destinations outside the cell or in the backbone network. We do not explicitly model the path between the base station and the backbone server, since we assume that the bottleneck of the end-to-end connection between the mobile host and the backbone server is in the wireless domain.[2] To study the impact of traffic load on the performance of the two network models, we use a CBR application (atop TCP) that limits the maximum data rate generated by each mobile host. We vary the per-flow data rate from 8Kbps (lightly loaded) to 32Kbps (heavily loaded) to emulate the use of different applications with different data rate requirements. In addition to the rate-controlled sources, we also show results for backlogged sources by using the FTP application, where the sending rate is purely determined by the congestion control mechanism in TCP. Finally, we use the Pareto on/off traffic source to emulate the bursty and heavy-tailed traffic observed in the web-browsing application [27]. The shape parameter of the Pareto traffic source is set to 1.5, the mean burst time is set to 1s, the mean idle time is set to 2s, and the data rate during the burst time is varied from 12Kbps to 48Kbps.

- *Channel Model:* We use a single channel with a data rate of 2Mbps. The channel model used is a combination of the free space propagation model and the two-ray ground reflection model, where the signal degrades as $1/r^2$ within the cross-over distance (about 90m) and as

[2] For simplicity we assume that the mobile host acts as the traffic source. Since we do not consider asymmetric channels in this work, the direction of the traffic does not impact the simulation results shown.

$1/r^4$ beyond. We do not explicitly model random channel fading and shadowing, but assume the use of intelligent modulation and coding schemes for addressing wireless errors. The signal thresholds for successful packet reception and packet capture are modeled using the default *ns-2* parameters.

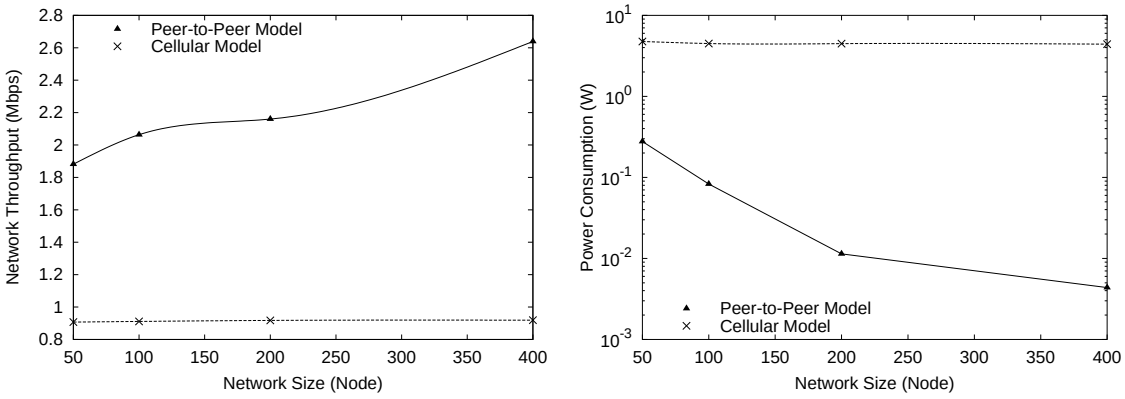
- *Power Model:* We assume all nodes in the network use the same transmission power for communication in both models. The transmission power used in the cellular network model is set to the value such that all mobile hosts can communicate directly with the base station. For the peer-to-peer network model, by default the transmission range is set to 250m. However, when considering the capacity of the network, mobile hosts use the minimal transmission power calculated based on the network topology to keep the network connected [52]. The total power consumption incurred at each mobile host includes both the transmission power and the reception power. Packet reception accounts for a constant power consumption in both models and is independent of the transmission power. The parameter is set using the default *ns-2* value. For fair comparison, all packets including the routing packets used in the peer-to-peer network model are counted.

7.2 Motivation

In this section we present simulation results comparing the performance of the cellular and peer-to-peer network models in terms of network throughput and per-node power consumption. We compare the performance of the two models along a variety of parameters including network size, node distribution, and traffic locality. The objective is to motivate the performance benefits of the peer-to-peer network model, and set the baseline for results presented in Section 7.3.

7.2.1 Network Size

To investigate the performance of the two network models with respect to the network size, we vary the number of nodes in the network from 50 to 400 and randomly distribute the nodes over the grid. We consider a scenario where the source and destination are randomly chosen, and the number of flows is equal to the number of nodes. We measure the end-to-end throughput achieved at each destination (TCP sink), and calculate the network throughput as the sum of all per-flow throughput.



(a) Network Throughput

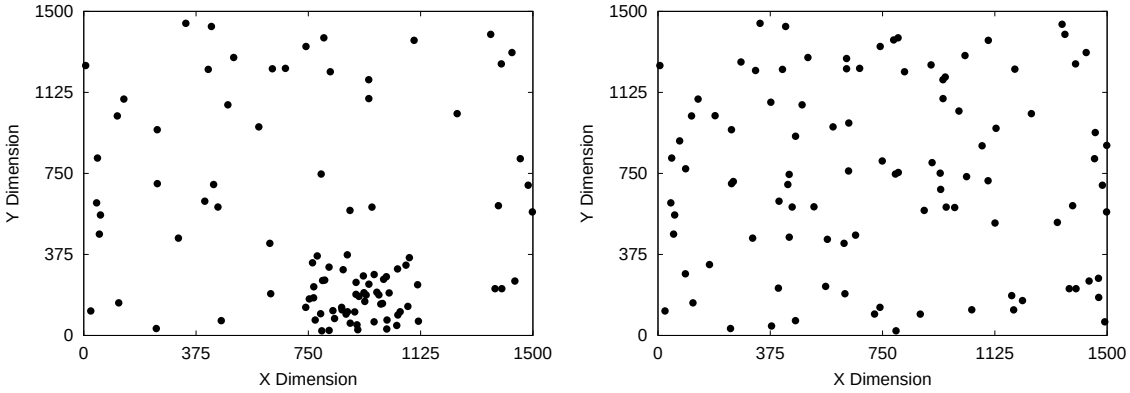
(b) Power Consumption

Figure 33: Impact of Network Size

Network throughput can be considered as a measure of network capacity from the perspective of end-to-end flows. We calculate the (per-node) power consumption using the model described in Section 7.1. We do not consider the power consumption at the base station in both models.

Figure 33(a) shows the network throughput of the two models with respect to the size of the network. As expected, the capacity of the cellular network model does not increase with the network size. It stays at around 1Mbps since an end-to-end flow requires two link transmissions to and from the base station (note the channel data rate is 2Mbps). It is clear that the peer-to-peer network model is able to achieve higher network throughput than the cellular network model. Moreover, the capacity increases with the network size. An interesting result to note is that the increase in network throughput is slower than the increase in spatial reuse (as we showed in Figure 31) due to the multi-hop routes incurred [46, 52]. Nevertheless, the peer-to-peer network model can indeed leverage the spatial reuse gain and achieves higher network throughput as the number of nodes increases in the network.

Figure 33(b) compares the per-node power consumption for the two models. In the cellular model, there is no significant change in the per-node power consumption as the network size increases since we assume all nodes to use the same transmission power for communication as mentioned in Section 7.1. In the peer-to-peer model, however, the average power consumption decreases significantly as the network size increases. The reason is that the minimal transmission power for



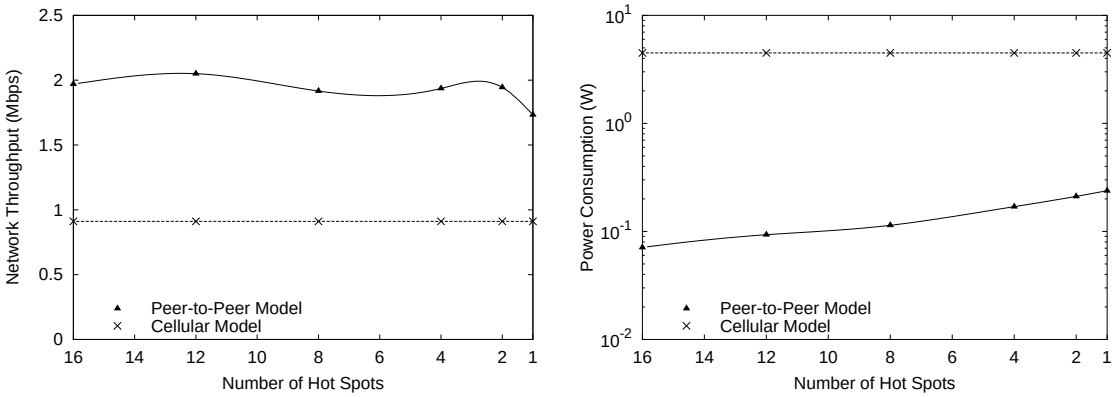
(a) $h = 1$ (b) $h = 16$

Figure 34: Skewed Node Distribution

network connectivity decreases as the network density increases. This is notwithstanding the fact that the average hop count for an end-to-end flow increases with decreasing transmission range, resulting in an increased number of link transmissions per packet. Recall that in the channel and power models we employ in the simulations, a packet transmission of distance r accounts for an additional usage of $O(r^4)$ of the battery power (for the 100-node topology the minimum transmission power used to keep the network connected is approximately 200mW). Compared to a cellular network of n nodes, although flows in the peer-to-peer network model traverse more number of hops of the order $O(\sqrt{n})$, the transmission range decreases as $O(\frac{1}{\sqrt{n}})$. Hence, the amount of transmission power consumed in the peer-to-peer model is less than that consumed in the cellular model by an order of $O(n^{3/2})$. The reduction of more than an order in the power consumption is in fact another key reason that drives the investigation of the peer-to-peer network model as an alternate model in future wireless networks [37].

7.2.2 Node Distribution

We show in Section 7.2.1 that for different network sizes with random topologies, the peer-to-peer model achieves better performance than the cellular model both in terms of network throughput and power consumption. In this section we focus on a network of 100 nodes, and study the effect of introducing "hot spots" on the performance of the two network models. To create skewed node distribution, we divide the 1500m $\times$ 1500m grid into a 4 $\times$ 4 array of same-sized smaller grids



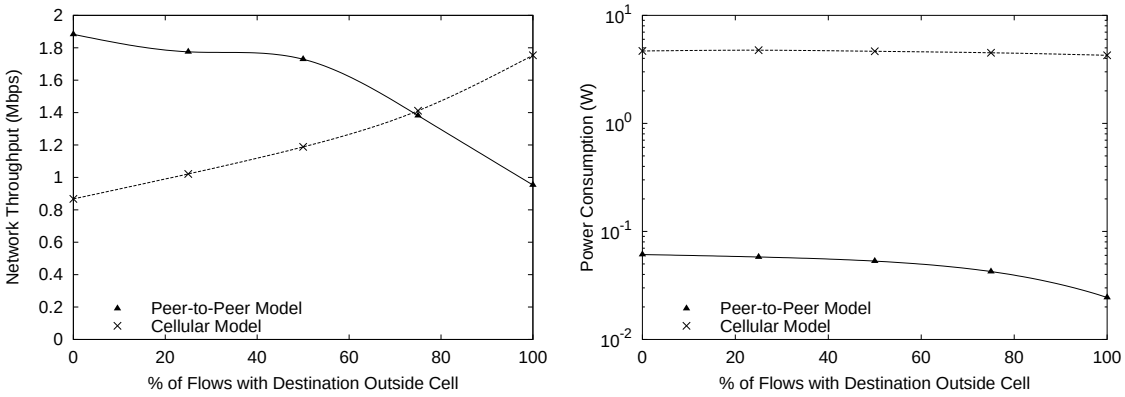
(a) Network Throughput

(b) Power Consumption

Figure 35: Impact of Node Distribution

(each with a side of 375m). We randomly distribute 50 of the 100 nodes in the 1500m $\times$ 1500m grid as before. However, for the other 50 nodes we randomly pick h of the 16 smaller grids and distribute the 50 nodes only among the h smaller grids. We vary the value of h from 1 (where all 50 nodes are within the same small grid) to 16 (where the entire distribution is the same as the default case). Figure 34 shows two sample node distributions for $h = 1$ (one hot spot) and $h = 16$ (no hot spots).

Figure 35(a) shows the impact of such non-uniform distribution on the throughput performance of the two network models. The performance of the cellular model is not affected by the node distribution because nodes act in a "peer-agnostic" fashion. For the peer-to-peer model, we observe that although the non-uniform distribution does impact the throughput (specifically for $h = 1$), the peer-to-peer model still has a higher throughput than the cellular model. This can be explained as follows: (i) Although the contention among nodes in hot spots increases, the contention among nodes outside the hot spots in fact decreases (recall that the total number of nodes is constant for different scenarios). For flows that do not traverse the hot spots, their throughput will in fact increase. (ii) The average hop count decreases with the increased non-uniformity of the distribution since nodes in the hot spot are "closer" to each other. Specifically, for flows with source and destination nodes within the hot spots, the multi-hop route may in fact degenerate to a single hop. Although the contention within hot spots increases, since the source and destination are just one-hop away, the



(a) Network Throughput

(b) Power Consumption

Figure 36: Impact of Traffic Locality

performance will not be worse than the cellular network model.

Note that since nodes outside the hot spots are more sparsely distributed, the minimal transmission power for network connectivity increases when hot spots are introduced. We observe in Figure 35(b) that the per-node power consumption in the peer-to-peer network model increases with the increasing non-uniformity of the topology. However, its performance is still better than the cellular network model even in the worst scenario.

