Networking over Next-Generation Satellite Systems

by

Thomas Ross Henderson

B.S. (Stanford University) 1990
M.S. (Stanford University) 1991

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

in

Engineering– Electrical Engineering
and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Randy H. Katz, Chair
Professor Dorit S. Hochbaum
Professor Steven R. McCanne

Fall 1999

The dissertation of *Thomas Ross Henderson* is approved:

Chair Date

Date

Date

University of California at Berkeley

Fall 1999

Networking over Next-Generation Satellite Systems

Abstract

Networking over Next-Generation Satellite Systems

by

Thomas Ross Henderson
Doctor of Philosophy in Engineering– Electrical Engineering and Computer Sciences

University of California at Berkeley

Professor Randy H. Katz, Chair

Thanks to both the rapid deployment of the Internet and advances in satellite technology, the market for broadband satellite services is poised for substantial growth in the coming decade. Current communications satellite systems have generally been designed to provide either voice or data transaction (low data rate) services through small terminals, or trunking (high data rate, or broadband) services through large terminals. However, technological advances are enabling new systems that combine broadband data rates with small terminals, thereby providing more affordable "last-mile" network access to home and small business users worldwide. In particular, two types of broadband satellite systems are under development: high-power satellites deployed at traditional geostationary (GEO) orbits, and large constellations of satellites deployed at much lower (LEO) orbits.

In this thesis, we explore research problems that have arisen from this shift in satellite network architectures. When using GEO satellites to provide Internet access service, the performance of the Internet's Transmission Control Protocol (TCP) is degraded by the high latency and high degree of bandwidth asymmetry present in such systems. We therefore undertook a comprehensive study of TCP performance in the context of broadband satellite systems used for network access. We first studied whether TCP's congestion avoidance algorithm can be adjusted to provide better fairness when satellite connections are forced to share bottleneck links with other (shorter delay) connections. Our data suggests that adjustments of the policy used in that algorithm may yield substantial fairness benefits without compromizing utilization. We next demonstrated how minor variations in TCP implementations can have drastic performance implications when used over satellite links (such as a reduction in file transfer throughput by over half), and from our observations constructed a satellite-optimized TCP implementation using standardized options. We explored the performance of TCP for short data transfers such as Web traffic, and found that two experimental options relating to how TCP starts a connection, when used together, could reduce the user-perceived latency by a factor of two to three. However, because not all of these options are likely to be deployed on a wide scale, and because even the best satellite-optimized TCP implementation is vulnerable to the fairness problems identified above, we explored the performance benefits of splitting a TCP connection at a protocol gateway within the satellite network, and found that such an approach can allow the performance of the satellite connection to approach that of a non-satellite connection. Carrying this work one step further, we construct a satellite-optimized transport protocol that can be used in such a split-connection environment, and demonstrate how it outperforms

TCP in a satellite environment characterized by large amounts of bandwidth asymmetry. In particular, our protocol, which we have dubbed the "Satellite Transport Protocol," uses up to an order of magnitude less traffic on the bandwidth-constrained reverse channel than is needed by TCP.

In contrast to research on GEO systems, research on LEO systems is still in its infancy. While LEO systems are being designed specifically to avoid the high latencies found in GEO systems, their design is challenging from a packet routing perspective due to the highly time-varying nature of the LEO network topology. Moreover, even the most basic system properties of such constellations is not well documented in the literature. We constructed a detailed packet-level LEO network simulator and identified some fundamental delay performance results of commercially proposed constellations. We then explored whether or not geographic-based network addresses can be constructively used in the design of distributed or centralized packet routing systems. We found that the construction of a distributed packet routing algorithm based on geographic-based packet forwarding is fundamentally challenging due to subtle topological properties of LEO networks. However, we demonstrated that geographic-based addresses are useful in centralized routing systems, enabling significant reductions in routing traffic and on-board routing tables. Specifically, we constructed routing strategies based on geographic-based addresses that potentially reduce by an order of magnitude the amount of routing traffic exchanged between satellite nodes and a centralized routing center on the Earth's surface, and the number of forwarding table entries required for non-local destinations.

Broadband satellite networks are likely to become an important niche of the future Internet because of their unique ability to provide network access from almost any point on the globe, particularly those underserved by terrestrial infrastructure. However, because the design of Internet protocols is driven by the performance of the wired Internet, satellite network engineers must be vigilant in assisting in the design and deployment of satellite-friendly protocols and in considering how satellite networks interwork with the wired Internet. Broadband LEO networks are likely to be deployed later than their GEO counterparts, and the design of these networks is still in its infancy, particularly since such networks are significantly more ambitious technically than anything that has been previously attempted. It is our hope that the findings presented in this thesis may contribute to a better understanding of how to design protocols for these next-generation systems while stimulating further work in this area.

Professor Randy H. Katz
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Acknowledgements

Pursuing a doctoral degree is a long journey that one cannot make alone, and in this space I'd like to thank the many people who have contributed to my particular journey.

First, I thank my parents for all of the love, support, and encouragement they have provided me throughout my life. I am forever grateful for the sacrifices they made to give me the wonderful opportunites that I have had.

Randy Katz has been a fantastic research advisor on many levels. I have been continually impressed with his ability and dedication in leading a very large group of students towards their degrees, while also teaching and serving as department chair. Randy gave me a lot of freedom to pursue my research, while also providing wise advice and helpful discussions along the way. I feel very fortunate to have had Randy as an advisor and role model during my time at Berkeley.

I am grateful to the many members of the BARWAN research group for their help, friendship, and support. Giao Nguyen helped me a great deal in programming related to the *ns* simulator. Venkat Padmanabhan contributed device drivers used in many of my network experiments. Keith Sklower helped out tremendously by porting satellite card drivers from the Windows platform to Unix. Hari Balakrishnan wrote the initial TCP SACK implementation that I used in experiments. I also especially thank Hari, Venkat, Giao, Elan Amir, and Tina Wong for several technical discussions and reviews of draft versions of this research. For everyone else associated with BARWAN, I thank you for your contributions to my work and to a fun and challenging research project.

I am grateful to the other two professors on my committee, Steve McCanne and Dorit Hochbaum, for their interest and concern along the way and for their help in writing this dissertation. Both provided me with technical assistance, and the classes that they taught on Computer Networks and Network Flows were especially helpful to my work. I would also like to thank Jean Walrand for serving on my quals committee, and Joseph Kahn and Avideh Zakhor for sponsoring my research in my first year at Berkeley.

I thank Sally Floyd for her help with my initial TCP research by providing unpublished research results as well as several discussions on different aspects of TCP. The TCP fairness work reported herein derives from a class project that I undertook with Emile Sahouria, who in particular wrote several components of the background HTTP traffic generators used herein. Bruce Mah and Steve Gribble provided HTTP traffic traces that drove that traffic generator.

The *ns* simulator has been invaluable for my research, and I would like to thank everyone in the VINT project who has made *ns* such a powerful tool for networking research.

I thank Son Dao at HRL Laboratories for supporting my work both through sponsoring BARWAN and employing me as an intern over two summers. I enjoyed working with Son and the other members of HRL, including Yongguang Zhang, Bo Ryu, Dennis Connors, and Dante DeLucia. Hughes DirecPC also graciously agreed to host a Berkeley computer at their network uplink center that was used for some of my satellite network experiments.

I would also like to thank the many people who have provided me with unpublished technical data or assistance, including Don Hoffman and Marie-Jose Montpetit at Teledesic, and Lloyd Wood.

Finally, I thank my wife Sue, whose love, support, and dedication throughout this whole journey has helped get me through it all. From putting up with long hours to helping me celebrate

little victories along the way, I couldn't have asked for better companionship.

Tom Henderson
December, 1999

Chapter 1

Introduction

The latter half of the 1990s has seen a resurgence of interest in satellite-based data networks. Satellite communication systems have long been one of the hallmarks of advanced communications technology, with their remarkable and distinctive ability to link most of the populated areas of the Earth. Yet, until recently, the satellite communication industry had increasingly begun to look more like a dinosaur, with competition from fiber optic and terrestrial wireless networks steadily eating away at the industry's most profitable markets.

As of this writing, however, the satellite industry is poised for rapid growth, with funding in place for the deployment of technically ambitious, multi-billion dollar systems, and growing competition in both service provision and hardware manufacturing. Figure 1.1 illustrates one analyst's projection of the growing market for satellite broadband data services [137], and other analysts project that the growth in the satellite industry will outpace the growth of the entire communications market over the next ten years, as the satellite sector's market share rises from 2.3% today to 6% a decade from now [94]. What has triggered this rapid turnaround? The answer lies in the confluence of two economic and technological trends:

1. **The Internet boom** The 1990s will likely be remembered as the decade during which the Internet came of age. There is presently an incredible (and increasing) demand for faster and cheaper Internet services, and many companies are scrambling to offer these broadband services. Satellite networks provide a fast way to reach customers because they do not rely on buildout of a high-speed terrestrial network, which may take years to accomplish in many areas of the world. Moreover, with the advent of the World Wide Web [12], broadband Internet access tends to be highly asymmetric in traffic usage, with users downloading (consuming) much more information than they generate. As we shall discuss, this type of traffic pattern matches well with satellite networks, where it is much cheaper to receive data at broadband rates than to transmit at such rates.

2. **Advances in satellite technology** The rapid technological progress that has spurred the growth of the Internet has also helped to significantly advance the state-of-the-art in satellite technology. Most notably, miniaturization of electronics has allowed more and more sophisticated satellite and terminal hardware to be economically deployed. Satellites, which once were mainly repeaters in space, have much more on-board processing functions and have the capability to juggle multiple directional "spot" beams on the Earth's surface while also com-

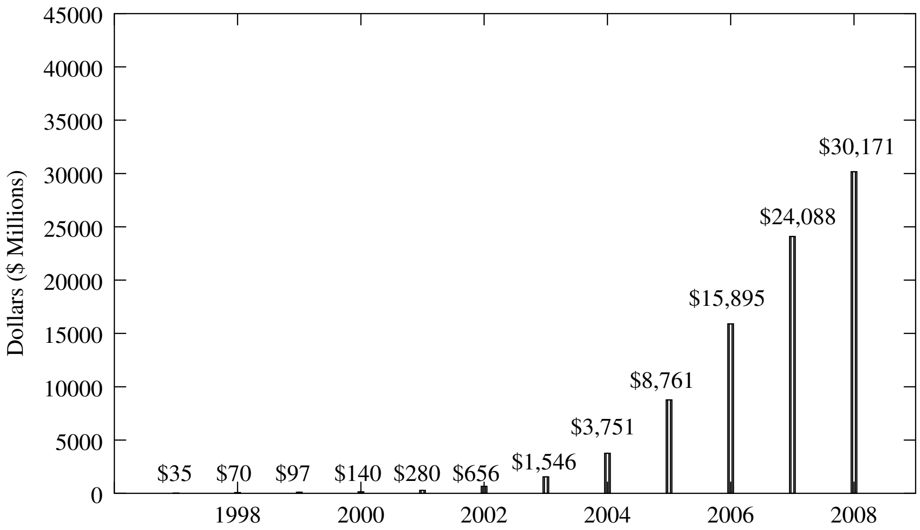


Figure 1.1: Projections of Annual Satellite Broadband Revenue (Source: Merrill Lynch and Co.).

municating with other satellites via high frequency radio links. Sophisticated constellations of low-earth-orbiting satellites, and handsets and terminals that can track the motion of these satellites, are now being designed and deployed. New frequency bands at 20-30 GHz, the use of which was once precluded by the lack of affordable high-frequency hardware, are now being opened to satellite communications, greatly increasing the available bandwidth of newer satellite systems. Probably the most widely-observable evidence of the impact of advances in electronics on this industry can be seen in the growth of direct-to-home (DTH) satellite television services, one of the most rapidly deployed consumer electronics product in history. DTH services, using small, affordable satellite dishes, have brought satellite services into the mainstream in a way that was not possible using technology of a decade ago.

While the use of satellite networks as a part of the Internet backbone dates back almost twenty five years, the use of satellites to provide high-speed network access is relatively new. The success of new satellite networks in delivering high-speed access hinges on the ability of the underlying protocols to function correctly and efficiently in the satellite environment, an environment characterized by (for traditional geostationary (GEO) satellites) much longer propagation delays than are found in terrestrial networks, and (for newer low-earth-orbiting (LEO) satellites) a rapidly time-varying network topology. In this dissertation, we concentrate on the application of satellite systems to provide this "last-mile" access connectivity, and focus in particular on two problems relevant to Internet data networking over these broadband next-generation systems: i) improving the performance of reliable transport protocols over high-latency paths, and ii) routing strategies for networks of low-earth-orbiting satellite systems. Before introducing these problems in more detail, we first digress to describe the fundamental characteristics and technological trends of both satellite communication systems and the present-day Internet.

1.1 Satellites and the Contemporary Internet

1.1.1 A Brief Overview of Satellite Communications

The field of satellite communication systems is a rich, multidisciplinary field involving several areas of electrical, aeronautical, and mechanical engineering. Several books provide overviews of the field as a whole; among the works in wide use today are those by Maral and Bosquet [81], Gordon and Morgan [50], and Pratt and Bostian [113].

The first idea of using a satellite orbiting at a geostationary altitude (35,780 km above the equator) to provide communication services is attributed to author Arthur C. Clarke in 1945. The first artificial communications satellite (SCORE in 1958) did not follow long after the Sputnik launch in 1957, and the first commercial geostationary satellite (INTELSAT 1, or "Early Bird") in 1965 ushered in the era of overseas telephony via satellite. This first INTELSAT satellite had a capacity of 480 telephone channel at an annual cost of $32500 per channel [81].

In the 1970s and 1980s, both the market for satellite communication services and the technology grew rapidly. Besides providing international telephony and data services between large earth stations owned by national carriers, communication satellites were increasingly used for video (television) distribution. The international organization INMARSAT was founded to provide telephony and data services to maritime customers. As we describe later, the first satellite network experiments based on packet switching (the Atlantic Packet Satellite Network, or SATNET) commenced in 1976. Finally, the construction of systems based on Very Small Aperture Terminals (VSATs) for transaction-oriented traffic such as credit card verification and database management was begun in the 1980s.

As mentioned above, in the 1990s the growth of alternative, cheaper technologies such as high speed fiber optic networks has gradually eliminated much of the international telephony service for non-mobile customers. However, technological advances enabled the creation of direct-to-home (DTH) satellite television services that are competitive with cable television systems. And because of the explosion of interest in the Internet, in the latter 1990s satellite channels have begun to be used for trunking between international Internet Service Providers and the US backbone.

In the coming decade, having been revitalized by the demand for broadband Internet access and broadcast television, the satellite market is poised for large growth. The two biggest technological advances are likely to be the emergence of systems at Ka-band (20-30 GHz) and systems composed of tens to hundreds of low-earth-orbiting (LEO) satellites. Ka-band systems are advantageous because they permit satellite terminals smaller than one meter in diameter to be used for two-way communications, and because the amount of spectrum allocated in this new frequency band is larger than previous allocations at lower frequencies. As we explore later in this thesis, LEO systems promise to offer services with much lower latency and terminal power requirements than those offered by geostationary satellites. Figure 1.2 illustrates the current placement of the roughly 200 commercial geostationary communications satellites in orbit around the Earth; with Ka-band systems and directional antennas that allow more than one satellite to occupy an orbital slot, the density could double in the next decade. Table 1.1 summarizes some commercially proposed LEO systems under development. In the long run, satellites are well positioned to offer broadcast services at competitive rates and to provide general communications capacity to points on the globe that are not well served by terrestrial networks. However, it is clear that there is insufficient allocated spectrum capacity for satellites to significantly displace terrestrial wireline or wireless networks, even if

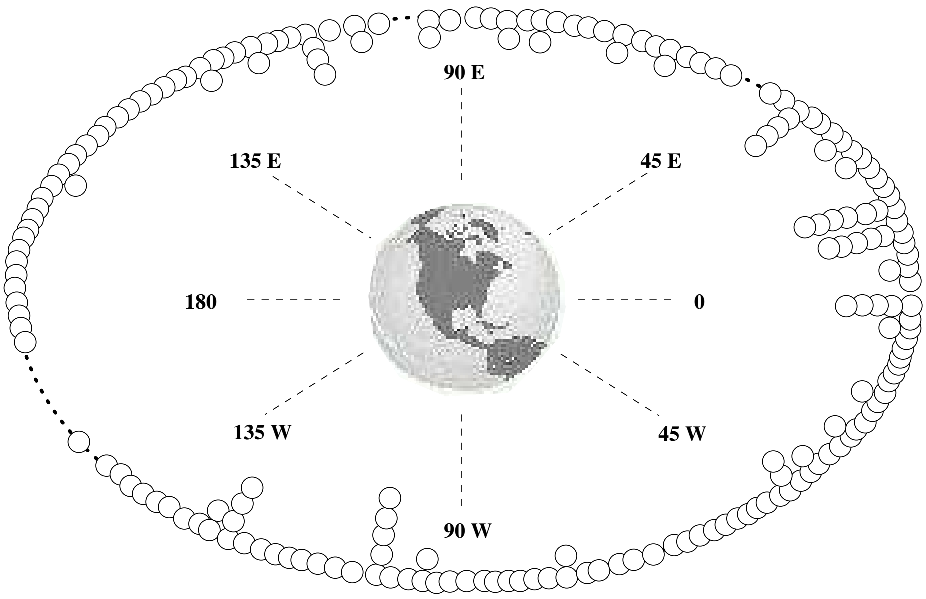


Figure 1.2: Current commercial communications satellites in geostationary orbit (Source: Hughes Space and Communications Company; reproduced with permission).

it were economical to do so [70].

1.1.2 The Internet Protocol Architecture

The term "the Internet" refers to a wide collection of packet switching networks that are tied together through the common use of the Internet Protocol (IP) and its associated routing and addressing conventions. Each network can be thought of as a separate "autonomous system" that takes responsibility for delivering traffic within its own network however it sees fit while conforming to standard protocol mechanisms at exchange points (interfaces) with other participating networks. The most distinguishing characteristic of this network architecture is that it is decentralized and has no single administrator. Another key aspect of the architecture is how the various protocols interrelate. Figure 1.3 illustrates a popular view of the Internet protocol architecture, sometimes called the "hourglass figure," which illustrates how there is one common protocol (IP) used by everyone (at network exchange points) but that protocol layers above and below the IP layer are more heterogeneous [110]. In fact, it was the need to interconnect different networks such as the original ARPAnet, SATNET, and the Mobile Radio Network in the San Francisco Bay area that propelled the usage of the Internet protocol [121]. In Figure 1.3, we have labelled five network "layers" that are commonly associated with the Internet architecture (also sometimes referred to as the "TCP/IP" architecture). In this thesis, we will explore problems involving protocols that lie at the network and transport layers.

The current Internet can be characterized by the following features:

- **Best effort packet delivery** The Internet makes no guarantees about bandwidth, latency,

	Iridium	Teledesic	Skybridge	Globalstar	ICO
Uses/services	voice, messaging fax	broadband access, private networks	broadband access, private networks	voice, messaging fax	voice, messaging position location
Data rates (Kb/s)	2.4	64,000 down 2000 up	20,000 down 2000 up	7.2	2.4
Number of satellites	66	288	80	48	10
Orbital planes	6	12	8	8	2
Altitude (km)	780	1375	1450	1410	10,400
Frequency band	L-band	Ka-band	Ku-band	L/S-band	S-band
Payload type Payload type	circuit switched	packet switched	repeater	repeater	repeater
Satellite crosslinks	yes	yes	no	no	no
System costs ($ billions)	3.7	9	4.2	2.2	2.6

Table 1.1: Summary of major LEO system proposals (data from various sources, but mainly [143]).

sequencing, or even the successful delivery of a packet. Instead, packets are routed among destinations as best as the routing infrastructure can do, and it is up to higher-layer protocols that operate end-to-end between corresponding hosts to provide whatever service guarantees are necessary (such as in-order, reliable delivery). This architectural decision has contributed to the nice scaling properties of current Internet deployment, albeit at the cost of supporting poorly those applications that require strict performance guarantees from the network.

- **Heterogeneity** Any way you look at the Internet, there is a tremendous amount of heterogeneity. The performance characteristics of end-hosts and communications links operating in the Interent vary widely. The success of the network relies on successful deployment and operation of protocol mechanisms that mitigate the problems posed by this heterogeneous environment. Two recent research projects in our research group (one on multimedia-related proxies for heterogeneous networks [7] and one on transport protocol performance over heterogeneous wireless networks [9]) focused specifically on dealing with heterogeneity in the Internet.

- **Huge installed base** Throughout the 1990s, the number of hosts on Internet has grown exponentially. As of July 1999 there were over 56 million hosts, double the number present at the start of 1998 [63]. As a result, it is increasingly hard to make protocol changes that do not interoperate cleanly with existing hosts on the network. New protocols or protocol enhancements that require changes to end hosts are not likely to be quickly adopted unless they either enable a new service not easily supported by the existing protocol base or provide a very large performance enhancement. Therefore, the onus is on the developer to elaborate a clear deployment path for any proposed enhancements and to demonstrate that the proposed

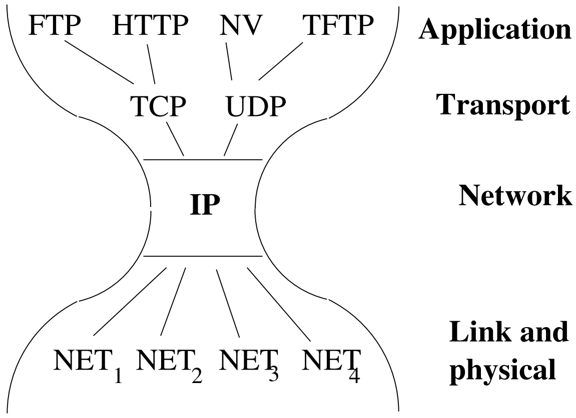


Figure 1.3: The Internet protocol architecture.

changes interoperate well with the installed base.