7.2.3 Traffic Locality

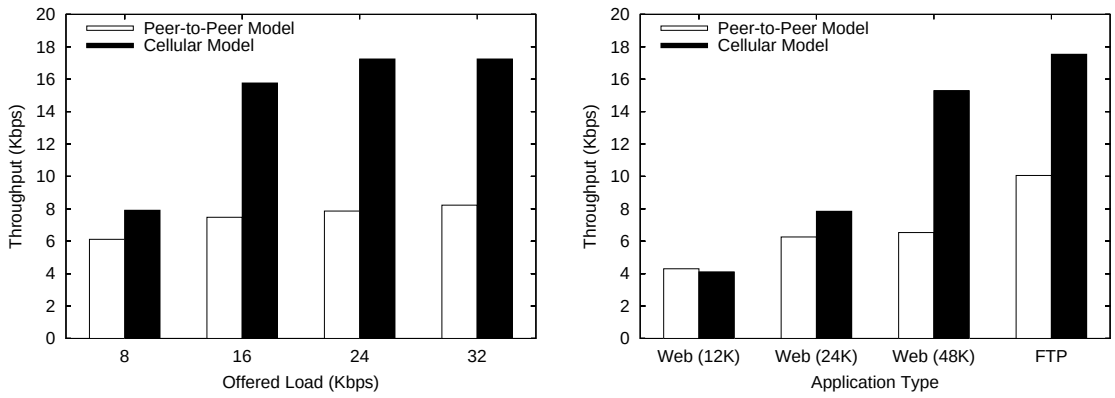
We have so far considered the scenarios with random source-destination pairs. However, as we mentioned earlier, in conventional cellular wireless environments, accesses to backbone servers (e.g. Web, Email and FTP) form a majority, and hence the base station is the destination for most flows – which we refer to as non-local flows. In this section, we study the impact of such non-local flows on the performance of the two network models. We consider a random topology with 100 nodes where the transmission range of each node using the peer-to-peer network model is set to 250m. The total number of flows is set to 100, but we vary the percentage of non-local flows from 0% (destinations are randomly chosen from mobile hosts) to 100% (all mobile hosts use the base station as the destination).

Figure 36(a) shows the throughput performance of the two network models for different percentages of non-local flows in the network. We observe that the performance of the cellular network

model increases with decreasing traffic locality. This is because non-local flows involve only one wireless transmission (source to the base station) while local flows require two wireless transmissions (from the source to the base station and from the base station to the destination). Hence in the cellular network model, the network throughput for a purely non-local scenario is about twice the value for a purely local scenario. On the other hand, for the peer-to-peer network model we observe that its performance starts decreasing as the percentage of non-local flows increases, and falls down to below the performance of the cellular network when 100% of the flows are non-local. The throughput decrease with increasing non-local traffic can intuitively be explained as a result of the increased contention in the neighborhood of the base station. However, it does not explain why the peer-to-peer network model *should* perform worse than the cellular network model – considering both have the same bottleneck (the base station). We perform a detailed performance analysis to investigate the true reasons behind the performance degradation in Section 7.3. As shown in Figure 36(b), the peer-to-peer network model under all traffic localities still achieves better performance than the cellular network model in terms of power consumption. Hence we focus on the throughput performance in the following.

7.3 Internet Access Scenario

We have observed in Section 7.2 that the peer-to-peer network model *does* have performance benefits over the cellular network model in translating the spatial reuse gain into higher network throughput. While such performance gains remain valid for various network sizes, transmission ranges, and skewed node distributions, the peer-to-peer network model suffers from significant performance degradation when the traffic locality is low. In this section, we take a closer look at the phenomenon to identify the reasons that cause the performance degradation. We consider the worst-case scenario where all mobile hosts choose the base station as the destination. In addition to the impact of traffic locality, we also identify other performance tradeoffs in the peer-to-peer network model including unfairness and impact of host mobility. To keep the focus of this section, we consider the throughput performance of the two models including: (i) average throughput achieved by all mobile hosts in the network, (ii) throughput distribution of all mobile hosts, and (iii) instantaneous throughputs observed by individual mobile hosts.



(a) CBR Application

(b) Web and FTP Applications

Figure 37: Throughput Average

7.3.1 Throughput

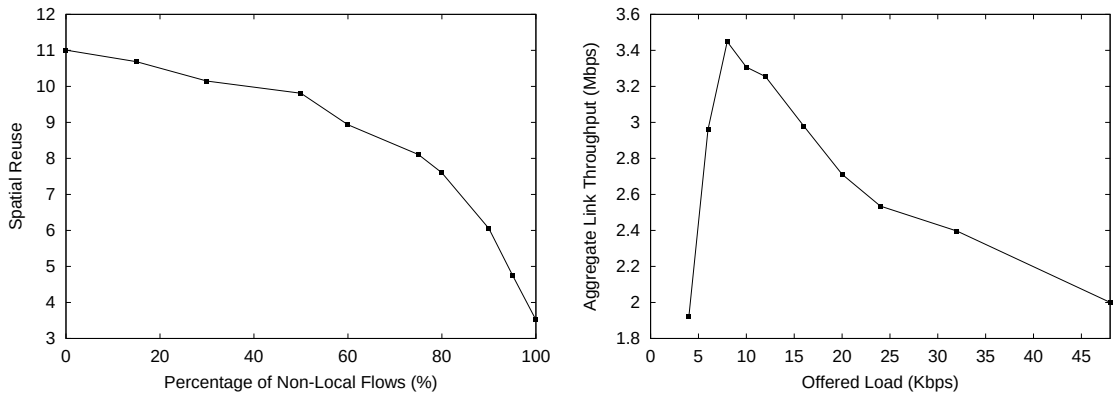
In Figure 37, we compare the performance of the two network models by showing the end-to-end throughput averaged over all 100 flows in the network. The per-flow throughput is measured at the TCP sink for a duration of 100 seconds. Figure 37(a) is obtained by using a rate-limited traffic source (CBR application) with data rates varied from 8Kbps to 32Kbps, while Figure 37(b) is obtained by using a backlogged traffic source (FTP application), and Pareto traffic source (Web application) with the burst data rate varied from 12Kbps to 48Kbps as described in Section 7.1. The results are averaged over 10 random network topologies, and DSR is used as the routing protocol for the peer-to-peer network model.

Contrary to results shown in related work [46, 52, 68] that spatial reuse can increase the end-to-end throughput, we observe from Figure 37 that not only is the spatial reuse benefit not translated into better per-flow throughput, but the performance in the peer-to-peer network model is in fact lower than that in the cellular network model – for various applications and data rates used. The discrepancy arguably results from the change in the traffic distribution: in related work the sources randomly choose any other nodes in the network as destinations, but in Figure 37 the base station is the destination for all sources. Since the channel around the base station has to be shared by all flows in the network, any increase in the spatial reuse elsewhere in the network cannot be fully realized due to the base station bottleneck. While the bottleneck is the same for both network

models, flows traverse multiple hops in the peer-to-peer network model as opposed to one hop in the cellular network model. Therefore, it seems intuitive that the end-to-end throughput achieved in the peer-to-peer network model will be lower than that in the cellular network model. *However, we contend that the mere existence of multi-hop flows is the not main reason for the lower throughput performance in the peer-to-peer network model.*

To understand the performance of the peer-to-peer network model in the presence of base station bottleneck and multi-hop routing, we consider a packet scheduler that schedules the end-to-end transmissions of flows in the network. A flow is inserted into the schedule (of finite time slots) if and only if all of its link transmissions can be scheduled without interfering with the link transmissions of other flows already in the schedule. Now denote the utilization $\sigma(\psi)$ of a schedule ψ as the total number of link transmissions summed over all slots. Let Ψ^{τ} be a schedule with the minimum number of slots τ such that each flow in the network is scheduled at least once, and let Ψ^{τ}_{max} be the schedule with the maximum utilization. We can then calculate the spatial reuse κ in the network as $\kappa = \sigma(\Psi^{\tau}_{max})/\tau$. In other words, unlike related work that considers independent one-hop transmissions and hence fails to address the bottleneck in the downstream of multi-hop flows [52, 71], here we consider only link transmissions that can contribute to successful end-to-end transmissions of flows in the network. It is clear that the value of such *pragmatic* spatial reuse achieved not only depends on the distribution of nodes, but also depends on the distribution of flows in the network.

Figure 38(a) shows the change in spatial reuse when the distribution of flows varies. Each data point is an average of results obtained using 10 random network topologies. For each topology, the total number of flows is kept the same, but the percentage of non-local flows changes. Evidently, as the percentage of non-local flows increases, the base station gradually becomes the bottleneck, and hence the number of simultaneous link transmissions that can contribute to end-to-end transmissions decreases (from 11.01 at 0% non-locality to 3.52 at 100% non-locality), accounting for the throughput degradation in the peer-to-peer network model. While the spatial reuse does decrease, we note that the average hop count for all flows in the network also decreases from 4.90 to 3.57. This is an interesting result since it shows that in the peer-to-peer network model, even with 100% non-local flows (the target environment of this work), on average there should be one ($\approx \frac{3.52}{3.57}$) complete end-to-end transmission per slot – same as the one-hop flow in the cellular network model!



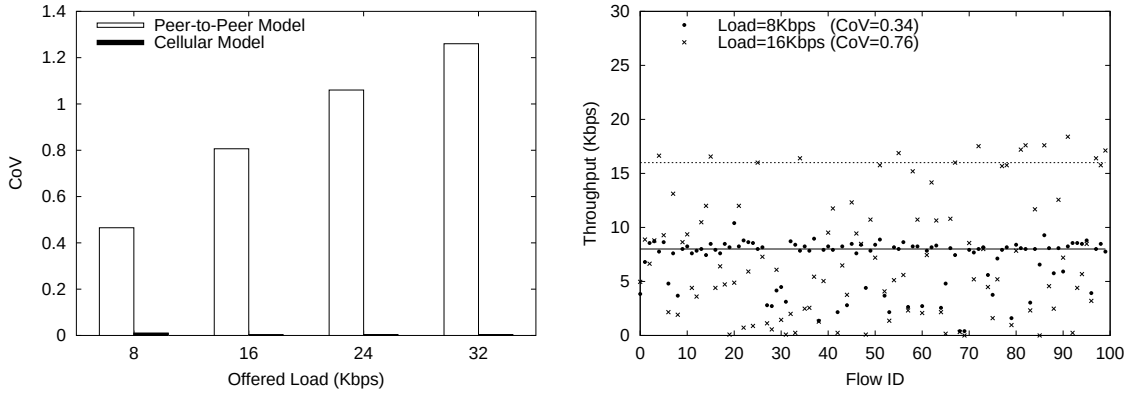
(a) Base-Station Bottleneck

(b) Protocol Inefficiencies

Figure 38: Accounting for Performance Degradation

Therefore, even with base station bottleneck and multi-hop routing, ideally the peer-to-peer network model should be able to achieve the same throughput performance as the cellular network model.

The reason why the peer-to-peer network model achieves a much lower throughput than the cellular network model as shown in Figure 37, is in fact due to the overheads and inefficiencies of the protocols used. The protocols used in the cellular network model are centralized, and operate over a single hop with the base station performing most of the coordination. However, in the peer-to-peer network model, the protocols used are distributed (e.g. IEEE 802.11 DCF mode and DSR), and operate over multiple hops. The inefficiencies that arise because of the distributed operation at the medium access and routing layers [18, 53, 117], and the multi-hop operation at the transport layer [51] (the multi-hop path results in more variations in latency and losses that significantly impact the throughput performance of TCP) translate into a further degraded performance. Figure 38(b) provides a visualization of the inefficiencies introduced in the peer-to-peer protocol stack that we use for simulation. We measure the successful one-hop (link) transmission at the MAC layer of each node, and plot the sum of link throughputs achieved in the entire network when the offered load varies from 4Kbps to 48Kbps. It is clear that as the offered load increases from 4Kbps to 8Kbps, link utilization increases leading to increased link throughput. However, as the offered load increases beyond 8Kbps, the achieved throughput decreases significantly. Note that the aggregate link throughput is in fact a measure of the pragmatic spatial reuse described earlier when



(a) Coefficient of Variation

(b) Throughput Distribution

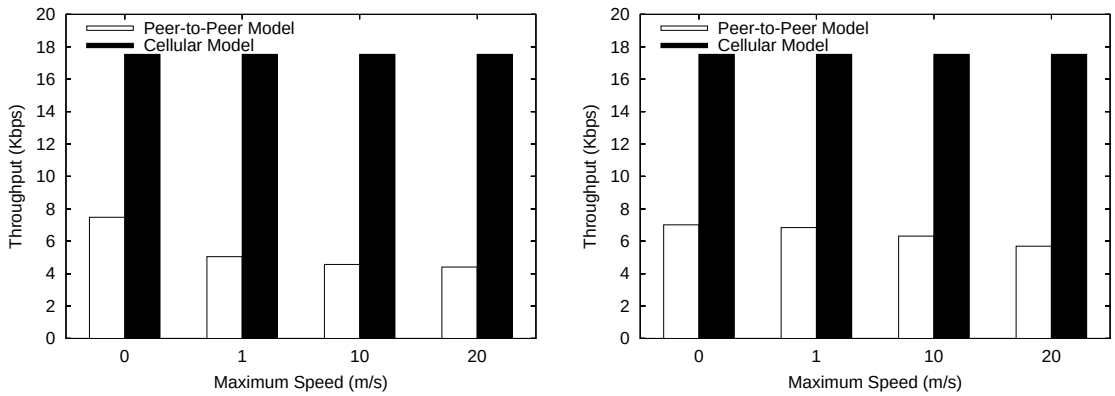
Figure 39: Throughput Fairness

TCP is used as the transport layer protocol. If we divide the peak aggregate throughput (3.44Mbps) by the data rate of the channel (2Mbps), we find that the maximum spatial reuse achieved is 1.72. Compared with the value of 3.52 at the 100% non-locality point in Figure 38(a), the spatial reuse achieved for the peer-to-peer protocols used climbs to only about half of the ideal value. It falls afterward due to the increased inefficiencies of the protocols used at higher traffic load. (Recall that Figure 38(b) only accounts for successful one-hop data transmissions.) We note that the decrease in the maximum spatial reuse achieved (about half) also accounts for a corresponding decrease in the maximum throughput achieved shown in Figure 37.