1.1.3 Satellites and the Internet– A Match Worth Making?

It is already well known and unavoidable that the end-to-end packet delay over GEO satellite links is commonly several hundred milliseconds worse than most high-speed Internet connections. Although future LEO systems are aimed at reducing this absolute delay, the transmission (error) performance of these systems is uncertain at this time and is likely to be worse than what users typically observe with wireline systems. Both these long latencies and (potentially) high error rates can cause performance problems for Internet connections. Often, the performance problems can be overcome by proper protocol design, but experience has proven that it is unrealistic to expect that protocol designers will take the special characteristics of a satellite link into account when designing protocols– satellites are too much of a niche market for optimization. Nevertheless, we believe that the Internet and satellite links can be a very good match for providing broadband access, for the following reasons:

- **Current Internet performance is spotty** Packet loss rates in the Internet can be quite high; it is not uncommon for packet loss rates on portions of the Internet to approach 10 to 20 percent at times due to congestion. Many current Internet users use dial-up access, which (at rates of 30 Kb/s) can add hundreds of milliseconds of delay for large packets. Additionally, Web server response times, while usually fast, can sometimes be very sluggish (i.e., server response times have heavy-tailed distributions [51]). In such an environment, satellite link performance does not look bad in comparison. In fact, for some applications such as low-bit-rate video, it is possible for satellite transmissions of a video stream to be decoded earlier than a terrestrially-routed stream if the terrestrially-based stream must overcome a higher packet loss rate. Because there is no admission control nor fine-grained traffic policing in the Internet, it is not clear whether the performance will improve markedly anytime soon.

- **Elasticity of popular applications** Geostationary satellite connections are considered problematic for "real-time" applications involving human interactions; for example, telephone calls over satellite links encounter enough latency that the timing nuances of normal conversation can be disturbed for many users. These types of applications have been termed "non-elastic" [125]. In contrast, the two most popular uses of the network– Web related activities and email– are "elastic" in that they do not require stringent performance guarantees from the network for users to be satisfied. While it is likely that delay-sensitive wide-area applications (such as IP telephony) will evolve in the near future, LEO systems are being designed specifically to better support such applications.

- **Bandwidth asymmetry matches traffic asymmetry** Satellite networks that are designed to enable low cost user terminals are typically built around a "star" network topology, in which a large hub antenna and large power amplifiers is used to broadcast a high bit rate stream to customers. However, because the most expensive component of a customer terminal is a power amplifier, a low wattage power amplifier is used and consequently the customer cannot use a large return carrier for transmissions back to the hub. Fortunately, the type of asymmetric bandwidth provided by such systems is well suited for Web browsing, by far the most popular Internet application in terms of network usage. Recent traffic traces of client PCs have shown an asymmetry ratio between inbound and outbound bandwidth usage of over 5 to 1. Satellite operators have already capitalized on this traffic asymmetry by offering hybrid satellite systems, such as the DirecPC network which allows users to download data at rates up to 400 Kb/s but which relies on a dial-up return channel (typically around 30 Kb/s) [100].

1.2 Contributions and Thesis Overview

1.2.1 Statement of Research Problems

In this thesis, we concentrate on *the application of satellite systems to provide broadband access to the Internet*. This is in contrast to using such networks for trunking or transit connectivity between carrier networks, as has often been done historically. Because point-to-point satellite connections will continue to be more expensive than fiber-based options (if such options exist), satellite communications will most likely be used i) for broadcasting of information to multiple users and ii) for broadband access where viable terrestrial infrastructure is lacking. In this application context, we focus on the following two problems:

- the performance of reliable transport protocols over GEO satellite links, and
- the design of unicast routing protocols for LEO satellite networks.

The problem of designing and deploying reliable transport protocols that perform adequately over satellite links is well established one, but in our opinion the problem has not been completely solved and in particular we approach the problem from different angles. Specifically, we consider the transport protocol performance that a satellite-based user is likely to encounter when his or her connection traverses a portion of the wired Internet, in contrast to looking at transport connections in isolated satellite environments. We explore changes that can be made in existing transport protocol implementations and specialized protocol gateways that can be deployed within satellite broadband access networks. We especially emphasize studying protocol performance in the context of other competing Internet connections that share the same path as the measured connection; such an emphasis is not often found in the available literature. And rather than focusing exclusively on file transfers, we explore the performance of small data transfers commonly associated with Web browsing.

In contrast, the design of unicast routing protocols for LEO networks is an emerging problem, with much of the previous work on networking for LEO systems focusing on connection-oriented routing rather than packet switching. This is because it has long been thought that Asynchronous Transfer Mode (ATM) networks would form the backbone of all future broadband wide-area networks. Since the future of ATM is no longer clear, and because we believe that a connectionless networking paradigm is better suited for rapidly time-varying network topologies, we instead choose to focus on IP-based routing. By constructing a network simulator that is able to provide detailed packet-level simulations of future LEO networks, we explore not only some fundamental performance properties of such constellations but also more specialized routing techniques tailored specifically for proposed LEO networks.

1.2.2 Contributions

Regarding the two main problems identified above, we were able to make a number of contributions, which we summarize here and discuss in more detail in the following chapters:

- Our study of TCP performance over GEO satellite links is among the first that explores in detail the interactions between satellite TCP connections and other (non-satellite) connections that share part of the same end-to-end path. One long-standing problem in this context has been the fairness performance of TCP's congestion avoidance algorithm when multiple connections with different round trip delays share a bottleneck link. Using simulation models, we provide evidence that, while TCP fairness problems may not be easily solvable in a manner that can be incrementally deployed by making changes to host implementations, small changes to a satellite connection's congestion avoidance algorithm can substantially improve the fairness of the bottleneck link usage without comprimizing link utilization. In particular, in congestion situations we were frequently able to double the throughput of satellite connections by making their congestion avoidance policies slightly more aggressive than normal, but not so aggressive as to unfairly penalize other competing connections. Next, we highlight how imperfect implementations of standardized TCP options relating to loss recovery and congestion avoidance can lead to poor file transfer performance over satellite links and, through experiment and simulation, construct a reference implementation of these options that can achieve good performance in a non-congested satellite environment. We also quantify how much two proposed TCP options (TCP for Transactions and options for increasing TCP's initial window) reduce the latency of short transfers, finding that the use of both options can reduce the user-perceived latency by a factor of two to three. However, since these options are not guaranteed to become widely deployed, and since the file transfer performance of even satellite-optimized connections can be derailed by the fairness problem discussed above, we investigate the performance benefits that can be achieved by using TCP protocol gateways in a satellite network. In particular, we find that well-tuned TCP protocol gateways, which split a single TCP connection into two separate connections, can achieve performance comparable to connections that do not traverse a GEO satellite link.

- Given that TCP protocol gateways are possible in the network architecture, we explore whether protocols other than TCP may be more suitable for the long-delay and bandwidth-asymmetric environment of satellite access networks. We describe the overall design and performance of a satellite-optimized transport protocol (which we have named “STP”) that is specifically designed for a broadband satellite network characterized by high degrees of bandwidth asymmetry. Some benefits of this protocol relative to TCP’s performance include good performance in a high loss environment, less sensitivity to large variations in the round trip delay experienced by packets, and a reduction of up to an order of magnitude in the amount of bandwidth used on the reverse channel to return acknowledgments.

- Finally, we conduct a detailed investigation of packet routing alternatives in the context of LEO satellite networks. Our study is believed to be among the first that focuses on the connectionless packet routing problem for the highly time-varying network topologies of LEO networks. We describe the construction of a LEO network simulator, suitable not only for packet routing studies but also for other aspects of networking over future satellite constellations. This simulator reveals some interesting fundamental delay performance properties of LEO networks that have not yet been described in the literature. After illustrating some of these properties, we turn our attention to packet routing. Although existing distributed routing protocols can be made to work in this environment, we seek to exploit the specialized topological properties and system constraints of LEO networks in the design of new routing strategies. We first explore the hypothesis, advanced by other researchers, that by making locally optimal packet forwarding decisions that minimize the geographic distance to the destination, one can obtain routes that are close to optimal in terms of delay performance. Although we find that a distributed protocol based on this concept is fundamentally difficult to construct because of the structure of commercially-proposed network topologies, we are able to demonstrate the benefits of geographic-based addresses in centralized routing systems. We develop an addressing strategy for a particular cellular structure on the Earth’s surface and prove its optimality from the standpoint of maximizing opportunities for address aggregation of geographically contiguous cells. By taking advantage of temporal consistencies in routing tables and address aggregation possibilities, we propose a centralized routing strategy that, when compred to traditional non-hierarchical routing approaches, leads to a reduction of over an order of magnitude in both the amount of routing traffic that must be conveyed between network nodes and the number of satellite routing table entries used for non-local destinations.

1.2.3 Thesis Overview

The remainder of this dissertation is organized as follows.

In the next chapter we delve deeper into background material related to the problems we address in this thesis, and survey the related work that provides the foundation for the research presented herein.

In Chapter 3 we describe our overall research methodology and provide an overview of the simulation environment that we used to generate numerical results and the experimental testbed used to study the performance working implementations of the protocols we constructed.

Chapters 4 through 6 form the core of the thesis, with Chapters 4 and 5 focusing on the problem of transport protocol performance over GEO satellite links, and Chapter 6 exploring the problem of unicast packet routing in LEO satellite constellations.

Chapter 4 is concerned with the performance of TCP in a satellite environment, focusing first on potential remedies to fairness problems inherent in TCP's congestion avoidance algorithm, then studying the interaction of different TCP implementation options in a satellite environment, and finally exploring the potential gains achievable through the use of TCP protocol gateways in satellite access networks.

In Chapter 5 we introduce our satellite transport protocol (STP). We begin by describing the basic design and operation of the protocol. We then study its performance through the use of simulation models and an implementation.

We begin Chapter 6 with a detailed discussion on the construction of a simulation environment to study the problem of networking in LEO satellite constellations. After presenting some fundamental delay performance results obtained from our simulator, we focus in the remainder of the chapter on the strategy of using geographic-based addressing information to simplify routing.

Finally, in Chapter 7 we conclude by summarizing our our work and discussing directions for future work.

Chapter 2

Background and Related Work

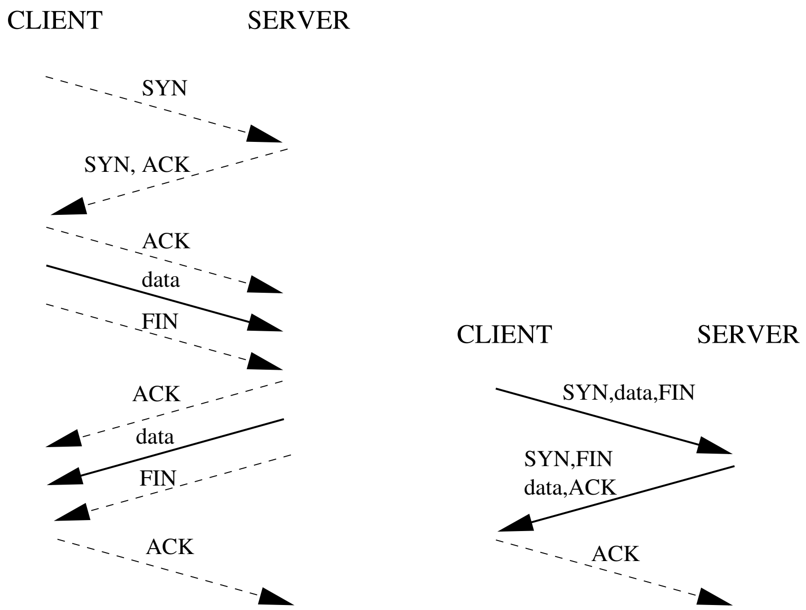
In this chapter we survey work related to our own, both to point out the many contributions of previous researchers and to place our contributions in the proper context. We organize this survey around the two main themes of our research on networking over broadband satellite systems: *reliable transport protocol performance over satellite links*, and *routing for LEO satellite networks*. In both cases, we first provide the reader with background information, followed by a discussion of previous research related to our own. We conclude by summarizing how our research builds on this previous work.

2.1 Reliable Transport Protocols in a Satellite Environment

The Internet is a *best effort* network, which means that packets are neither guaranteed to arrive at the intended destination at all, nor guaranteed to arrive at the destination in the order that they were sent. This fundamental design feature of the Internet has allowed it to scale well, because reliability is implemented at the end-hosts and not within the network. To provide applications with a guaranteed, in-order, data delivery service, a reliable transport protocol must operate over this unreliable network. Many of the most popular Internet applications, such as the Web, file transfer, electronic mail, and remote terminals, rely on end-to-end reliability between hosts. Almost all of this traffic uses one dominant transport protocol; namely, the Transmission Control Protocol (TCP). In this section, we first describe the basics of TCP operation, focusing on those aspects that are most relevant to satellite links. Next, we survey the large body of work that has aimed at improving TCP performance over satellite links and other network paths that exhibit characteristics similar to satellite links. Finally, we discuss work on other Internet-related reliable transport protocols.

2.1.1 Transmission Control Protocol (TCP) Overview

This subsection describes aspects of TCP operation relevant to the research described in this thesis. TCP was originally specified and implemented for the ARPANET in the 1970s; the original Internet RFC was written in 1981 but was derived from several earlier ARPANET specifications [111]. For a more comprehensive overview of TCP, the interested reader is directed to [127]. Over the years, a large number of reliable transport protocols have been invented, but TCP



(a) Typical data transaction using TCP (b) Data transaction using T/TCP

Figure 2.1: Typical packet sequences for TCP and T/TCP.

is currently used almost exclusively for reliable unicast transport service in the Internet. Hence, we will focus our attention primarily on TCP.

Basic TCP Operation

TCP provides a reliable, end-to-end, byte-streaming data service (with guaranteed in-order delivery) to applications. A transmitting TCP accepts data from an application in arbitrarily-sized chunks and packages it in variable-length segments for transmission in IP datagrams, with each byte of data indexed by a sequence number. The TCP receiver responds to the successful reception of data by returning an acknowledgment to the sender, and by delivering the data to the receiving application; the transmitter can use these acknowledgments to determine if any data requires retransmission. If on the sending side the connection closes normally, the sending application can be almost certain that the peer receiving application successfully received all of the data.

TCP is typically implemented in the operating system kernel, and accessed through an Applications Programming Interface (API). The most well known and used API is known as *sockets*, and it provides user-level programs with access to network services like TCP through standard system calls [127].

Connection Establishment and Release

TCP exchanges specially labelled segments to establish, release, and reset a connection. Three segments are typically required to establish a TCP session: the connection initiator (typically called the *client*) first sends a SYN segment to the connection responder (typically called the *server*), the server responds with an acknowledgment (ACK) of the SYN concatenated with its own SYN, and the client then sends back an ACK of the second SYN. To close a connection, both sides send a FIN segment to each other, and respond with an ACK of the FIN. Figure 2.1a shows a typical segment exchange.

Seven packets are usually required to transfer as little as 1 byte of data. To support small transactions, an extension known as TCP for Transactions (T/TCP) was standardized in 1994 [15]. Figure 2.1b shows how the seven packets can be reduced to three for a small exchange. Support of the T/TCP extension has been slow, however, for two main reasons. First, there are security concerns over denial of service attacks based on T/TCP (an attacker could flood a server with SYNs; in T/TCP's case, each SYN received causes the server to immediately allocate system resources even if the connection will ultimately be rejected). Second, T/TCP requires the application to use the `sendto()` or `sendmsg()` system calls when instantiating a connection; however, most applications use the `connect()` and then `send()` or `write()` system calls.

Basic Loss Recovery

The basic loss recovery mechanism for TCP is a retransmission timer located at the sending end. After a TCP sender sends data, it waits for a *timeout interval* for the receiver to ACK the data. If no ACK is received by the end of the timeout interval, the data is retransmitted and a new timer is started based on a new timeout interval. In most implementations, not every segment is timed– there is only one outstanding segment being timed at any given time. The timeout interval for a segment is based on the estimated round-trip time (RTT) of the connection, and subsequent timeout intervals for the same segment are doubled each time; this process is known as *exponential backoff* of the retransmission timer. The estimated RTT of a connection is obtained by repeatedly timing packet exchanges to obtain RTT samples and subsequently passing the samples through an exponentially weighted moving average filter to obtain a *smoothed* round trip time (srtt) estimate. The initial RTT is assumed to be very large (greater than one second) or may be obtained via a cache. The RTT measurement is usually very coarse in current implementations, and the timeout interval is also very conservative, with the base timeout interval usually set to $srtt + 4 * rttvar$, where $rttvar$ is the mean linear deviation of the RTT measurements.

Congestion Avoidance and Control

TCP has been heavily used in the Internet for over a decade, and a large part of its success is due to its ability to probe for unused network bandwidth while also backing off its sending rate upon detection of congestion in the network; this mechanism is known as “congestion avoidance” [66]. An additional mechanism known as “slow start” is used upon the start of the connection to more rapidly probe for unused bandwidth. The operation of these mechanisms is described in detail in [127], and is briefly summarized here. TCP maintains a variable known as its *congestion window*, which is initialized to a value of one segment upon connection startup. The window represents the amount of data that may be outstanding at any one time, which effectively determines the TCP sending rate. During slow start, the value of the congestion window doubles every round trip time (RTT), until either a threshold is reached (*slow start threshold*, initially set to an arbitrarily large value), or a loss is detected. All losses are interpreted as congestion events in TCP, so, using the basic loss recovery mechanism described above, upon a timeout the slow start threshold is set to the value of the congestion window, the congestion window is subsequently reset to one segment,

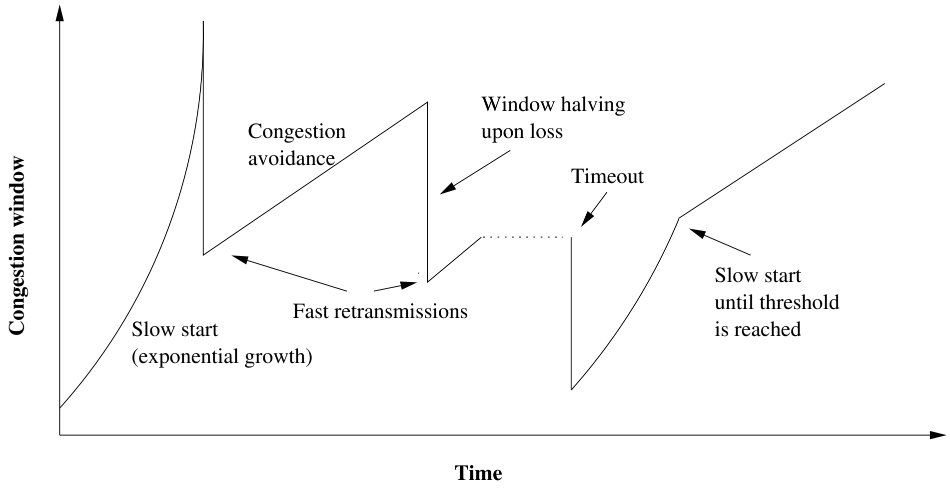


Figure 2.2: Basic operation of TCP slow start and congestion avoidance (from [9]).

and TCP begins to slow start again after retransmitting the missing segment. When the window size grows larger than the slow start threshold, TCP enters the congestion avoidance phase, where it adds approximately one segment to its window every one or two RTTs. This is a much slower, linear growth phase of the congestion window.

Slow start and congestion avoidance were introduced into TCP in the late 1980s; TCP implementations that implement slow start and congestion avoidance with the basic loss recovery mechanism described above are known as TCP "Tahoe" implementations.

Enhanced Loss Recovery and Congestion Avoidance

The basic loss recovery described above was the only loss recovery mechanism implemented in the TCP (Tahoe) releases of the late 1980s. An enhancement to TCP Tahoe was added around 1990 to form TCP "Reno." Note that in TCP Tahoe, each time a loss occurs, TCP must wait for a timeout to retransmit the missing segment. Because the timeout interval is conservative, the TCP sender ends up idling for a relatively long period of time (on the order of one to two seconds). Furthermore, the connection must reenter slow start each time a loss occurs. For satellite connections especially, this timeout period and the following slow start result in several seconds during which the throughput is very low and channel bandwidth may be wasted. TCP Reno introduced the "fast retransmit" and "fast recovery" mechanism. TCP Reno assumes that the arrival of three or more duplicate ACKs is a good indication that the segment beyond that which is being ACKed has been lost. Rather than wait for a timeout, it retransmits the segment immediately, and reduces the congestion window to half of its previous value. It then allows the TCP sender to send a new segment for each duplicate ACK received, to keep the pipe full during this recovery phase. If the retransmission is again lost, TCP must wait for a timeout. For single loss events, TCP Reno is very effective in recovering the loss without a damaging reduction of throughput. TCP Reno is described in more detail in [129]. Figure 2.2 illustrates an example of how TCP's congestion window evolves over time [9].

For all of its effectiveness at recovering from single loss events, TCP Reno has what is widely considered a bug when it comes to multiple loss events in a single window [36, 60]. The problem is that if multiple losses occur in a window of data (i.e., within the same RTT interval), TCP Reno only performs fast retransmit of the first missing segment, and often must wait for a timeout for subsequent lost segments. TCP implementations that fix this bug are known as TCP "NewReno" [42]. In Chapter 4, we will discuss the implications of this bug on satellite connections in greater detail.