7.3.2 Fairness

The average throughput presented in Section 7.3.1 is a measure of the aggregate performance achieved by all flows in the network. However, it does not portray the distribution of individual throughputs. In this section we present results demonstrating the fairness properties (or lack thereof) of the two network models. Intuitively, since the base station in the cellular network model performs a centralized round-robin scheduling, it is possible to achieve the ideal fairness. However, in the peer-to-peer network model, because the protocols used are distributed, and the source node can be located randomly in the network, the throughputs enjoyed by individual flows can be very different.

To capture the degree of unfairness, we use the coefficient of variation (CoV) by normalizing



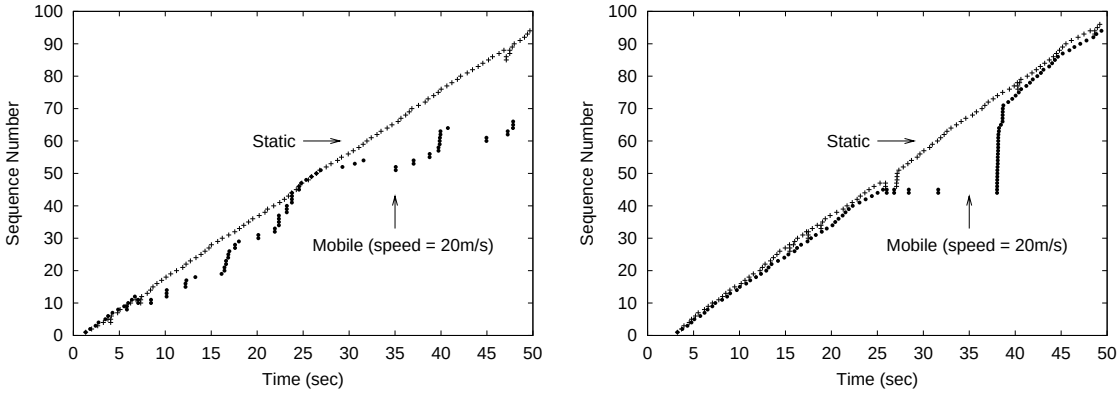
(a) DSR (b) DSDV

Figure 40: Impact of Mobility (Average Throughput)

the standard deviation to the mean [47]. Figure 39(a) compares the CoVs of the peer-to-peer and cellular network models for the same scenario used in Figure 37(a). It is clear that the peer-to-peer network model *does* exhibit a much higher degree of variation than the cellular network model. Figure 39(b) gives a closer look at the unfairness property of the peer-to-peer network model, where the throughput distributions of flows in the network with 8Kbps and 16Kbps offered loads are compared. It can be observed that due to the protocol inefficiencies, the peer-to-peer network model is inherently unfair even when the network is not heavily loaded. Moreover, unfairness increases when the offered load increases. Note that the only difference between the two scenarios is the data rate generated at the source. Ideally, the effect of increasing offered load should be the increase in the utilization of the channel that was otherwise not fully utilized when the offered load is low. However, by comparing the throughput distributions for the two offered loads, we find that while some flows do achieve higher throughput for increasing load (e.g. $f99$), some fail to retain the already achieved throughput, and instead suffer from throughput degradation (e.g. the throughput of $f11$ drops from 7.8Kbps to 3.6Kbps). This is another exposition of the inefficiencies of the distributed protocols used in the peer-to-peer network model.

7.3.3 Mobility

We have thus far considered only static scenarios where the hosts are not mobile during the entire simulation period. In this section, we present results for mobile scenarios using the waypoint



(a) Degradation and Fluctuations

(b) Burstiness

Figure 41: Impact of Mobility (Instantaneous Throughput)

mobility model described in Section 7.1. Figure 40(a) and Figure 40(b) compare the throughput performance of the two network models with increasing mobility, by using DSR and DSDV routing protocols respectively. We observe that the throughput in the cellular model is unaffected by mobility since we consider only a single cell in this work. However, the performance of the peer-to-peer model degrades with increasing mobility due to the overheads caused by route failures and route re-computations, associated losses, and the transport protocol's reaction to such phenomena [8, 51].

To understand the microscopic behavior of the peer-to-peer network model in the presence of mobility, we show in Figure 41(a) and Figure 41(b) a sample trace of TCP's sequence number progression when DSR and DSDV are used respectively. For each figure, we plot the sequence number of the same flow in static and mobile scenarios. It is clear from Figure 41(a) that in the mobile scenario, the sequence number progression is intermittent with pauses as long as 5 seconds. Compared with the static scenario, we observe that not only does mobility cause degradation of the average throughput (refer to the sequence number at $t = 50s$), but it also introduces fluctuations in the instantaneous throughput (observe the frequent changes in the slope). In addition, mobility may induce undesired behaviors of the protocols used in the peer-to-peer network model, such as the "burstiness" shown in Figure 41(b). Note that after the route failure is recovered at $t = 38s$, TCP bursts out all packets queued in its socket send buffer that it failed to transmit when the route was broken. Therefore, although the sending rate of the application used is only 16Kbps, a total of 30

packets are burst out during $t = 38s$ and $t = 40s$, leading to an effective sending rate of 120Kbps! Even the sequence number in the mobile scenario manages to keep up with that in the static scenario at $t = 50s$, such burstiness is clearly undesirable, as it may result in packet losses, or aggravating the problem of protocol inefficiencies at high *induced* load (e.g. the unfairness problem discussed in Section 7.3.2).

7.4 Summary

Although the peer-to-peer network model enjoys better spatial reuse due to the use of short-range transmissions, we have shown in this chapter that it does not achieve better throughput performance compared to the cellular network model in the Internet access scenario. We also showed that the peer-to-peer network model exhibits poor fairness characteristics and suffers from further throughput degradation with increasing host mobility. We summarize the key reasons for the drawbacks of the peer-to-peer model as follows:

- *Base Station Bottleneck:* Although the peer-to-peer network model uses short-range transmissions and hence increases the degree of spatial reuse in the network, such spatial reuse gain cannot be fully utilized in a cellular environment. Specifically, since every flow in the cell is destined for servers in the backbone network, the base station is used as the destination of all traffic in the wireless domain. This results in the channel around the base station becoming a bottleneck, thus limiting the throughput performance of the peer-to-peer network model to that of the cellular network model.

- *Protocol Inefficiencies:* The peer-to-peer network model has traditionally been used only in stand-alone network environments, and hence the network protocols used are typically distributed in nature with more focus on robustness and overall performance rather than on providing good per-flow service. As a result of the distributed operations, the protocols are inherently inefficient and in turn result in the peer-to-peer network model exhibiting poor throughput and unfairness in a cellular environment.

- *Host Mobility:* The peer-to-peer network model uses short-range transmissions and multi-hop routes. Mobility at any of the nodes along the multi-hop routes thus can potentially introduce

route failures and even network partitions. On the other hand, in the cellular network model, mobile hosts use large transmission powers to directly reach the base station, and the only mobile node along the route is the source itself. The impact of mobility is therefore greater in the case of the peer-to-peer model than in the case of the cellular model. Mobility not only makes mobile hosts suffer from throughput degradation, but also causes throughput fluctuations, and further exposes the inefficiencies of the protocols used.

Notwithstanding the drawbacks of lower throughput performance, the peer-to-peer network model *does* achieve better performance than the cellular network model in terms of lower power consumption as we showed in Section 7.2. In fact, one of the key reasons to pursue approaches to improve the throughput performance of the peer-to-peer network model is to leverage its power consumption (short-range transmission) benefits. In the rest of the work, we propose two fundamental principles called *base station assistance* and *multi-homed peer relay* that can be incorporated in the peer-to-peer network model for addressing the drawbacks of the peer-to-peer network model in cellular wireless data networks. The first principle leverages assistance from the base station, while the second principle leverages the relaying capability of multi-homed peer hosts. We show that when the two principles are used in tandem with the peer-to-peer network model, better performance in terms of higher throughput, fair service, and resilience to mobility can be achieved.

CHAPTER 8

BASE STATION ASSISTANCE

In a pure peer-to-peer network model, the base station simply serves as the destination of the multi-hop routes in the wireless domain, without performing any special operations different from other mobile hosts. In this chapter, we propose *base station assistance* as the first principle where the base station plays an active role in assisting the operations of the peer-to-peer network model. We present two instantiations of the principle: (i) *assisted protocols* that use the base station to assist in developing more efficient and effective protocols for the peer-to-peer network model, and (ii) *dual-mode operation* that allows flows in the network to dynamically switch between the cellular and peer-to-peer modes of operation for achieving better performance, in accordance with the network conditions and the protocols used. We use network simulation to show the effectiveness of the base station assistance in addressing protocol inefficiencies and network artifacts in the peer-to-peer network model.

8.1 Assisted Protocols

One option to solve the problem of protocol inefficiencies is to develop better distributed algorithms. However, another feasible option in the network environment we consider is to leverage the existence of the base station and the availability of control channels that exist between the base station and mobile hosts in the cell. Because the base station is more resourceful (in terms of electric and computing powers) and omniscient (for auditing and accounting purposes) than mobile hosts, it can be used not only to reduce overheads due to distributed operations, but also to realize a protocol that is otherwise infeasible using only distributed algorithms, such as load-balanced routing and topology control [52, 53, 81]. Although base-station-assisted protocols can be developed for any layer of the protocol stack, in this section we focus on an approach called *assisted scheduling* where the base station assists the channel access of mobile hosts in the network.

The IEEE 802.11 MAC protocol [56], while popularly used as the *de facto* medium access control standard in the peer-to-peer network model, is primarily designed for a wireless LAN environment and has been shown to fare badly in terms of throughput and fairness achieved in a multi-hop wireless network [53, 117]. Specifically, it is shown in [53] that the performance limitation of the IEEE 802.11 MAC protocol in multi-hop networks results from (i) the use of *RTS-CTS handshakes* that shut down any host around the transmitter or the receiver, and (ii) the design of a *node-based fairness model* that penalizes hosts supporting a larger number flows. An ideal MAC protocol for multi-hop wireless networks should maximize spatial reuse by allowing any host not interfering with existing transmissions to transmit or receive. Moreover, it should support a flow-based fairness model by considering end-to-end transmissions for resolving contention in channel access [53]. *While it is non-trivial to implement such an ideal protocol using purely distributed algorithms, we now discuss how the base station can be used to assist in realizing the ideal MAC protocol.*

Essentially, in assisted scheduling, the base station periodically draws up a schedule for multi-hop transmissions within the network based on the information provided by mobile hosts. The transmission schedule maximizes throughput subject to fair per-flow service. The base station then broadcasts the schedule to the mobile hosts through the control channel. Note that the role of the base station in the proposed approach is very similar to that in a conventional cellular network [41]. The difference however lies in the fact that the scheduling is now done for multi-hop flows instead of individual mobile hosts. Figure 42 lists the variables used in the assisted scheduling algorithm, and Figure 43 presents the algorithm.