TCP Reno and NewReno can only recover from one loss event every RTT. In an environment where the RTT is large, this leads to a very slow recovery for bursty loss events. TCP with Selective Acknowledgments [83], also known as TCP SACK, standardizes a new TCP option that allows the receiver to report a large number of missing segments at one time. This option is particularly beneficial in the satellite environment, as we show in Chapter 4.

Finally, an experimental TCP implementation known as "Vegas" attempts to implement a congestion avoidance mechanism that avoids losses by reducing the window upon a detection of an increase in the RTT (which would indicate queues building along the path) [16]. Unfortunately, TCP Vegas has not been shown to work well in a heterogeneous environment; in particular, in a satellite environment, it exhibits rather poor performance because it is very slow to probe for unused bandwidth [146].

2.1.2 TCP Performance over Satellite Links

Satellite networks formed a part of experimental internets beginning in the mid 1970's (in the form of the Atlantic Packet Satellite Network, or "SATNET" [65]), and TCP is reported to have worked correctly over such links, albeit at bit rates in the tens of Kb/s [123]. However, performance problems did not manifest themselves in a network where the maximum link capacity was 56 Kb/s. In this section, we summarize some of the solved and unsolved TCP performance problems in a satellite environment. Partridge and Shepard also discuss several of these causes for poor satellite TCP performance in [104].

Key Issues for Satellite Transport

The main characteristics of the end-to-end path that affect transport protocol performance are latency, bandwidth, packet loss due to congestion, and losses due to transmission errors. If part of the path includes a satellite channel, these parameters can vary substantially from those found on wired networks. In this thesis, we make the following assumptions about the performance characteristics of future systems:

- **Latency:** The three main components of latency are propagation delay, transmission delay, and queueing delay. In the broadband satellite case, the dominant portion is expected to be the propagation delay. For connections traversing GEO links, the one-way propagation delay is typically on the order of 270 ms, and may be more depending on the presence of interleavers for forward error correction. Variations in propagation delay for GEO links are usually removed by using Doppler buffers. Therefore, for connections using GEO links, the dominant addition to the end-to-end latency will be roughly 300 ms (one way) of fixed propagation delay. In the LEO case, this can be an order of magnitude less. For example, satellites at an altitude of 1000 km will contribute roughly an additional 20 ms to the one way delay for a single hop; additional satellite hops will add to the latency depending upon how far apart are the satellites. However, the delay will be more variable for LEO connections since, due to the relative motion of the LEO satellites, propagation delays will vary over time, and the connection path may change. Therefore, for LEO-based transport connections, the propagation delay will generally be smaller (such as from 20-200 ms), but there may be substantial delay variation added due to satellite motion or routing changes, and the queueing delays may be more significant [49].

- **Asymmetry:** With respect to transport protocols, a network exhibits asymmetry when the forward throughput achievable depends not only on the link characteristics and traffic levels in the forward path but also on those of the reverse path [11]. Satellite networks can be asymmetric in several ways. Some satellite networks are inherently bandwidth asymmetric, such as those based on a direct broadcast satellite (DBS) downlink and a return via a dial-up modem line. Depending on the routing, this may also be the case in future hybrid GEO/LEO systems; for example, a DBS downlink with a return link via the LEO system causes both bandwidth and latency asymmetry. For purely GEO or LEO systems, bandwidth asymmetries may exist for many users due to economic factors. For example, many proposed systems will offer users with small terminals the capability to download at tens of Mb/s but, due to uplink carrier sizing, will not allow uplinks at rates faster than several hundred Kb/s or a few Mb/s unless a larger terminal is purchased.

- **Transmission errors:** Bit error ratios (BER) using legacy equipment and many existing transponders have been poor by data communications standards; as low as 10^{-7} on average and 10^{-4} worst case. This is primarily because such existing systems were optimized for analog voice and video services. New modulation and coding techniques, along with higher powered satellites, should help to make normal bit error rates very low (such as 10^{-10}) for GEO systems. For LEO systems, multipath and shadowing may contribute to a more variable BER, but in general those systems are also expected to be engineered for "fiber-like" quality most of the time.[1]

- **Congestion:** With the use of very high frequency, high bandwidth radio or optical inter-satellite communications links, the bottleneck links in the satellite system will likely be the links between the earth and satellites. These links will be fundamentally limited by the uplink/downlink spectrum, so as a result, the internal satellite network should generally be free of heavy congestion. However, the gateways between the satellite subnetwork and the Internet could become congested more easily, particularly if admission controls were loose.

Progress in Improving TCP for Satellite Channels

Over the past decade, a number of TCP extensions have been specified which improve upon the performance of the basic protocol in such environments:

[1] With advances in error correction, links are more likely to be in one of two states: error free, or completely unavailable.

- **Window scale [67]:** TCP's protocol syntax originally only allowed for windows of 64 KB, which limited throughput in practice to roughly 400 Kb/s. The window scale option significantly increases the amount of data which can be outstanding on a connection by introducing a scaling factor to be applied to the window field. This is particularly important in the case of satellite links, which require large windows to realize their high data rates. Because of window scale, researchers have recently reported TCP throughputs over geostationary satellite links (in controlled environments with no congestion or bit errors) in excess of 100 Mb/s [25].

- **Selective Acknowledgments (SACK) [83]:** Selective acknowledgments allow for multiple losses in a transmission window to be recovered in one RTT. TCP SACK was discussed above in Section 2.1.1.

- **TCP for Transactions (T/TCP) [15]:** TCP for Transactions, among other refinements, attempts to reduce the connection handshaking latency for most connections, reducing the user-perceived latency from two RTTs to one RTT for small transactions. This reduction can be significant for short transfers over satellite channels. We introduced T/TCP above in Section 2.1.1.

- **Path MTU discovery [90]:** This option allows the TCP sender to probe the network for the largest allowable Message Transfer Unit (MTU). Using large MTUs is more efficient and helps the congestion window to open faster.

Even though some of these options have been specified for over five years, not all implementations use them today. The lack of widespread vendor support for satellite-friendly protocol options has historically been a hindrance to achieving high performance over satellite networks. Recently, to alleviate this, the Internet Engineering Task Force (IETF) has put together a document that describes the standard TCP options and configurations that improve performance over satellite channels [5].

Unresolved Problems

Despite the progress on improving TCP, there remain some vexing attributes of the protocol that impair performance over satellite links. For these problems, there are no standardized solutions, although some are currently under study:

- **Slow start "ramp up":** TCP's slow start mechanism, while opening the congestion window at an exponential rate, may still be too slow for broadband connections traversing long RTT links, resulting in low utilization. This problem is exacerbated when slow start terminates prematurely, forcing TCP into the linear window growth phase of congestion avoidance early in the connection [104]. Researchers are now considering allowing a TCP connection to use an initial congestion window of 4380 bytes (or a maximum of 4 segments) rather than one segment [4]. Transfers for file sizes under roughly four thousand bytes (many Web pages are less than this size) would then usually complete in one RTT rather than two or three. In the following, we refer to this policy as "4K slow start" (4KSS). Other researchers have investigated the potential for caching congestion information from a recently used connection to start the new connection from a larger initial window size [101],[132].

- **Link asymmetry:** The throughput of TCP over a given forward path is maximized when the reverse path has ample bandwidth and a low loss rate, because TCP relies on a steady stream of acknowledgments (ACKs) to advance its window and clock out new segments in a smooth manner. When the reverse path has limited bandwidth, the TCP acknowledgment stream becomes burstier, as ACKs are clumped together or dropped. This has three effects: i) the sending pattern becomes more bursty, ii) the growth of the congestion window (which advances based on the number of ACKs received) slows, and iii) the "fast retransmit" mechanism that avoids retransmission timeouts becomes less effective. Since TCP acknowledgments are cumulative, researchers have recently studied ways to reduce the amount of ACK traffic over the bottleneck link by "ACK congestion control" and sender algorithms that grow the window based on the amount of data acknowledged (such as the byte counting strategy studied in [2]) and that "pace out" new data transmission by using timers [11]. This has the drawback of requiring transport-layer implementation changes at both ends of the connection. An alternative approach reintroduces the original ACK stream at the other end of the bottleneck link ("ACK filtering and reconstruction") [11, 119]. This does not require changes at the TCP sender, but is more challenging to implement. Finally, if the MTU for the constrained reverse channel is small, the path MTU discovery mechanism will select the small MTU for the forward path also, reducing performance.

- **Handling of transmission errors** TCP treats all losses as a sign of congestion. If a segment is lost to a transmission error, TCP misinterprets the loss as congestion and inappropriately responds by reducing the congestion window. Unnecessary reductions of the congestion window are particularly damaging to throughput over satellite channels. Fortunately, recent advances in concatenated error control codes can make most broadband satellite channels relatively error free. Nevertheless, even mild error rates on very high speed satellite links can have a crippling effect on throughput [26].

- **Implementation details** In many implementations, applications must explicitly request large sending and receiving buffer sizes to trigger the use of window scaling options. For example, default socket buffer sizes for many TCP implementations are set to 4 KB [56]. Unfortunately, this requires users to manually configure applications and TCP implementations to support large buffer sizes; moreover, some applications and operating systems do not permit such configuration, including common Web servers [56] and Windows NT. Since larger socket buffers consume more memory, it is not likely that larger socket buffers will be turned on by default. Also, because TCP can only negotiate the use of window scaling during connection setup, unless it has cached the value of the RTT to the destination, it cannot invoke window scaling upon finding out that the connection is a long RTT connection. As we mentioned above, even if T/TCP is present in an implementation, applications based on the sockets Application Programming Interface (API) often use system calls that prevent the usage of T/TCP. Because the TCP standard is not rigorously defined or followed, different vendor implementations often have different (and buggy) behavior (see, for example, [108] and [17]). The subtle performance effects of these variations can significantly manifest themselves over satellite channels.

- **TCP fairness** Perhaps the most challenging problem is that TCP's congestion avoidance algorithm results in drastically unfair bandwidth allocations when multiple connections with different RTTs share a bottleneck link. The bias goes against long RTT connections by a factor of RTT^{α}, where $\alpha < 2$ [73]. This problem has been observed by several researchers [54, 80, 39, 40, 43, 41, 73], but a viable solution has not yet been proposed, short of modifying network routers to isolate and protect competing flows from one another [131]. Furthermore, bandwidth asymmetry exacerbates the fairness problems by shutting out certain connections for long periods [74]. In [43], the authors discuss a "constant rate" window adjustment algorithm similar to the one which we explore. They observe that RED gateways and Reno-style enhancements to TCP are insufficient to correct the bias inherent in the standard algorithm. In [39], the performance of a constant rate increase algorithm is evaluated via simulation and qualitative analysis for connections with long RTTs which traverse multiple gateways. The author explores the performance when all connections in the simulation topology employ the modified algorithm, and shows that the performance of the constant rate algorithm meets at least one accepted measure of fairness, while the performance of standard TCP clearly does not. In [41], Floyd explores the issues surrounding alternative window increase algorithms; the constant rate adjustment policy explored in Section 4.1 builds on this work. Finally, Lakshman and Madhow study the performance of TCP/IP in networks with high bandwidth-delay products [73]. The authors observe that TCP is "grossly unfair" towards connections with higher round-trip delays, and suggest that an alternate dynamic window algorithm is a high priority for future research, although they do not endorse any new algorithm.