$L(n)$ → location information provided by mobile host n
$R(f)$ → route used by flow f
$B(f)$ → service backlog of flow f
$S(f)$ → service enjoyed by flow f
CM → connection matrix of the network
SR → transmission schedule to be released
SO → schedule carried over to the next scheduling period
LF → list of flows with service backlog
sp → length of the transmission schedule

Figure 42: Assisted Scheduling Algorithm (Variables)

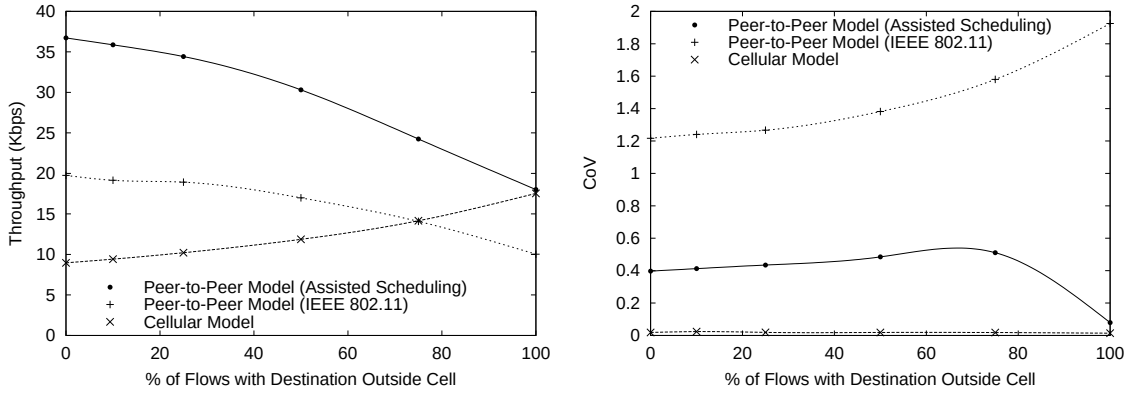
Periodically, the mobile host n uses the control channel to update the base station with the

```
Mobile Host and Base Station Communication
    From each mobile host n:
1       Send neighbor list L(n) (or GPS information)
2       Send flow backlog B(f) for flow f with source node n
    From base station:
3       Broadcast transmission schedule SR

At Base Station
    State Maintenance:
4       Whenever L(n) is updated:
5           Update the connection matrix CM of the network based on L(n)
6           Obtain new route R(f) for every flow f using CM
    Scheduling:
7       Initialize SO to ∅
8       At time t, draw a new transmission schedule with sp time slots:
9           Initialize SR to SO
10          Put every new backlogged flow f in LF
11          While LF is not empty:
12              Find flow f with minimum service S(f)
13              status ← Schedule-flow(f, t)
14              if status is SUCCESS
15                  decrement backlog counter B(f)
16                  increment service counter S(f)
17                  if B(f) is zero
18                      remove f from LF
19              else
20                  remove f from LF
    Schedule-flow(f, t)
21      sno ← t
22      Traverse each hop h of R(f) starting from the first hop
23          Find the first slot s >= sno such that transmission on hop h
            will not interfere with transmissions already scheduled in slot s
24          if (s < t + sp)
25              schedule hop h in slot s and update SR
26          elseif h is not the first hop
27              schedule hop h in slot s and update SO
28          else
29              return FAILURE
30          sno ← s + 1
31      return SUCCESS
```

Figure 43: Assisted Scheduling Algorithm (Pseudo-Code)

location information $L(n)$, which can be in the form of a neighbor list or the GPS location (line 1). It also informs the base station of the service backlog $B(f)$ of the flow f for which it acts as the source node (line 2). Upon receiving the updated information, the base station computes a connection matrix *CM* of the network and the optimal route $R(f)$ for each flow f (lines 4-6). For each scheduling period (every sp time slots), the base station iterates through the list of flows with backlogged services (line 10), and schedules for each flow the hops along the path that the flow traverses (lines 21-31). Once all the flows are accommodated within the schedule, the base station iterates once again through the schedule and attempts to fill in more end-to-end transmissions for the



(a) Throughput

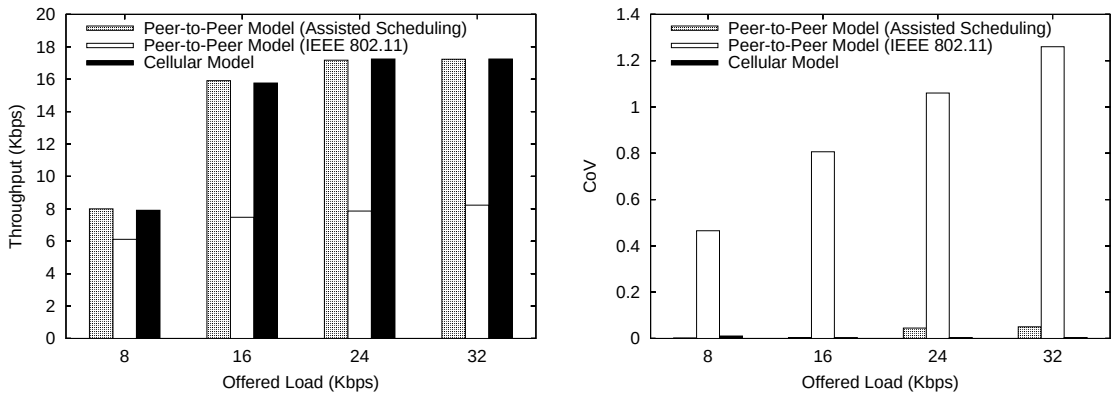
(b) Fairness

Figure 44: Assisted Scheduling Performance (Traffic Locality)

flows within the schedule (lines 11-20). The process is repeated until the schedule cannot be filled in with any flow. A transmission schedule *SR* of length *sp* is broadcast to mobile hosts every *sp* time slots (line 3). Unlike in related approaches [71], the base station at all stages tries to provide fair service before trying to enhance throughput. In other words, when the "refilling" process is done, flows with less service are provided priority over flows with more service. Flows that have schedules beyond the current scheduling period, have slots reserved during the next scheduling period (*SO*) irrespective of the newly contending flows during the next scheduling operation (i.e. flows once scheduled are not preempted).

At each mobile host, a single output queue is maintained for all packets to be forwarded. When the MAC layer requests for a packet from a specific flow (according to the schedule drawn by the base station), a *selective dequeue* mechanism is used to dequeue the first packet (relative to the head of the queue) that belongs to that flow. For results presented in this section, the base station computes a shortest-path route for each flow. Note that more optimal routes (e.g. load-balanced routes [53]) can be chosen by the base station, although at the expense of more complexity.

In Figure 44 we show the performance of the peer-to-peer network model using assisted scheduling when the percentage of non-local flows varies from 0% to 100%. For reference, we also show the performance of the cellular network model and the peer-to-peer network model using purely distributed protocols. We observe that assisted scheduling greatly improves the performance of the



(a) Throughput

(b) Fairness

Figure 45: Assisted Scheduling Performance (Traffic Load)

peer-to-peer network model. Although its performance degrades as the number of flows traversing through the base station increases (due to the increased bottleneck at the base station), it achieves the same performance as the cellular network model when the percentage of non-local flows is 100% – as opposed to worse when using the IEEE 802.11 MAC protocol that we showed in Figure 36(a). The fairness is also improved due to more flows enjoying substantial higher throughputs. As traffic non-locality increases, more and more flows traverse through the same bottleneck, and hence the CoV decreases in the case of assisted scheduling. (The IEEE 802.11 MAC protocol supports a node-based fairness model, and hence its unfairness increases with traffic non-locality.) Figure 45 shows the performance of assisted scheduling with respect to offered load when 100% of the flows are non-local. The performance benefits of assisted scheduling in reducing the protocol inefficiencies for achieving higher throughput and lower unfairness still hold true for varying traffic load.

The performance gains achieved by using assisted scheduling can be attributed to the following reasons: (i) The base station has global information and hence can draw up the optimal schedule for maximizing throughput and fairness. (ii) Since the schedule is constructed in a centralized fashion without requiring the RTS-CTS handshakes between mobile hosts, the spatial reuse is increased. Moreover, the per-flow fairness model is achieved via the centralized flow scheduling algorithm. (iii) The protocol inefficiencies due to contentions and collisions in the distributed operations of the IEEE 802.11 MAC protocol are reduced.

8.2 Dual-Mode Operation

Assisted scheduling presented in Section 8.1 avoids the overheads and inefficiencies in the IEEE 802.11 MAC protocol, and allows the peer-to-peer network model to achieve comparable performance to the cellular network model in terms of throughput and fairness. However, note that the assisted scheduling algorithm presented in Figure 43 could fail if the network is partitioned such that some of the hosts cannot find multi-hop routes to reach the base station. While one possible solution along the line is to develop an *assisted topology control* protocol [52] (for adapting the transmission powers) for use in tandem with the assisted scheduling protocol, in this section we present a different approach called the *dual-mode operation* that does not rely on changing the protocols used in the peer-to-peer network model.

In the proposed approach, the base station supports dual modes of operation: the cellular mode and the peer-to-peer mode. The two modes are provided time-division access to the channel, and the division between the cellular and peer-to-peer modes is based on the number of flows selected to be served in the cellular mode. Flows served in the cellular mode do not receive any service in the peer-to-peer mode and vice versa. However, irrespective of which flows are selected to be served in the cellular mode, all mobile hosts in the network participate in packet forwarding during the peer-to-peer mode. A flow in the peer-to-peer mode will switch to the cellular mode for any non-performance observed in the peer-to-peer mode due to, say, inefficient protocol stacks, topology constraints, or mobility induced degradation. On the other hand, once switched to the cellular mode, the flow will switch back to the peer-to-peer mode whenever the network state is such that it can achieve the desired throughput. *The goal of using the dual-mode operation is to allow individual flows in the network to dynamically choose the mode of operation that can provide better services.* Figure 46 lists the variables used in the algorithm for the dual-mode operation, and Figure 47 presents the algorithm.

In the dual-mode operation, the channel is time-divisioned into periods of length rp, which is further divided between the cellular and peer-to-peer modes (lines 1-2). In the initial state of the network, all flows are served in the peer-to-peer mode. Each mobile host periodically monitors the throughput in an interval of length mp (set to multiples of rp), and requests the base station to

n → number of flows in the network
SF → set of flows currently operated in cellular mode
cT → time division allocation for cellular mode
rp → cellular mode repetition period
mp → throughput monitoring period
up → division update period
$Tp(i)$ → route partition timer (timeout=pp) for flow i
$Ts(i)$ → cellular mode sojourn timer (timeout=sp) for flow i
$M(i)$ → mode of operation {`CELLULAR, PEER`} for flow i
$P(i)$ → peer-to-peer mode connectivity {`PARTITION, CONNECT`} for flow i
$g(i)$ → throughput over mp for flow i
$G(i)$ → aggregate throughput for flow i
$R(i)$ → reference throughput for flow i

Figure 46: Dual-Mode Operation Algorithm (Variables)

serve its flow in the cellular or peer-to-peer mode (lines 3-6) based on the observed performance. Specifically, a mobile host i keeps track of the short-term throughput $g(i)$ (over the last mp period), and long-term throughput $G(i)$ (since its inception) achieved. A mobile host switches its flow to the cellular mode only if both the short-term and long-term throughputs are lower than the reference throughput $R(i)$ it would have observed in a pure cellular network. The reference throughput is a lower bound on the throughput a mobile host desires to enjoy, and is decided when the mobile host initially joins the network. For simplicity we assume that the reference throughput is $\frac{C}{n}$ for flows with the destination outside the cell.

Another reason for a flow in the peer-to-peer mode to switch to the cellular mode is due to mobility induced network partitions. Mobile hosts discover a network partition via the callback from the peer-to-peer routing protocol (lines 24-30). A mobile host operating in the peer-to-peer mode considers itself partitioned from the destination and switches its flow to the cellular mode if route errors last for more than a duration of length pp (lines 31-35). Note that even if route errors do not trigger a switch to the cellular mode, the mobile host may still switch to the cellular mode due to throughput degradation (lines 3-4). When a mobile host i switches its flow to the cellular mode, a timer $Ts(i)$ is associated with the flow (line 20). The timeout sp is the amount of time the flow will stay in the cellular mode before it will be reverted back to the peer-to-peer mode (lines 36-37). Flows that are not partitioned anymore, but were switched to the cellular mode because of a partition, are also reverted back to the peer-to-peer mode (lines 29-30).

Every up time units, the base station consolidates requests from mobile hosts to join or leave

At Mobile Host i

Every rp time:

1 participate in cellular mode for cT period
2 participate in peer-to-peer mode for the remaining period

Every mp time:

3 if $M(i)$ is PEER and $g(i) < R(i)$ and $G(i) < R(i)$
4 send **request**[i, JOIN] to the base station
5 elseif $M(i)$ is CELLULAR and $G(i) > R(i)$ and $P(i)$ is CONNECT
6 send **request**[i, LEAVE] to the base station

Selective Dequeue:

7 in cellular mode
8 if $M(i)$ is CELLULAR
9 dequeue only packets belonging to flow i
10 else
11 do not dequeue any packets
12 in peer-to-peer mode
13 if $M(i)$ is PEER
14 dequeue head-of-line packets
15 else
16 dequeue only packets *not* belonging to flow i

Receive **division**[time t, set S]:

17 $cT \leftarrow t$
18 if $i \in S$
19 $M(i) \leftarrow$ CELLULAR
20 start $Ts(i)$ if not set
21 else
22 $M(i) \leftarrow$ PEER
23 stop $Ts(i)$ if set

Callback from routing protocol with reason r:

24 if r is ROUTE-ERROR
25 start $Tp(i)$ if not set
26 elseif r is ROUTE-OKAY
27 $P(i) \leftarrow$ CONNECT
28 stop $Tp(i)$ if set
29 if $M(i)$ is CELLULAR and $Ts(i)$ expired
30 send **request**[i, LEAVE] to base station

When partition timer $Tp(i)$ expires:

31 $P(i) \leftarrow$ PARTITION
32 if $M(i)$ is PEER
33 send **request**[i, JOIN] to base station
34 else
35 start route probes until $P(i)$ is CONNECT

When sojourn timer $Ts(i)$ expires:

36 if $P(i)$ is CONNECT
37 send **request**[i, LEAVE] to base station

At Base Station

Every rp time:

38 participate in cellular mode for cT period

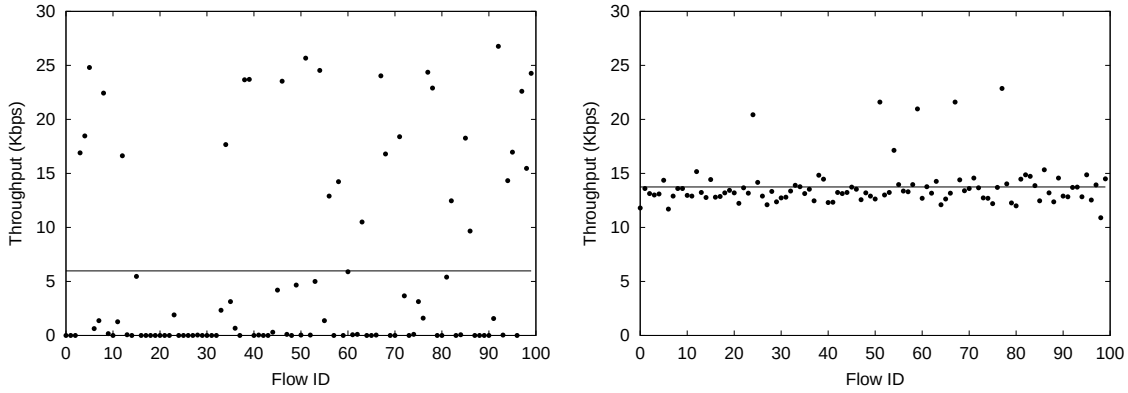
Every up time:

39 $cT \leftarrow rp * \frac{|SF|}{n}$
40 broadcast **division**[cT, SF] to mobile hosts

Receive **request**[mobile host i, action a]:

41 if a is JOIN
42 $SF \leftarrow SF + \{i\}$
43 else
44 $SF \leftarrow SF - \{i\}$

Figure 47: Dual-Mode Operation Algorithm (Pseudo-Code)



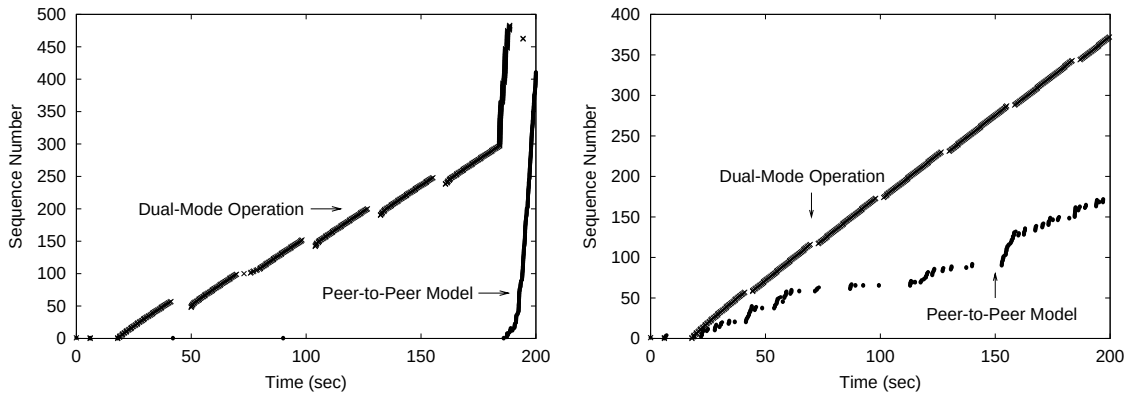
(a) Peer-to-Peer Model

(b) Dual-Mode Operation

Figure 48: Dual-Mode Operation Performance (Fairness)

the cellular mode. It then sends to the mobile hosts the updated set of selected flows SF to operate in the cellular mode, and the corresponding time-division cT to be used for the next up period (lines 39-40). Each mobile host, upon receiving the updated information from the base station, sets its cT timer, updates its mode of operation $M(i)$, and appropriately configures its link layer for selective dequeue. A mobile host in the cellular mode can dequeue packets in its link buffer only if it is selected to operate in the cellular mode (lines 7-11). However, in the peer-to-peer mode all mobile hosts dequeue and forward packets normally, except for those flows that can be dequeued in the cellular mode (lines 12-16). When operating in the cellular mode, the base station and mobile hosts communicate directly as in a conventional cellular network. For example, the base station can perform direct polling (as in WLANs) or broadcast a transmission schedule (as in WWANs) to the subset of mobile hosts in SF. Flows coming into the cell from the distribution network are served under the same fairness scheme used for serving outgoing flows.

We now show the performance of the peer-to-peer network model using the dual-mode operation. In Figure 48 we compare the throughput distribution of flows using the pure peer-to-peer network model with that of flows using the dual-mode operation. As we have seen in Section 7.3.2, the peer-to-peer network model exhibits a very high degree of unfairness due to protocol inefficiencies. However, we observe in Figure 48(b) that the throughput distribution using the dual-mode operation is significantly improved. Flows that starve in the peer-to-peer mode are switched to the



(a) Mobility (b) Congestion

Figure 49: Dual-Mode Operation Performance (Instantaneous Throughput)

cellular mode to enjoy a better throughput performance.

Figure 49 shows the sequence number progression of one flow in the pure peer-to-peer network model and the dual-mode operation respectively. Figure 49(a) is obtained in a mobile scenario, while Figure 49(b) is obtained in a static scenario. We observe in Figure 49(a) that due to network partitions and the exponential backoffs in TCP, the flow in the peer-to-peer network model is not able to deliver any packet to the destination until around $t = 185s$, after which the sequence number increases rapidly. When operated using the dual-mode operation, the same flow is switched to the cellular mode during network partitions, as evident from the steady increase in the sequence number progression for the same duration. The "gaps" in the sequence number progression indicate that the flow is periodically switched back to the peer-to-peer mode for possible reversion. Once the network re-configures such that the flow can deliver packets to the base station in the peer-to-peer mode, it is reverted back to the peer-to-peer mode and starts enjoying the improved throughput due to the increase in spatial reuse. Note that even when the network is not partitioned, a flow can still be *unfairly* treated due to topology constraints and protocol inefficiencies. However, as Figure 49(b) shows, irrespective of the reasons causing throughput degradation, such a flow will be served in the cellular mode.

8.3 Future Research Issues

We have described in this chapter two approaches that leverage the existence of the base station to assist the operation of the peer-to-peer network model in cellular wireless networks. While the purpose of the strawman realizations presented in this chapter is to study the performance improvement and to motivate further investigation of such base-station-assisted approaches in the peer-to-peer network model, in the following we discuss several practical considerations and research issues for these approaches.

8.3.1 Communication Overheads

A key component in the base-station-assisted approach is the use of the control channel between the base station and mobile hosts. For example, in assisted scheduling, the control channel is used for each mobile host to convey its location (or neighborhood) information to the base station, and for the base station to broadcast the transmission schedule to all mobile hosts. In the dual-mode operation, the control channel is used for each mobile host to convey the request to join or leave the cellular mode, and for the base station to broadcast the set of flows selected to be operated in the cellular mode as well as the corresponding channel division information. While the use of the control channel may appear to be an overhead, existing cellular networks already have several control channels available for exchanging information between the base station and mobile hosts, such as transmission schedule and power control information [30, 41]. We also note that some WLAN systems already provide the neighborhood information that assisted scheduling uses, in the form of *link quality map* where the link quality and connectivity between each pair of mobile hosts in the network are maintained [32].

8.3.2 Host Complexities

Mobile hosts in the pure peer-to-peer network model are required to implement all protocol functionalities required by the distributed operations such as medium access and routing. However, with the assistance from the base station, such complexities can be greatly reduced. For example, in assisted scheduling, the base station is responsible for computing the transmission schedule based on

the desired fairness model, while mobile hosts merely follow the schedule for channel access without performing any sophisticated operations. However, note that the complexity tradeoff between the base station and mobile hosts can be a design parameter in developing assisted protocols. The base station can, for example, assist the operations of the IEEE 802.11 MAC protocol in terms of collision resolution and hidden terminal detection, without totally taking over the channel access in the peer-to-peer network model. Similarly, for the dual-mode operation, we presented in Figure 47 a mobile-host-centric algorithm, where mobile hosts are responsible for monitoring throughputs, detecting network partitions, and deciding the appropriate mode of operation. However, as we have shown in [54], it is possible to shift the complexities to the base station by implementing a base-station-centric algorithm, where the base station tracks the network topology and makes the switching decision for each flow, if so desired.

8.3.3 Scheduling Algorithm

We use assisted scheduling as an example to show the performance benefits of base-station-assisted protocols. The concept of centralized scheduling is however not new in cellular networks and multi-hop packet networks. For example, in multi-hop packet radio networks, (base-station) coordinated channel access schemes such as "spatial TDMA" (STDMA) [79, 87] have been used to increase the capacity of the network. HIPERLAN/2 networks also use TDMA-based centralized scheduling for maximizing channel utilization and supporting QoS requirements. Nonetheless, as we showed in [53], in multi-hop wireless networks, the flow scheduling algorithm presented in Section 8.1 achieves better performance than the node or link scheduling algorithm used in STDMA-like approaches. We note that the algorithm presented in Figure 43 is simplistic than sophisticated, and it does not handle issues such as mobility, variable packet size, and reverse traffic. However, related work exists that addresses similar issues, and hence can be incorporated for refining the assisted scheduling algorithm – e.g. topology-transparent schedules that are immune to topology change due to node mobility [26], traffic-controlled schedules that adapt to the dynamics of the traffic [42], and cross-layer schedules that consider the joint optimization of routing, scheduling and power control [77].

8.3.4 Mode Multiplexing

In the algorithm presented for the dual-mode operation, the cellular and peer-to-peer modes are provided time-division access to the channel. However, the framework does not stipulate a specific channel division scheme, and other schemes like frequency-division or code-division can also be employed. Moreover, the framework is not confined to a single-channel environment (such as IEEE 802.11 WLANs), but can be extended to a multi-channel environment (such as 3G WWANs) where individual mobile hosts are provided different channels to communicate with the base station. In particular, if mobile hosts use multi-channel MAC protocols for peer-to-peer communication (see, for example, related work presented in [115]), the dual-mode operation can be considered as one that optimally divides the number of channels (from the pool of channels) to be used in the cellular and peer-to-peer modes.

Note that in Figure 47, we restrict flows to be served exclusively either in the cellular or peer-to-peer mode at any time instant, in order to prevent out-of-order arrivals from impacting TCP's performance. However, TCP may still react adversely in such a dual-mode operation for the following two reasons: (i) When a flow switches between the cellular and peer-to-peer modes, it is possible that packets transmitted in the peer-to-peer mode just before entering the cellular mode will be delivered later than those transmitted during the cellular mode, and vice versa. When channel capacities are high, there can be enough out-of-order packets to trigger a fast retransmit at the TCP source, thus leading to an unnecessary cutdown of the window size. (ii) Packet losses in the peer-to-peer mode may cause TCP to experience timeouts and reset its window size. If later the flow is switched to the cellular mode and offered transmission slots, TCP will not be able to transmit because the window size is reset. Evidently, the fundamental problem is that TCP is designed for a single path, and it cannot simultaneously use multiple paths (or multiple modes in this context) exhibiting very different characteristics. One solution to relieve these constraints is to use a multi-state transport protocol as we proposed in Part I such that TCP can maintain one state for each mode independently. The progression of the two modes therefore will be decoupled without interfering with each other, and thus flows can be allowed to operate in both modes simultaneously to achieve the aggregate throughput.

CHAPTER 9

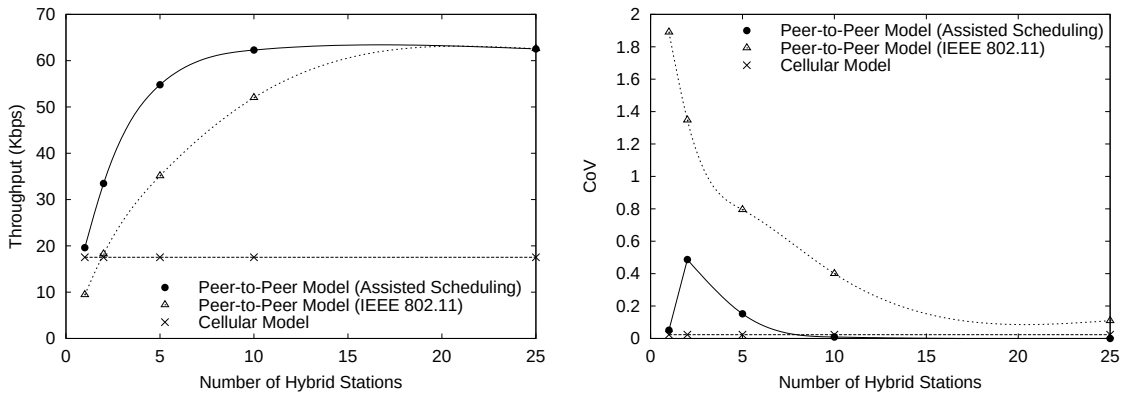
MULTI-HOMED PEER RELAY

We have shown in Chapter 8 that, by seeking the assistance from the base station, the peer-to-peer network model can achieve significant performance improvement in terms of throughput, fairness, and resilience to mobility. However, its performance is still limited to that of the cellular network model due to the base station bottleneck. While a viable option to alleviate the channel bottleneck around the base station is to perform better dimensioning of the cell (e.g. reducing the cell size) and thus limit the number of mobile hosts that the base station has to serve, in this chapter we explore along another dimension that leverages the fundamental operation of the peer-to-peer network model, namely peer relays. Briefly, we extend the peer-to-peer network model from relaying traffic purely within one network domain to relaying traffic between two different network domains – with the help of hosts that have multiple network interfaces. Such multi-homed hosts that have Internet access through any of the available network interfaces can be used to relay traffic (destined to the backbone network) from one interface to another, therefore serving as the destinations for flows in the network with bottleneck. Since the base station of the bottleneck network is no longer the only destination for Internet access, the channel bottleneck around the base station can be relieved. Note that the cellular network model will not be able to benefit from the availability of these multi-homed hosts due to its peer-agnostic nature, where the base station is the single point of communication for all mobile hosts in the network. In this chapter, we propose *multi-homed peer relay* as the second principle to be incorporated in the peer-to-peer network model. We present two instantiations of the principle that allow the peer-to-peer network model to remove the base station bottleneck observed in cellular wireless networks: (i) relay between the wireless and wired networks using hybrid stations, and (ii) relay between heterogeneous wireless networks using multi-mode mobile hosts. We use network simulation to show the effectiveness of the multi-homed peer relay in leveraging the base station bottleneck in cellular wireless environments, and achieving the spatial reuse gains provided by the peer-to-peer network model.