Further Research Efforts

Research on improving TCP performance over satellite and wireless links has increased over the past five years. Three recent research efforts stand out. The first is the development of a modified version of TCP known as the *Space Communications Protocol Standards– Transport Protocol (SCPS-TP)* for the general space environment [34]. SCPS-TP proposes a new TCP option which would enable several changes to basic TCP mechanisms, including the following: distinguishing between packet loss and packet errors (to react differently to the two events), using the TCP Vegas congestion avoidance algorithms, identifying link outage events, performing header compression, and using selective negative acknowledgments. However, SCPS-TP does not advocate a particular strategy for handling asymmetric channels, although several possibilities are discussed. A more comprehensive study on the use of TCP over asymmetric channels was recently performed at Berkeley [11], although the motivation for the study was packet radio and wireless cable networks. The authors investigated several techniques for reducing the frequency of ACKs generated by the TCP receiver, by examining both network agent-based solutions that do not require host modifications and solutions involving modifications to the TCP implementation. By combining strategies from SCPS-TP and the Berkeley modifications for asymmetry, it is possible to construct a modified TCP which behaves quite similarly to the Satellite Transport Protocol that we describe in Chapter 5, although it requires implementation changes at both the sender and receiver, or receiver-side gateways. Finally, the University of Kansas has been active in experimenting with TCP performance over high-speed satellite channels available on the NASA Advanced Communications Technology Satellite (ACTS), which provides channels at up to OC-12 (622 Mb/s) rates [25]. The IETF is currently documenting ongoing satellite-related TCP research in [3].

2.1.3 Other Related Protocols

In Chapter 5, we study the design, implementation, and performance of a new transport protocol, which we call the *Satellite Transport Protocol* (STP), that is proposed as a substitute for TCP in a satellite environment. STP is an evolution of the ATM link-layer protocol known as the *Service Specific Connection Oriented Protocol (SSCOP)* [64]. SSCOP itself was primarily a synthesis between two research efforts in the 1980s. Researchers at AT&T developed the "SNR" protocol for high bandwidth-delay product networks [97]; the protocol is named after its inventors Sabnani, Netravali, and Roome. In parallel, COMSAT Laboratories was working on selective-repeat strategies for satellite networks [88, 27]. Standardization proposals based on these efforts were combined to form SSCOP. Timer-driven acknowledgment mechanisms similar to those in SSCOP date back to 1984 [32]. The error performance of SSCOP was studied in [58], while the performance of SNR was examined in [78] and [33]. Finally, similar protocol design principles have been incorporated into wireless link layer protocols (e.g., [96]).

Over the past twenty years, a number of transport protocols have been designed for different networking environments– a survey of many of these protocols can be found in [31]. Perhaps the two most notable satellite-oriented transport protocols that have been developed are the NETBLT protocol [30], developed by Clark, Lambert, and Zhang in the 1980s, and the Xpress Transport Protocol (XTP) version 4.0 [145].

XTP is a very flexible transport protocol designed for applications ranging from real-time embedded systems to multimedia distributions to applications distributed over a wide area [145]. XTP can support these many diverse environments because it exposes a lot of policy decisions to the applications through an API much richer than the standard sockets API. For example, the XTP API allows the application to configure multicast group management, priority schemes, error control options, flow control options, the rate at which data is acknowledged, etc. The fundamental difference between STP and XTP is that XTP provides more services and exposes a lot more policy to the application via an enhanced API, whereas STP provides only one service– a reliable, byte-streaming data service. As a result, STP is specifically optimized for bit efficiency and low latency in the satellite environment, and does not require changes to the sockets API or the applications (which must intelligently configure connection parameters when using XTP). XTP incorporates several protocol mechanisms chosen for STP, including rate and burst control, efficient transaction performance, selective negative acknowledgments, unsolicited requests for retransmissions, and a polling mechanism to solicit acknowledgments.

NETBLT was specifically designed for bulk data transfer over a wide variety of networks, including those with satellite channels. Configured to run over IP, NETBLT differs from TCP in that data transfer is flow controlled via (non-adaptive) rate control rather than window control, and the parameters of rate control are negotiated during connection setup and periodically throughout the connection (although using rate control as part of congestion control is not specified). Also, in NETBLT the data is arranged in large fixed block sizes called "buffers," rather than being treated as a byte stream. Applications are aware of these data boundaries and pass contiguous buffers to the transfer protocol. The STP protocol we describe in Chapter 5 resembles NETBLT in its use of selective acknowledgments, and in its support of rate control to supplement window control.

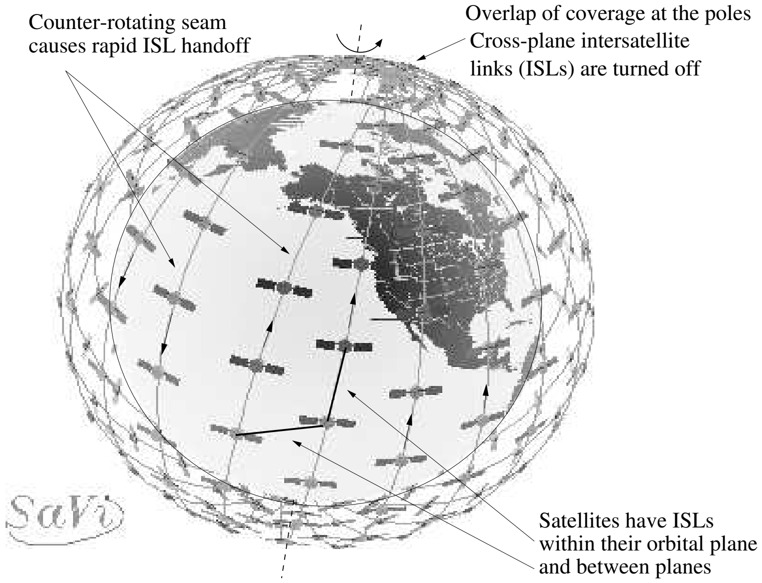


Figure 2.3: Example of a polar-orbiting satellite constellation. The figure (and subsequent SaVi-labelled figures) was generated using the SaVi software developed by the Geometry Center at the University of Minnesota.

2.2 Packet Routing for LEO Networks

In this section, we describe the fundamental characteristics of LEO networks that are relevant to the packet routing problem we study in Chapter 6. We also summarize prior work that is relevant to our research. For overviews of other aspects of LEO systems, the interested reader is directed to [105, 68]. Our emphasis herein is on emphasizing those features that are salient to the packet routing problem and discussing their impact on the design.

2.2.1 Network Characteristics of LEO Constellations

Constellation Design

Most commercially-proposed LEO constellation designs place the satellites in a number of near-polar orbital planes, in which the satellites are uniformly distributed in near-circular orbits around the plane, and in which the planes are roughly evenly spaced around the globe (Figure 2.3). The closer the orbital inclination tends towards a purely polar orbit, the more difficult it is to launch the satellites– purely polar orbits are considered too difficult to launch. Although this design has a concentration of coverage at the poles, it has been found to be a superior design for total Earth coverage with a modest number of satellites [13], and it has the advantage that most of the intersatellite communications links are not rapidly time-varying. In general, a user anywhere on the

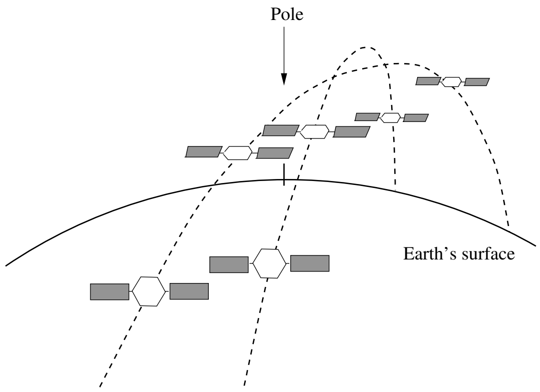
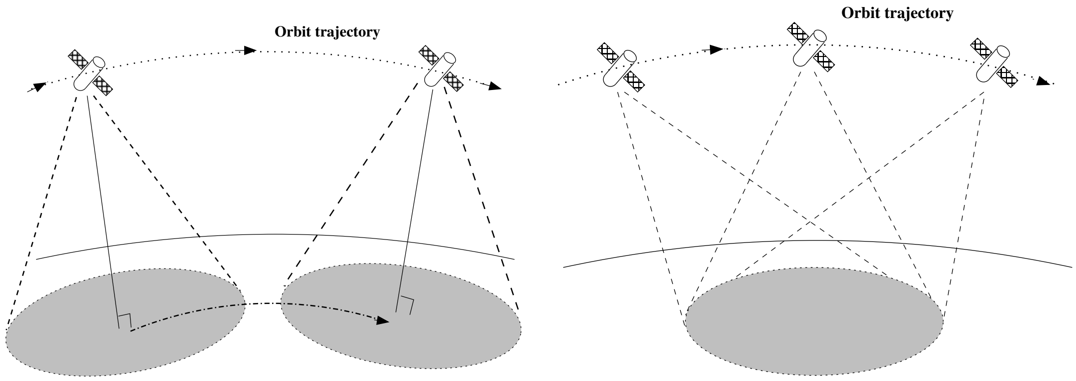


Figure 2.4: Illustration of how orbital planes intersect near the poles. A minimal amount of orbital eccentricity guarantees that one orbit passes over the other and no collisions occur.

Earth’s surface should be able to view (above a certain elevation mask[2]) at least one satellite at any time. We assume that, in general, more than one satellite may be above the elevation mask. Each satellite is equipped with an antenna system capable of directed coverage of portions of the Earth’s surface. To obtain higher system capacity, the antenna system incorporates frequency reuse via decomposition of the coverage area into a number of smaller spot beams (i.e., cells). At an altitude on the order of one thousand kilometers, the satellites orbit the Earth roughly every two hours, so that continuous coverage requires link handoff between terminals and satellites. The footprint track of the satellites also has an east-west component as well as the north-south component, since as the satellites orbit in their fixed plane, the Earth rotates beneath them.