9.1 Hybrid Stations (Wireless–Wired Relay)

With the large number of wired static hosts in the Internet geographically spread, we consider an approach where static hosts (equipped with both wired and wireless interfaces) can optionally "subscribe" to wireless services. Such *hybrid stations* use the same peer-to-peer network protocols, and act like any other mobile hosts in the wireless domain. However, their primary role will be to serve as relays between the wireless domain and the wired Internet using the backbone connectivity that they possess, thus alleviating the base station bottleneck. *Note that the hybrid stations will not perform the role of a base station such as channel allocation and packet scheduling.* Instead, like other mobile hosts in the cell, they use the wireless service provided by the base station. The re-configuration from the regular Internet hosts to the hybrid stations thus can be as simple as plugging in a wireless interface card, which can be motivated by free wired access (via cable or ADSL modems) or a share of the revenue from the wireless service provider. Hybrid stations are hence distinguished from the "wireless routers" or "forwarding terminals" proposed in related work that require the use of dedicated channels, or changes to channel access standards [29].

While the hybrid stations participate in peer-to-peer communication like any other mobile hosts, the key difference in their functionality from regular mobile hosts is that they can relay packets received from the wireless interface to the wired network. Specifically, the hybrid station, upon receipt of a packet from its wireless interface, sends out the packet through its wired interface. The packet is then routed to the base station in the wired network without traversing the multi-hop routes in the wireless domain. On the reverse path, a packet from the remote corresponding host destined for the mobile host will reach the base station as usual. The base station then re-routes the packet to the hybrid station that serves the mobile host the packet is destined for. Upon receipt of the packet from its wired interface, the hybrid station sends out the packet through its wireless interface, and routes the packet to the target mobile host using the peer-to-peer routing protocol (e.g. DSR). Note that hybrid stations are chosen to serve the mobile host by the wireless routing protocol in the route discovery phase (recall that the hybrid station functions like other mobile hosts in the wireless domain), and handoffs from one hybrid station to another are performed as the result of wireless routing protocol switching routes.



(a) Throughput

(b) Fairness

Figure 50: Hybrid Stations Performance

The principal advantage gained from using hybrid stations is the removal of the channel bottleneck surrounding the base station. In Figure 50, we present simulation results for the performance of the peer-to-peer network model in the presence of hybrid stations. We randomly place the hybrid stations in the network, and increase the number of hybrid stations from 1 to 25. The per-flow data rate for each of the 100 mobile hosts is set to 64Kbps. We compare the throughput and fairness performance of the cellular and peer-to-peer network models using either assisted scheduling or the vanilla IEEE 802.11 MAC protocol. Several observations can be made from the results: (i) The average throughput of the peer-to-peer network model with hybrid stations is significantly better than that of the cellular network model (both for using the vanilla protocol and the assisted protocol). While the throughput achieved in the cellular network model is limited by the bottleneck at the base station, the peer-to-peer network model can leverage the existence of hybrid stations and remove the base station bottleneck. (ii) The number of hybrid stations required is not of the order of the number of mobile hosts in the network. Specifically, as shown in Figure 50(a), in the presence of 10 hybrid stations, the achieved throughput can reach the target flow rate of 64Kbps when assisted scheduling is used. Note that the locations of the hybrid stations are randomly distributed, instead of carefully planned. (iii) Network fairness is improved using the hybrid stations. While we already observed in Figure 45(b) that the fairness provided by the peer-to-peer network model using assisted scheduling is comparable to that provided by the cellular network model, Figure 50(b) shows that

even with the vanilla IEEE 802.11 MAC protocol, the peer-to-peer network model can significantly improve on fairness in the presence of hybrid stations. This is because more and more flows are able to achieve higher throughputs than they would have enjoyed using the cellular network model. Note the increased unfairness using assisted scheduling is due to the increased throughput for mobile hosts near the hybrid stations. As the number of hybrid stations increases, more flows can enjoy the increased throughput and thus the CoV reduces.

9.2 Multi-mode Mobile Hosts (Wireless–Wireless Relay)

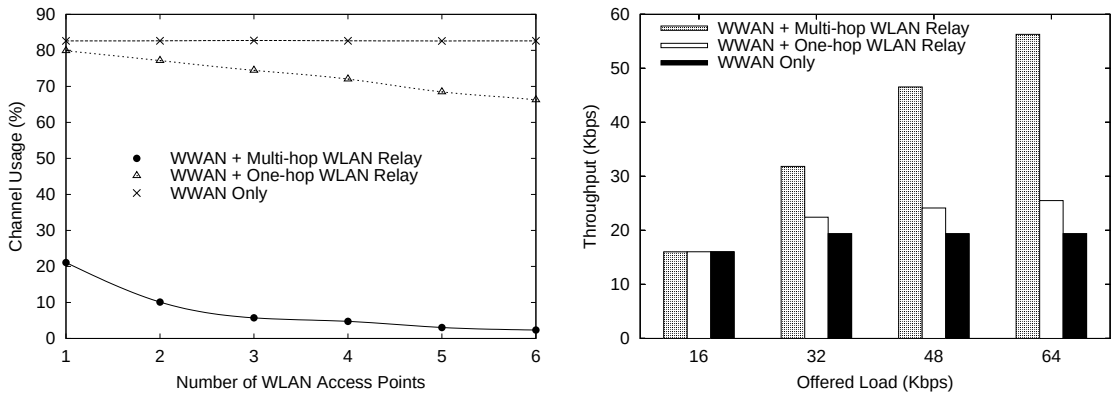
The hybrid stations relieve the channel bottleneck around the base station by relaying traffic between the wired and wireless networks. In this section, we show a different instantiation of multi-homed peer relay – relay between two heterogeneous wireless networks. With the increasing heterogeneity of wireless access technologies, many mobile hosts today are equipped with multiple wireless interfaces that allow them to have "anywhere, anytime" access to the Internet. When such a multi-mode mobile host moves to an area covered by multiple wireless access technologies (where more than one interface is active), it can be used to relieve the base station bottleneck in the more "crowded" network via multi-homed peer relay. For example, consider a scenario where there are scattered WLAN (WiFi) hot spots within a WWAN (3G) macro-cell, and assume that only the WWAN suffers from the base station bottleneck. Since a multi-mode mobile host with both wireless interfaces can access the Internet using either interface, any traffic (destined to the backbone network) generated by other mobile hosts with only the WWAN interface can be relayed to the multi-mode host. Hence, mobile hosts in the WWAN do not need to rely on the base station for Internet access, and the channel bottleneck around the base station can be relieved.

Note that although several approaches have been proposed in a similar context to interwork WWAN and WLAN systems [19, 31, 92], they are different from the multi-homed peer relay proposed in this work. In the WWAN-WLAN interworking architectures proposed in related work, *only mobile hosts with the additional WLAN interface that are within the coverage area of the WLAN* can connect to the WLAN access point, and benefit from the high data rate provided by the WLAN. As soon as the mobile host moves out of the coverage area of the WLAN access points, it needs to fallback to the WWAN connection, and competes for the bottleneck-channel around the

WWAN base station. On the other hand, in the proposed multi-homed peer relay, *any mobile host with a WWAN interface can benefit from the service provided by the WLAN as long as there is at least one multi-mode mobile host within the coverage area of the WLAN,* even if the former is not equipped with the WLAN interface and is not within the coverage of the WLAN access point. This is possible due to the cooperative and multi-hop natures of the peer-to-peer network model used in the multi-homed peer relay: a mobile host can reach the destination in multiple hops via the help of peer hosts, with even multiple network interfaces involved in the traffic relay. Since the WLAN coverage area is typically much smaller than that of the WWAN, the multi-homed peer relay can greatly improve the opportunity for a mobile host to be served through the WLAN, thus achieving higher performance improvement than the interworking approaches.

It is evident that the presence of multi-mode mobile hosts will have a similar effect to the presence of hybrid stations on improving the performance of the peer-to-peer network model that we showed in Figure 50. As the number of multi-homed hosts increases, the throughput and fairness enjoyed by all mobile hosts will improve substantially. In this section, however, we use a slightly different scenario to compare the effectiveness of multi-homed peer relay against the interworking approach in relieving the bottleneck at the base station. We consider the aforementioned WWAN and WLAN environments, and assume that the simulation models described in Section 7.1 are for the WWAN environment. We introduce WLAN access points to the same network grid, but set the channel data rate to 11Mbps and the transmission range to 250m. All mobile hosts are equipped with both WWAN and WLAN interfaces, but they preferentially use the WLAN connection for Internet access. In multi-homed peer relay, if a mobile host fails to find a multi-hop route to any WLAN access point, or it fails to achieve the desired throughput through the WLAN (say, due to increased traffic in the WLAN), it communicates directly with the WWAN base station instead. On the other hand, in the interworking approach, the mobile host uses the WLAN connection only when it is within (one-hop) the coverage area of the WLAN access point. As before, it switches to the WWAN when it fails to achieve the target throughput.

Figure 51(a) shows the channel bottleneck around the WWAN base station when the number of WLAN access points in the network increases from 1 to 6. We divide the WWAN coverage area into a 3×3 array of same-sized grids, and randomly choose a subset of grids to place the



(a) Base Station Bottleneck

(b) Throughput

Figure 51: Multi-mode Mobile Hosts Performance

WLAN access points. Each WLAN access point is located at the center of the chosen grid. We measure the channel usage around the WWAN base station when the target data rate for each of the 100 flows is set to 16Kbps. It is clear that when neither the interworking approach nor the multi-homed peer relay is used (labeled as "WWAN only" in the figure), all flows traverse through the WWAN base station, and hence the channel usage around the base station is kept at 80% or so. When the interworking approach is used (labeled as "WWAN + one-hop WLAN relay" in the figure), the bottleneck reduces as the number of WLAN access points used increases. However, note that the channel usage around the base station is still about 65% with 6 WLAN access points, due to the random movement of the mobile host, and the limited coverage of the WLAN access points. On the other hand, when the multi-homed relay is used (labeled as "WWAN + multi-hop WLAN relay" in the figure), the base station bottleneck can be substantially reduced to less than 5% for the same number and distribution of WLAN access points. The "freed" channel around the base station can then be used to serve more users or increase the data rate of current users. In Figure 51(b) we fix the number of WLAN access points at 3, and show the per-flow throughput achieved when the target data rate of each flow increases from 16Kbps to 64Kbps. It is clear that if no WLAN access point is used, the per-flow throughput is limited to less than 20Kbps due to the base station bottleneck. Otherwise, the presence of WLAN access points helps improve the throughput achieved. However, the achieved throughput in the one-hop interworking approach is significantly lower than

that achieved in the multi-hop, multi-homed peer relay approach. We conclude that using the peer-to-peer network model to *interwork* WWANs and WLANs can achieve a much better performance than using only the cellular network model.

9.3 Future Research Issues

We have shown in this chapter that using multi-homed peer relay in the peer-to-peer network model can effectively overcome the performance limitation imposed by the base station bottleneck. While it is not the focus of this work to propose a complete system based on this principle, in the rest of the section we discuss several research issues in developing the protocols for multi-homed peer relay.

9.3.1 Handoffs

The use of multi-homed hosts such as hybrid stations introduces two different types of handoffs not present in conventional cellular networks. In the following, we use the hybrid stations as an example, and explain how these handoff issues can be addressed: (i) *Handoffs between Hybrid Stations:* Even if the movement of the mobile hosts is confined to within the cell served by the base station, due to the smaller transmission range of the hybrid stations, a mobile host may need to "handoff" from one hybrid station to another during the lifetime of the connection. However, such handoffs are very different from the conventional handoffs between base stations. Since hybrid stations communicate with the mobile host using the peer-to-peer routing protocol, handoffs occur implicitly via the underlying wireless routing protocol switching routes. The route error detection and route recovery procedures employed in such a routing protocol can handle handoffs between hybrid stations, without explicit signaling as in conventional handoffs. (ii) *Handoffs between the Hybrid and Base Stations:* Although hybrid stations are used to relieve the bottleneck at the base station, they themselves can cause bottleneck if they are capacity-limited, or if they serve more mobile hosts than the base station does (say, with only one or two hybrid stations in the cell). Therefore, it may become necessary to handoff from the base station to the hybrid station and vice versa, depending on the service perceived by the mobile host. Such handoffs are triggered due to the degraded quality of service, rather than the degraded quality of signal. However, note that frequent occurrences of such handoffs (e.g. "ping-pong" switching between the base station and the hybrid

station) may cause undesired performance degradation. A simple solution to avoid unnecessary handoffs from the base station to the hybrid station is for the hybrid station to limit the number of connections it serves, based on the number of serving mobile hosts or the quality of service (throughput) each connection receives. A hybrid station will not reply to the route request (from the wireless routing protocol) any mobile host issues when its relaying capacity is reached.