The fact that there are multiple satellites above a given terminal’s elevation mask does not necessarily imply that a given terminal can communicate with more than one satellite. To communicate with a satellite, the terminal must lie within the radiation pattern of that satellite’s directional antenna. Since power management is a concern in LEO systems, especially on the dark side of the Earth, satellite systems such as Iridium deactivate redundant antenna beams to reduce power [62, 49]. However, by providing coverage to an area from more than one satellite, the system availability can be increased in several ways. First, the system can compensate for shadowing by terrain and buildings by offering alternative satellites. Second, during daylight hours, if the satellite is located along the same line of sight as the sun, communication will be impossible for a period of time even if the satellite is high in the sky. This is known as a *sun outage* and occurs also in GEO systems, although only for a few minutes each day around the spring and fall equinoxes when the sun crosses the Earth’s equatorial plane. Third, increased bandwidth can be provided to a particular geographic area (for example, an area with a lot of users) by using spot beams from neighboring satellites.

[2]The *elevation mask* is the minimum elevation angle of the satellite (with respect to the tangent to the Earth’s surface at the terminal’s location) above which communications are considered to be possible.



(a) Nadir-pointing footprint (b) Fixed-earth footprint

Figure 2.5: Satellite-fixed (nadir-pointing) vs. Earth-fixed footprints.

Intersatellite Links

The satellites are connected via a network of intersatellite links (ISLs). Typically, a given satellite will have ISLs to between four and eight of its nearest neighbors– payload constraints will likely prohibit the use of more than eight ISLs. ISLs are projected to be high-capacity HF or optical links– therefore, in this type of system, the bottleneck links will be the ground-to-satellite links (GSLs), due to the limited RF spectrum available for such links. There are three types of ISLs. *Intraplane* ISLs, which connect a satellite to two or four of its nearest neigbors within the same plane, can be treated as fixed links in the topology. *Interplane* ISLs, which connect a satellite to its nearest neighbors in adjacent, co-rotating planes, are variable links for a number of reasons. First, the distance between satellite planes changes as a function of latitude. Second, phasing may not be maintained between the planes, causing the satellites of different planes to slowly drift with respect to one another. Third, the interplane ISLs are switched off in the vicinity of the poles because the antenna pointing mechanism cannot track the rapidly changing angle between the satellites fast enough [49, 140]. Finally, note that in a polar constellation (Figure 2.3), there are two regions in which the planes are counter-rotating, thereby forming a "seam" in the topology. *Cross-seam* ISLs are a special case of interplane ISLs. Cross-seam ISLs, if they exist, are rapidly handed off to the next satellite. If cross-seam ISLs do not exist, communication between two locations on opposite sides of the seam must be routed over a pole. The Iridium system does not support cross-seam ISLs, while Teledesic plans to support them. Keller and Salzwedel have analyzed the problem of cross-seam ISLs in the Iridium system and have concluded that no special antenna steering requirements are necessary [71], although link acquisition may be challenging. In our research, we have assumed that cross-seam links can always be acquired. For example, Teledesic plans to use two interplane ISLs per satellite, but at the seam, only one ISL will be active; the other will be used to acquire the next (cross-seam) satellite [61].

Handoffs

In LEO systems, each satellite covers a portion of the earth's surface with a radiation pattern, or footprint. Each satellite's footprint is typically divided into a number of equal-sized cells, and a phased array antenna on-board the satellite periodically illuminates each cell and then "hops" to another one, creating a "hopping beam" (or scanning beam) schedule over time. The purpose of small cells and electronically-steered hopping beams is threefold. First, if the radiated power is concentrated on a small area, the link budget improves and terminals can use smaller antennas or power. Second, as in cellular systems, system capacity can be increased via frequency reuse. Finally, by varying the hopping dwell intervals, varying amounts of capacity can be allocated to the different cells.

Since the satellites move with respect to the Earth's surface, connections between a terminal and a satellite must be handed over to another satellite when the current satellite drops too low above the horizon. For example, the view time for an Iridium satellite is roughly ten minutes. Therefore, each system must have a technique for controlling handoffs of active communications sessions. There are two general techniques available, depending on the capabilities of the satellite antenna system. The first technique, *asynchronous* handoff, is appropriate for satellite antenna systems that have a nadir[3] pointing footprint. As the satellite moves across the sky, its footprint sweeps across the surface with a constant velocity (on the order of 5-10 km/s), as shown in Figure 2.5a. When a terminal reaches the edge of the current (leading) footprint, it is handed off to a new satellite whose (trailing) footprint is entering the area. This is the technique used in the Iridium system, and the handoff process is managed by a central control station that monitors each terminal to detect when it nears a coverage boundary [62]. At any point in time, some fraction of the terminals will be near a coverage boundary, so the system must be continually involved in handing off terminals.

An alternative handoff approach has been proposed by Restrepo and Maral [118]. If the satellite system is capable of electronically steering its beam so that it compensates for its motion, the satellite footprint can be fixed for a small interval. As shown in Figure 2.5b, this leads to "Earth-fixed cells" on the ground. After some time, all of the satellites will be moving away from their respective footprints; the system can then periodically reassign each satellite to a new fixed footprint on the ground. With this approach, the handoffs are *synchronous* since all handoffs occur when the network reorganizes, and the topology will remain static for on the order of tens of seconds to a few minutes. Note that if the system period, which is defined as the least common multiple of the satellite orbital period and the Earth's rotation period, is small, the constellation configuration can be thought of as evolving through a small set of discrete states. Although some authors [112, 24] emphasize the importance of a small system period, we assume that the network connectivity between the ground terminals and the satellite network will never be cyclic, so its influence on routing is less significant.[4]

Summary of Constellation Parameters

Table 2.2.1 summarizes key properties of the (proposed) Teledesic and Iridium constellations. Of these values, we will show in Section 6.3 that the seam separation, the elevation mask, and the presence of cross-seam ISLs have important implications on the routing and delay performance of the system, while the other orbital parameters listed are of importance mainly in that they influence the three key parameters listed above. Parameters for Iridium were drawn from [105], and those for Teledesic from [18, 92].

[3] The nadir point is the point on the Earth's surface that is intersected by a line between the satellite and the center of the Earth. It is also sometimes referred to as the subsatellite point.

[4] The Teledesic and Iridium systems do not have a small system period due to their choice of altitude.

	Iridium	Teledesic
Altitude	780 km	1375 km
Planes	6	12
Satellites per plane	11	24
Orbit inclination (deg)	86.4	84.7
Interplane separation (deg)	31.6	15
Seam separation (deg)	22	15
Elevation mask (deg)	8.2	40
Max. ISLs per satellite	4	8
Cross-seam ISLs	no	yes

Table 2.1: Parameters for the Iridium and Teledesic systems. Both systems are examples of polar orbiting constellations.

2.2.2 Routing in LEO Satellite Networks

Much of the previous work on LEO networks focuses on communications at the physical (transmission system design) and link (multiple access) layers, and on constellation design, but there is some previous work in the area of routing. We first provide an overview of some of the more comprehensive works on LEO networks in general. Next, we focus on works that concentrate on various aspects of routing in LEO networks.

LEO Systems

The following works provide good overviews of many aspects of LEO satellite systems without focusing on routing per se. Two books have recently been published on LEO communication systems. Pattan describes orbital mechanics, constellation design, multiple access, frequency issues, and antenna subsystems in [105]. Jamalipour focuses on two key issues: the implications of the projected non-uniform traffic density around the globe, and an analysis of spread spectrum multiple access [68].

Maral's tutorial paper on LEO satellite systems is a very good overview of the state of the art circa 1990 [82]. The paper discusses orbital configurations, basic routing issues, multiple access, and link analyses. Wood's Master's thesis is one of the most comprehensive discussions of LEO graph topology issues and tradeoffs in constellation design [142]. Gavish and Kalvenes have studied the relationship between satellite altitude and LEO delay performance, system capacity, and power system design [49]. They find that the altitude of satellites can be a critical design parameter depending on the various constraints of the systems. Finally, Werner describes LEO topology design issues, constraints on intersatellite links, and capacity and traffic engineering aspects, and presents a formal model for the analysis of network connectivity requirements [140].

Aside from the references listed immediately above, a number of papers focus primarily on constellation design. The work by Adams and Rider is often credited as the basis for the Iridium constellation [1]. The paper by Beste analyzes the design of satellite constellations to provide different levels of continuous, redundant coverage [13]. One of the earliest simulation models of

LEO networks, used to evaluate different constellation designs, is described in [29].

Research literature on proposed commercial LEO systems is difficult to find. There have been a number of high-level papers on the Iridium system– the papers by Grubb and Brunt are probably the most accessible and representative of the group [52, 22]. Hubbel contrasts the Iridium and (cellular) AMPS systems with respect to signaling and handoffs, providing useful details about how Iridium handoffs work [62]. Fossa has studied the performance of Iridium in the event of the loss of several satellites [45]. It is perhaps worth noting here that other similar works have focused on the military survivability of LEO networks, including [23, 14].

The Teledesic system is an outgrowth of an original system proposal called Calling [134], which seems to have been developed independently and concurrently with Iridium in the late 1980s. Although Calling was oriented towards telephony services, Teledesic has evolved into a broadband system based on packet switching. There is very little publicly available literature on Teledesic. The paper by Sturza describes various system design issues, while the presentation of Braun describes Teledesic in the context of extending the reach of the Internet through the system [130, 18]. Three patents assigned to Teledesic reveal possible aspects of the design. The first, described in more detail in subsequent chapters, is an Earth-fixed cell system for satellite spotbeam management [107]. This Earth-fixed cell approach to satellite handoffs is also described in a paper by Restrepo and Maral [118]. The second patent describes a satellite-based fast packet switch [106]. The third, which we will describe in more detail shortly, describes a possible routing architecture for the Teledesic system [77].

Network Routing in LEO Systems

The network routing problem in LEO systems encompasses the overall service strategy (connection-oriented or connectionless), the routing strategy (centralized or distributed), the actual protocols or algorithms used to manage the dynamic nature of the network, and the satellite handover strategy used by terminals and satellites. In this subsection we provide an overview of prior work that specifically relates to one or more of these issues.

A number of papers have examined issues related to virtual connection routing in connection-oriented LEO networks. One difficulty with connection-oriented routing arises when the communications session outlasts the visibility period of the initial and final satellites of the end-to-end path– the connection must necessarily be handed over to successor satellites. Uzunalioglu developed an algorithm for rerouting existing connections that balances the competing concerns of complete rerouting (for optimality) versus a simpler route augmentation to a portion of the existing path [135]. Uzunalioglu has also considered a technique called the Probabilistic Routing Protocol that can be used to select initial routes through the satellite network that have a low probability of requiring a connection reroute [136]. In [138], Werner proposed subdividing the time-varying LEO topology into intervals (states) of static topology, enumerating all of the possible virtual circuit combinations, and then picking a path that minimizes delay jitter by selecting a path across a series of states according to some optimization technique. Similar results by the same author are also reported in [139]. Most recently, Papapetrou et al. have considered the delay performance of LEO satellite constellations under self-similar and Poisson traffic by studying a simulation of Motorola's proposed Celestri system [103]. The authors consider using Djikstra's algorithm to determine appropriate virtual circuit routes through the network, and present results on the implications of self-similar traffic loads on LEO satellite topologies (namely, confirming that self-similar traffic is more bursty than

Poisson traffic and hence requires larger packet queues).