9.3.2 Routing Protocol

In multi-homed peer relay, routing needs to be performed across multiple networks. Assuming that the source mobile host is served by network *A*, and the multi-homed relay host is served by networks *A* and *B*, then multi-homed peer relay involves the routing protocol used in network *A* (between the source and the relay host), and the routing protocol used in network *B* (between the relay host and the corresponding host). For example, in the case of hybrid stations, network *A* uses the wireless peer-to-peer routing protocol such as DSR, while network *B* uses the wired backbone routing protocol such as IP routing.[1] Several design issues need to be considered: (i) *Mobile Host Anycast:* The terminating node of the peer-to-peer routing protocol used in network *A* can be any multi-homed relay host that can serve the source. This can be achieved by using the *anycast* in the peer-to-peer routing protocol. Alternatively, the relay host, upon receipt of the route discovery for the corresponding host, can reply valid routes back to the source. The source can then process the route reply as usual, and choose one with, say, the shortest route to the corresponding host. (ii) *Relay Host Tunneling:* While ideally the relay host can simply pass packets received from network *A* to network *B* without changing the source address, it might not always be the case. If network *B* employs *ingress filtering* [33], then the relay host needs to perform some form of address translation such as tunneling or IP encapsulation to ensure successful packet delivery. (iii) *Base Station Rendezvous:* Although the motivation of using multi-homed relay host is to bypass the channel bottleneck at the base station, to facilitate billing and avoid multi-path routing, it might be desirable to route packets through the base station in the wired domain. However, the disadvantages

[1] We note that communication between the relay and corresponding hosts may involve more than one routing protocols, as is the case for multi-mode mobile hosts described in Section 9.2. However, since we assume that the relay host can communicate with the corresponding host using service provided by network *B*, we collectively refer to these different routing protocols as one.

are the potential bottleneck in the core network the base station resides, and the suboptimal routes between the hybrid stations (or the access points in the case of multi-mode mobile hosts) and the corresponding host. (iv) *Reverse Route Pinning:* Since the source uses the address assigned by network *A*, downstream traffic will be routed to the base station (and network *A*), unless a reverse tunnel is established between the corresponding host and the serving relay host. The base station thus needs to find the relay host that serves the target mobile host if asymmetric routes were to be avoided. A simple solution is for the relay host to periodically update the base station on the list of mobile hosts it is currently serving. Approaches proposed in [23] in a different context may also be used.

9.3.3 Transport Protocol

The multi-homed peer relay alleviates the bottleneck at the base station by providing a different route between the mobile host and the corresponding host that does not traverse through the base station. While effective, the use of an alternative route can potentially impact the performance of reliable transport protocols like TCP, as we discuss in the following: (i) *Multi-Path Routing:* When packets belonging to the same connection are sent through different routes that exhibit mismatched round-trip times, out-of-order arrivals at the receiver can occur and trigger TCP's fast retransmit mechanism to cut down the congestion window unnecessarily. Such multi-path routing is possible in multi-homed peer relay, if handoffs between relay hosts and the base station occur during the lifetime of a connection (as described in Section 9.3.1). When the connection handoffs from, say, the base station to the hybrid station, packets sent through the hybrid station (after the handoff) can arrive earlier than those sent through the base station (before the handoff), due to the delay mismatches in the wireless domain and/or the wired domain – especially if the "base station rendezvous" strategy that we discussed in Section 9.3.2 is not used. In addition to out-of-order arrivals, frequent route switching between multiple paths can render TCP suboptimal. Since TCP performs congestion control based on the characteristics of the underlying path such as the round-trip time and pipe size (which it learns through repeated probing losses), route switching can potentially trigger the adverse reaction of TCP. For example, switching from a route with short RTTs to one with

long RTTs can prematurely cause the retransmission timer to timeout, and shut down the congestion window. On the other hand, switching from a route with long RTTs to one with short RTTs can cause packet losses (due to buffer overflow or congestion) if two routes are of comparable bandwidths. (ii) *Asymmetric Routing:* Asymmetric routing occurs when the forward and reverse traffic of the same connection traverses different routes. In multi-homed peer relay, such asymmetric routing can happen if the data packet traverses through the relay host, and the acknowledgment (ACK) comes back through the base station or a different relay host. Since TCP uses the return of ACKs to clock the transmission of data packets, it will function properly only if the incoming rate of ACKs is equal to the available data rate on the forward path. However, if ACKs traverse a different path with very different characteristics (say, round-trip time) from the forward path, TCP can potentially under-utilize or overload the forward path. We note that the multi-state transport protocols proposed in Part I can be used to address the problem due to multi-path routing in multi-homed peer relay.

CHAPTER 10

CONCLUSIONS

In this dissertation, we investigate the problem of providing mobile hosts with seamless and broadband wireless Internet access in the context of the increasing network heterogeneity and bandwidth scarcity in future wireless data networks. We identify the inability of existing approaches to provide solutions for supporting host mobility and utilizing wireless bandwidth that can scale with the heterogeneity of wireless networks and the proliferation of wireless devices respectively. In addressing network heterogeneity, we motivate a solution based on the transport layer protocol for supporting transparent host mobility across heterogeneous wireless networks. We establish *parallelism* and *transpositionality* as two fundamental principles to be incorporated in designing the transport layer solution. In addressing bandwidth scarcity, we motivate a solution based on the peer-to-peer network model for scaling network capacity with wireless user population. We establish *base station assistance* and *multi-homed peer relay* as two fundamental principles to be used in tandem with the peer-to-peer network model. We present instantiations based on the established principles, and demonstrate through theoretic analysis, packet simulation and testbed emulation their performance benefits.

While we approach the problem from two different perspectives, the solutions proposed are in fact consistent with each other. For example, the drive to develop new network models for improving the performance of wireless networks is a testimony to the increasing heterogeneity of wireless networks. The ability to support host mobility without relying on the specifics of the underlying networks allows mobile hosts to still enjoy seamless connectivity while the new network model is being deployed to support better services. Moreover, since the transport layer solution proposed to address network heterogeneity does not stipulate any specific transport layer functionalities such as congestion control that should to be used, it can accommodate different protocols proposed to optimize the performance of the new network model. We note that while the proposed solutions address the problems where existing approaches fail, the two complement each other in providing

a more holistic solution. For example, infrastructure-based approaches for mobility support can be used for intra-technology handoffs, while end-to-end approaches can be used for inter-technology handoffs. Solutions proposed in different layers of the network protocol stack can work hand-in-hand with solutions target new network models to provide mobile hosts with optimal wireless bandwidth.

In concluding the dissertation, we believe that the contribution of this work lies not only in proposing new solutions with demonstrated performance improvement, but also in identifying new research directions and establishing fundamental principles that future researches can build upon. We believe that future researches can benefit from this work for a more balanced view of the solutions involved in addressing the problems in wireless networking.

REFERENCES

[1] 3GPP, "3GPP TSG-RAN; Opportunity driven multiple access," TR 25.924, V1.0.0, Dec. 1999.

[2] H. Adiseshu, G. Parulkar, and G. Varghese, "A reliable and scalable striping protocol," in *Proceedings of ACM SIGCOMM*, Palo Alto, CA, USA, pp. 131–141, Aug. 1996.

[3] G. Aggélou and R. Tafazolli, "On the relaying capacity of next-generation GSM cellular networks," *IEEE Personal Communications Magazine*, vol. 8, no. 1, pp. 40–47, Feb. 2001.

[4] K. Ahmavaara, H. Haverinen, and R. Pichna, "Interworking architecture between 3GPP and WLAN systems," *IEEE Communications Magazine*, vol. 41, no. 11, pp. 74–81, Nov. 2003.

[5] Akamai, "*Akamai Accelerated Network Program*," `http://www.akamai.com`.

[6] M. Allman, H. Kruse, and S. Ostermann, "An application-level solution to TCP's satellite inefficiencies," in *Proceedings of Workshop on Satellite-Based Information Services (WOSBIS)*, Rye, NY, USA, Nov. 1996.

[7] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF RFC 2581, Apr. 1999.

[8] V. Anantharaman and R. Sivakumar, "A microscopic analysis of TCP performance over mobile ad-hoc networks," in *Proceedings of ACM SIGMETRICS*, Marina Del Rey, CA, USA, pp. 270–271, June 2002.

[9] B. Bakshi, P. Krishna, N. Vaidya, and D. Pradhan, "Improving performance of TCP over wireless networks," in *Proceedings of IEEE ICDCS*, Baltimore, MD, USA, pp. 365–373, May 1997.

[10] H. Balakrishnan and R. Katz, "Explicit loss notification and wireless web performance," in *Proceedings of IEEE GLOBECOM*, Sydney, Australia, Nov. 1998.

[11] H. Balakrishnan, V. Padmanabhan, S. Seshana, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, Dec. 1997.

[12] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LAN's," in *Proceedings of ACM SIGCOMM*, London, United Kingdom, pp. 212–225, Sept. 1994.

[13] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using interarrival times at the receiver," in *Proceedings of IEEE ASSET*, Richardson, TX, USA, pp. 10–17, Mar. 1999.

[14] E. Blanton, M. Allman, K. Fall, and L. Wang, "A conservative SACK-based loss recovery algorithm for TCP," IETF Internet Draft; *draft-allman-tcp-sack-13.txt*, Oct. 2002.

[15] Bluetooth. http://www.bluetooth.com.

[16] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka, "TCP-PR: TCP for persistent packet reordering," in *Proceedings of IEEE ICDCS*, Providence, RI, USA, pp. 222–231, May 2003.

[17] D. Bovet and M. Cesati, *Understanding the Linux Kernel*. Sebastopol, CA, USA: O'Reilly & Associates, Dec. 2002.

[18] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proceedings of ACM MOBICOM*, Dallas, TX, USA, pp. 85–97, Oct. 1998.

[19] M. Buddhikot, G. Chandranmenon, S.-J. Han, Y.-W. Lee, S. Miller, and L. Salgarelli, "Integration of 802.11 and third-generation wireless data networks," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, pp. 503–512, Mar. 2003.

[20] R. Cáceres and V. Padmanabhan, "Fast and scalable handoffs for wireless internetworks," in *Proceedings of ACM MOBICOM*, Rye, NY, USA, pp. 56–66, Nov. 1996.

[21] CAIDA, "Workload characterization," http://www.caida.org/analysis/workload.

[22] M. Callendar, "International Mobile Telecommunications-2000 standards efforts of the ITU (Special Issue)," *IEEE Personal Communications Magazine*, vol. 4, no. 4, pp. 6–7, Aug. 1997.

[23] A. Campbell, J. Gomez, S. Kim, A. Valkó, C.-Y. Wan, and Z. Turányi, "Design, implementation, and evaluation of Cellular IP," *IEEE Personal Communications Magazine*, vol. 7, no. 4, pp. 42–49, Aug. 2000.

[24] A. Campbell, J. Gomez, S. Kim, C.-Y. Wan, Z. Turányi, and A. Valkó, "Comparison of IP micro-mobility protocols," *IEEE Wireless Communications Magazine*, vol. 9, no. 1, pp. 2–12, Feb. 2002.

[25] F. Chiussi, D. Khotimsky, and S. Krishnan, "Mobility management in third-generation all-IP networks," *IEEE Communications Magazine*, vol. 40, no. 9, pp. 124–135, Sept. 2002.

[26] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 23–29, Feb. 1994.

[27] M. Crovella and A. Bestavros, "Self-similarity in the world wide web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[28] M. Elaoud and P. Ramanathan, "Adaptive allocation of CDMA resources for network-level QoS assurances," in *Proceedings of ACM MOBICOM*, Boston, MA, USA, pp. 191–199, Aug. 2000.

[29] N. Esseling, H. Vandra, and B. Walke, "A forwarding concept for HiperLAN/2," *Computer Networks*, vol. 37, no. 1, pp. 25–32, Sept. 2001.

[30] ETSI, "BRAN; HIPERLAN Type 2; System overview," TR 101 683, V1.1.1, Feb. 2000.

[31] ETSI, "BRAN; HIPERLAN Type 2; Requirements and architecture for internetworking between HIPERLAN/2 and 3rd generation cellular systems," TR 101 957, V1.1.1, Aug. 2001.

[32] ETSI, "BRAN; HIPERLAN Type 2; Data link control (DLC) layer; Part 4: Extension for home environment," TS 101 761-4, V1.3.2, Jan. 2002.

[33] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing," IETF RFC 2827, May 2000.

[34] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," IETF RFC 2582, Apr. 1999.

[35] J. Foerster, E. Green, S. Somayazulu, and D. Leeper, "Ultra-wideband technology for short- or medium-range wireless communications," *Intel Technology Journal*, May 2001.

[36] C. Fraleigh *et al.*, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network Magazine*, vol. 17, no. 6, pp. 6–16, Nov/Dec. 2003.

[37] M. Frodigh, S. Parkvall, C. Roobol, P. Johansson, and P. Larsson, "Future-generation wireless networks," *IEEE Personal Communications Magazine*, vol. 8, no. 5, pp. 10–17, Oct. 2001.

[38] Globalstar. `http://www.globalstar.com`.

[39] T. Goff, J. Moronski, D. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, pp. 1537–1545, Mar. 2000.

[40] N. Gogate and S. Panwar, "On a resequencing model for high speed networks," in *Proceedings of IEEE INFOCOM*, Toronto, Canada, pp. 40–47, June 1994.

[41] D. J. Goodman, *Wireless Personal Communications Systems*. Reading, MA, USA: Addison-Wesley, 1997.

[42] J. Grönkvist, "Traffic controlled spatial reuse TDMA in multi-hop radio networks," in *Proceedings of IEEE PIMRC*, Boston, MA, USA, pp. 1203–1207, Sept. 1998.

[43] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*. New York, NY, USA: John Wiley & Sons, 1985.

[44] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 477–486, Aug. 2002.

[45] GSM Association, "The wireless evolution," `http://www.gsmworld.com/technology/`.

[46] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

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

[48] T. Harrold and A. Nix, "Capacity enhancement using intelligent relaying for future personal communication systems," in *Proceedings of IEEE VTC Fall*, Boston, MA, USA, pp. 2115–2120, Sept. 2000.

[49] T. Henderson and R. Katz, “Satellite transport protocol (STP): An SSCOP-based transport protocol for datagram satellite networks,” in *Proceedings of Workshop on Satellite-Based Information Services (WOSBIS)*, Budapest, Hungary, Oct. 1997.

[50] HiperLAN2 Global Forum. http://www.hiperlan2.com.

[51] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” in *Proceedings of ACM MOBICOM*, Seattle, WA, USA, pp. 219–230, Aug. 1999.

[52] H.-Y. Hsieh and R. Sivakumar, “Performance comparison of cellular and multi-hop wireless networks: A quantitative study,” in *Proceedings of ACM SIGMETRICS*, Boston, MA, USA, pp. 113–122, June 2001.

[53] H.-Y. Hsieh and R. Sivakumar, “IEEE 802.11 over multi-hop wireless networks: Problems and new perspectives,” in *Proceedings of IEEE VTC Fall*, Vancouver, Canada, pp. 748–752, Sept. 2002.

[54] H.-Y. Hsieh and R. Sivakumar, “On using the ad-hoc network model in cellular packet data networks,” in *Proceedings of ACM MOBIHOC*, Lausanne, Switzerland, pp. 36–47, June 2002.

[55] H.-Y. Hsieh and R. Sivakumar, “A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts,” in *Proceedings of ACM MOBICOM*, Atlanta, GA, USA, pp. 83–94, Sept. 2002.

[56] IEEE 802.11 Working Group, “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications,” ANSI/IEEE Standard 802.11, Aug. 1999.

[57] IEEE 802.16 Working Group, “Broadband access for wireless metropolitan area networks (WMAN),” http://www.ieee802.org/16/.

[58] IEEE 802.20 Working Group, “Mobile broadband wireless access (MBWA),” http://www.ieee802.org/20/.

[59] I. Iliadis and Y.-C. Lien, “Resequencing in distributed systems with multiple classes,” in *Proceedings of IEEE INFOCOM*, New Orleans, LA, USA, pp. 881–888, Mar. 1988.

[60] Iridium. http://www.iridium.com.

[61] V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” IETF RFC 1323, May 1992.

[62] D. Johnson, D. Maltz, and Y.-C. Hu, “The dynamic source routing protocol for mobile ad hoc networks (DSR),” IETF Internet Draft; *draft-ietf-manet-dsr-09.txt*, Apr. 2003.

[63] W. Kellerer, H.-J. Vogel, and K.-E. Steinberg, “A communication gateway for infrastructure-independent 4G wireless access,” *IEEE Communications Magazine*, vol. 40, no. 3, pp. 126–131, Mar. 2002.

[64] L. Kleinrock and J. Silvester, “Spatial reuse in multihop packet radio networks,” *Proceedings of IEEE*, vol. 75, no. 1, pp. 156–166, Jan. 1987.

[65] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access with bounded slowdown," in *Proceedings of ACM MOBICOM*, Atlanta, GA, USA, pp. 119–130, Sept. 2002.

[66] B. Krishnamoorthi, "On Poisson queue with two heterogeneous servers," *Operations Research*, vol. 11, pp. 321–330, 1963.

[67] Y.-K. Kwok and V. Lau, "Design and analysis of a new approach to multiple burst admission control for cdma2000," in *Proceedings of ACM MOBICOM*, Rome, Italy, pp. 310–321, July 2001.

[68] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proceedings of ACM MOBICOM*, Rome, Italy, pp. 61–69, July 2001.

[69] Y.-D. Lin and Y.-C. Hsu, "Multihop cellular: A new architecture for wireless communications," in *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, pp. 1273–1282, Mar. 2000.

[70] S. Lu, T. Nandagopal, and V. Bharghavan, "A wireless fair service algorithm for packet cellular networks," in *Proceedings of ACM MOBICOM*, Dallas, TX, USA, pp. 10–20, Oct. 1998.

[71] H. Luo, S. Lu, and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks," in *Proceedings of ACM MOBICOM*, Boston, MA, USA, pp. 76–86, Aug. 2000.

[72] H. Luo, R. Ramjee, P. Sinha, L. Li, and S. Lu, "UCAN: A unified cellular and ad-hoc network architecture," in *Proceedings of ACM MOBICOM*, San Diego, CA, USA, pp. 353–367, Sept. 2003.

[73] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proceedings of IEEE ICNP*, Riverside, CA, USA, pp. 165–171, Nov. 2001.

[74] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP-Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proceedings of ACM MOBICOM*, Rome, Italy, pp. 287–297, July 2001.

[75] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining TCP congestion control," in *Proceedings of ACM SIGCOMM*, Palo Alto, CA, USA, pp. 281–291, Aug. 1996.

[76] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," IETF RFC 2018, Oct. 1996.

[77] S. Mengesha and H. Karl, "Relay routing and scheduling for capacity improvement in cellular WLANs," in *Proceedings of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Sophia-Antipolis, France, Mar. 2003.

[78] J. Nagle, "Congestion control in IP/TCP Internetworks," IETF RFC 896, Jan. 1984.

[79] R. Nelson and L. Kleinrock, "Spatial TDMA: A collision-free multihop channel access protocol," *IEEE Transactions on Communications*, vol. 33, no. 9, pp. 934–944, Sept. 1985.

[80] T. Otsu, I. Okajima, N. Umeda, and Y. Yamao, "Network architecture for mobile communications systems beyond IMT-2000," *IEEE Personal Communications Magazine*, vol. 8, no. 5, pp. 31–37, Oct. 2001.

[81] S.-J. Park and R. Sivakumar, "Load-sensitive transmission power control in wireless ad-hoc networks," in *Proceedings of IEEE GLOBECOM*, Taipei, Taiwan, pp. 42–46, Nov. 2002.

[82] C. Perkins, "Mobile IP," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 66–82, May 2002.

[83] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proceedings of ACM SIGCOMM*, London, United Kingdom, pp. 234–244, Sept. 1994.

[84] D. Phatak and T. Goff, "A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments," in *Proceedings of IEEE INFOCOM*, New York, NY, USA, pp. 773–781, June 2002.

[85] D. Porcino and W. Hirt, "Ultra-wideband radio technology: Potential and challenges ahead," *IEEE Communications Magazine*, vol. 41, no. 7, pp. 66–74, July 2003.

[86] J. Postel, "Transmission control protocol," IETF RFC 793, Sept. 1981.

[87] S. Ramanathan and E. Lloyd, "Scheduling algorithms for multi-hop radio networks," in *Proceedings of ACM SIGCOMM*, Baltimore, MD, USA, pp. 211–222, Aug. 1992.

[88] R. Ramjee, K. Varadhan, L. Salgarelli, S. Thuel, S.-Y. Wang, and T. La Porta, "HAWAII: A domain-based approach for supporting mobility in wide-area wireless networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 396–410, June 2002.

[89] S. Rappaport and L.-R. Hu, "Microcellular communication system with hierarchical macrocell overlays: Traffic performance model and analysis," *Proceedings of the IEEE*, vol. 82, no. 9, pp. 1383–1397, Sept. 1994.

[90] M. Riegel and M. Tuexen, "Mobile SCTP," IETF Internet Draft; *draft-riegel-tuexen-mobile-sctp-02.txt*, Feb. 2003.

[91] P. Rodriguez and E. Biersack, "Dynamic parallel-access to replicated content in the Internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 455–464, Aug. 2002.

[92] A. Salkintzis, C. Fors, and R. Pazhyannur, "WLAN-GPRS integration for next-generation mobile data networks," *IEEE Wireless Communications Magazine*, vol. 9, no. 5, pp. 112–124, Oct. 2002.

[93] A. Sanmateu, L. Morand, E. Bustos, S. Tessier, F. Paint, and A. Sollund, "Using Mobile IP for provision of seamless handoff between heterogeneous access networks, or how a network can support the always-on concept," in *Proceedings of EURESCOM Summit*, Heidelberg, Germany, Nov. 2001.

[94] J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP buffer tuning," in *Proceedings of ACM SIGCOMM*, Vancouver, Canada, pp. 315–323, Sept. 1998.

[95] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Dynamic power management for portable systems," in *Proceedings of ACM MOBICOM*, Boston, MA, USA, pp. 11–19, Aug. 2000.

[96] H. Singh and S. Singh, "Energy consumption of TCP Reno, Newreno, and SACK in multi-hop wireless networks," in *Proceedings of ACM SIGMETRICS*, Marina Del Rey, CA, USA, pp. 206–216, June 2002.

[97] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," in *Proceedings of ACM MOBICOM*, Seattle, WA, USA, pp. 231–241, Aug. 1999.

[98] H. Sivakumar, S. Bailey, and R. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *Proceedings of IEEE/ACM Supercomputing (SC)*, Dallas, TX, USA, Nov. 2000.

[99] A. Snoeren, "Adaptive inverse multiplexing for wide-area wireless networks," in *Proceedings of IEEE GLOBECOM*, Rio de Janeireo, Brazil, pp. 1665–1672, Dec. 1999.

[100] A. Snoeren, D. Andersen, and H. Balakrishnan, "Fine-grained failover using connection migration," in *Proceedings of USENIX USITS*, San Francisco, CA, USA, Mar. 2001.

[101] A. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of ACM MOBICOM*, Boston, MA, USA, pp. 155–166, Aug. 2000.

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

[103] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode, "Migratory TCP: Connection migration for service continuity in the Internet," in *Proceedings of IEEE ICDCS*, Vienna, Austria, pp. 469–470, July 2002.

[104] A. S. Tanenbaum, *Computer Networks*. Upper Saddle River, NJ, USA: Prentice Hall, 1996.

[105] The Network Simulator, "*ns-2*," `http://www.isi.edu/nsnam/ns/`.

[106] K. Thompson, G. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network Magazine*, vol. 11, no. 6, pp. 10–23, Nov/Dec. 1997.

[107] V. Tsaoussidis, H. Badr, X. Ge, and K. Pentikousis, "Energy/Throughput tradeoffs of TCP error control strategies," in *Proceedings of IEEE ISCC*, Antibes, France, pp. 106–112, July 2000.

[108] V. Tsaoussidis and C. Zhang, "TCP-Real: Receiver-oriented congestion control," *Computer Networks*, vol. 40, no. 4, pp. 477–497, Nov. 2002.

[109] UMTS Forum. `http://www.umts-forum.org`.

[110] WiFi Alliance. `http://www.weca.net`.

[111] WiMAX Forum. `http://www.wimaxforum.org`.

[112] G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2*. Reading, MA, USA: Addison-Wesley, Oct. 1997.

[113] G. Wu, M. Mizuno, and P. Havinga, "MIRAI architecture for heterogeneous network," *IEEE Communications Magazine*, vol. 40, no. 2, pp. 126–134, Feb. 2002.

[114] H. Wu, C. Qiao, S. De, and O. Tonguz, "Integrated cellular and ad hoc relaying systems: iCAR," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 19, no. 10, pp. 2105–2115, Oct. 2001.

[115] S.-L. Wu, Y.-C. Tseng, C.-Y. Lin, and J.-P. Sheu, "A multi-channel MAC protocol with power control for multi-hop mobile ad hoc networks," *The Computer Journal*, vol. 45, no. 1, pp. 101–110, Jan. 2002.

[116] X. Wu, S.-H. Chan, and B. Mukherjee, "MADF: A novel approach to add an ad-hoc overlay on a fixed cellular infrastructure," in *Proceedings of the IEEE WCNC*, Chicago, IL, USA, pp. 549–554, Sept. 2000.

[117] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130–137, June 2001.

[118] M. Zhang, B. Karp, S. Floyd, and L. Peterson, "RR-TCP: A reordering-robust TCP with DSACK," in *Proceedings of IEEE ICNP*, Atlanta, GA, USA, pp. 95–106, Nov. 2003.

[119] X. Zhao, C. Castelluccia, and M. Baker, "Flexible network support for mobility," in *Proceedings of ACM MOBICOM*, Dallas, TX, USA, pp. 145–156, Oct. 1998.

[120] M. Zorzi and R. Rao, "Is TCP energy efficient," in *Proceedings of IEEE MoMuC*, San Diego, CA, USA, pp. 198–201, Nov. 1999.