In another work applied initially to connection-oriented routing but also relevant to packet switching, Chang has proposed modeling a LEO system as a Finite State Automaton by dividing the system period [5] into fixed length intervals, during which the system is assumed to have a fixed topology [24]. A "visibility matrix" of potential satellite-to-satellite interconnections is computed for each state, and an optimum ISL assignment is computed for the state to best make use of the limited number of ISLs per satellite (i.e., the ISL topology is not fixed but is dynamic). The topology problem is solved jointly with an optimal routing problem that is based on the offered load. The optimal routing tables and link assignments are then uploaded to the satellites. The paper assumes that the topology is very regular and hence the system period and the number of states is small, which, as we show later in this thesis, is not the case in commercially-proposed networks. Another work that deals with optimizing the ISL topology so as to maximize network connectivity is [98]. The papers by Werner described above also model LEO topologies as a evolving through a finite set of states with fixed topology, and the paper by Papapetrou et al. [103] also capitalizes on the concept of a system period.

As mentioned above, both Patterson and Restrepo and Maral have proposed a cellular geometry for use in an Earth-fixed cell system [107, 118]. In such a system, the satellites continuously train their antennas onto a fixed footprint for a period of time, and then synchronously switch over the the next footprint. This technique has the potential to simplify the handovers, and consequently the packet routing problem, significantly. However, such a system comes at the cost of a degradation in the elevation mask used in the system, which has not been analyzed.

In one of the earliest works on packet switching via low earth orbit satellites, Brayer studied the packet routing problem with an emphasis on survivable and distributed algorithms [19]. In the proposed system, designed for a doubly-connected concentric ring topology, the routing is completely distributed, relying on shortest path routing if a route is known to exist and random routing otherwise. Nodes adaptively learn about routes to destinations by observing a path record coded into the header. Communications between rings (orbital planes) is not discussed in the paper.

Mauger and Rosenberg introduce the concept of defining a logical, virtual topology of cells on the ground, and performing routing of the packets with reference to the fixed virtual model [84]. Satellites that move above a given region become the embodiment of the virtual node. By providing a fixed virtual topology and by using virtual connections obtained through a restricted set of routing plans, the satellite network can provide quality-of-service guarantees. The authors recognize that there may be discrepancies between the virtual model and the actual interconnection of terminals to satellite links (since terminal handover may be performed independently of reassignment of satellites to virtual nodes), and compensate for this by proposing that ownership of cells is broadcast to all adjacent nodes so that routing to the final satellite in the path can be accomplished. The paper does not provide any quantitative analysis of this approach.

The paper by Shacham is one of the earliest works on multi-satelite networks to discuss many of the issues studied in this dissertation; namely, distributed routing protocols and addressing, as well as topology control and transport protocols [124]. Shacham advocates link-state routing that utilizes the predictability of topology changes and computation of multiple paths between nodes, as well as quality-of-service routing. The paper also discusses addressing, and is the first to propose basing addresses on the geographical locations of the terminals. The paper does not present any

[5] The *system period* is the least common multiple of the orbit period and the Earth's rotation period.

quantitative evaluations of any of the proposals, however. Hashimoto and Sarikaya also suggest using geographic information embedded in addresses to perform distributed packet routing [55]. However, they do not validate the correctness of their proposed algorithm. In a later chapter, we describe our attempts at implementing such a routing protocol in a commercially-proposed constellation such as Teledesic or Iridium.

Not much has been published on the technical details of the routing used by the Iridium or (proposed) Teledesic systems, but we have discovered three patents specifically related to routing in LEO constellations that have been assigned to the companies.[6] For Iridium, Rahnema describes a strategy for building routing tables so as to distribute as much as possible the load across various links while meeting certain route delay criteria [117]. Briefly, given some knowledge about the traffic demand between source-destination pairs and a set of candidate routes between those pairs that meets some delay criteria, an algorithm is described that iteratively selects among the candidate routes the route that results in the most uniform distribution of traffic load among the links considered so far. The patent does not discuss how to determine the order in which to consider source-destination pairs so as to achieve an optimal solution over all possible orderings, nor does it consider ground-to-satellite links in the topology. Related to this is an earlier patent by Rahnema that describes how alternate routes may be selected at random to balance load (while not discussing how to prevent routing loops from occuring in such a system) [116]. Finally, a recent patent by Liron describes in great detail how an algorithm very similar to link-state routing may be applied to the proposed Teledesic constellation [77]. Again, the patent does not discuss how the network tracks and accounts for the time-varying interconnection of terminals to the satellite mesh.

Finally, another widely studied class of multi-hop networks with rapidly changing topology is mobile ad-hoc networks. Two general classes of routing protocols exist for such networks– *proactive* and *reactive*. Proactive protocols continually update routing information so that, when a packet needs to be forwarded, routing information is already in place. The Wireless Routing Protocol, based on the class of distance-vector protocols, is a good example [93]. Reactive protocols instead invoke a route discovery procedure as needed. The Zone Routing Protocol is an example of a hybrid between proactive and reactive schemes, maintaining precise routing information within a certain radius, and querying for routes on demand for locations outside of the radius [53]. Although certain similarities apply, the routing problem for broadband LEO networks is different in that i) reactive protocols are likely to incur too much latency in a LEO environment, ii) most topology changes are predictable in a LEO network, and iii) the network graph structure in a LEO network is much more regular.

2.3 Summary of Related Work

The related work described herein lays the foundation for out contributions. The performance of transport protocols over satellite links has been well studied, but the problems are recognized by the research community to be hard problems and not yet completely solved. We focused our research effort on one particular aspect of TCP performance: the interaction between TCP algorithms and congestion-induced losses along an end-to-end path that contains both satellite channels and terrestrial network segments, in which the satellite channel potentially has bandwidth

[6] The assignment of patents to a company does not necessarily imply that the company will ultimately make use of the inventions.

asymmetry. Most previous satellite TCP work has only looked at the satellite channel in isolation, and often under the assumption that the satellite channel had a high bit error rate. We have assumed an environment more in line with current systems, for which the satellite bit error rate is very low, the bandwidth available in one direction differs drastically from that of the other direction, and for which a connection contains both terrestrial and satellite portions. We have been able to make contributions in the areas of specifying satellite-friendly implementation of standard TCP congestion control and loss recovery algorithms, and have also demonstrated improvements in performance due to non-standard protocol changes as discussed in Chapter 4. Furthermore, the issue of transport protocol performance over asymmetric satellite channels was not well treated by the previous literature, which led to the development of our Satellite Transport Protocol described in Chapter 5.

For routing in LEO networks, much of the previous work has focused on connection-oriented routing and handoff techniques, and the prior research on packet routing in LEO systems did not take commercially-proposed constellation designs into consideration (and therefore often oversimplified the design problem). In this thesis, we have concentrated our research efforts on packet routing, because of our belief that the future Applications Programming Interface (API) for data will continue to be based on IP, and because the IP service model (which permits packet reordering and does not have unduly strict quality-of-service requirements) fits well with the time-varying topologies found in LEO networks. As we explain in later chapters, our satellite routing research initially started as an effort to explore some techniques proposed by the work summarized above (such as performing distributed routing via geographic-based addresses), but evolved into an exploration of routing problems not previously considered in the literature.

Chapter 3

Methodology

We describe in this chapter our basic research methodology and key elements of our research infrastructure. We have used a combination of analysis, simulation, and experiments with real networks and protocol implementations to perform the research reported herein, and in the first section we describe our overall research strategy. Next, we describe the simulation environment used for our simulation studies and summarize the key extensions we have added. Finally, we describe elements of the Bay Area Research Wireless Access Network (BARWAN), which we used for our experimental work. We defer detailed descriptions of our measurement techniques and performance metrics to the later chapters.

3.1 Research Strategy

Our research strategy is depicted in Figure 3.1.[1] The figure indicates that we iterate cycles of analysis, simulation, and experimentation to converge on an effective solution to the research problems. The first phase of work was the definition of the problem and identification of performance bottlenecks. We started with two general problem areas critical to networking over next-generation broadband satellite systems: addressing the poor performance of the TCP protocol in a heterogeneous end-to-end environment that includes satellite channels (*satellite transport protocols*), and designing a core packet routing strategy for Low-Earth-Orbiting (LEO) satellite networks (*satellite routing*). In the case of satellite transport protocols, we first examined existing work in the field and used simulation and experiments with standard TCP implementations to uncover the causes of poor TCP performance over satellite channels. In this phase, we also relied on experimental results to validate our simulator, since we already had a working reference implementation. For satellite routing, we did not have access to any existing simulation tools or working systems, so our work in this phase was confined to examining the existing research literature.

Once we had a good idea of what the research challenges were, the second phase of work involved exploration of the design space and evaluation of potential solutions. Again, Figure 3.1 is a good illustration of the approach. In the case of satellite transport protocols, we first used the results of our benchmark performance results to analyze the problems and to formulate candidate solutions. Next, we used simulation to evaluate many of these solutions. In this phase, simulation was an easier

[1]This strategy was commonly used and cited by members of the BARWAN research team.

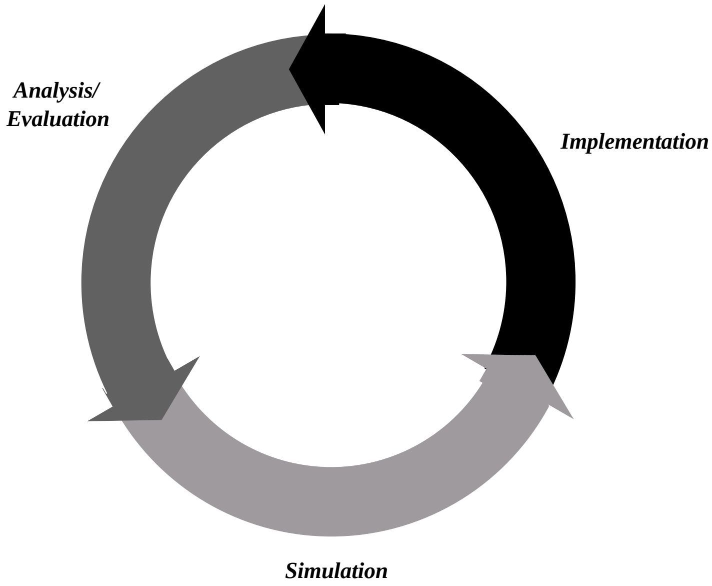


Figure 3.1: Three-phase research methodology– analysis, simulation, and implementation.

and more flexible approach to explore the design space than direct implementation was, because simulation is a controlled environment in which certain aspects of the design can be isolated and compared on an even basis. We also constructed a LEO network simulator by adding extensions to the *ns* simulator described below. Our extensions first used the existing routing infrastructure of *ns*; later, we improved the speed of simulation by optimizing the routing code for our network models. We then explored possible changes to satellite routing by constructing our own routing protocols and simulating their performance.

The third phase of the research consisted of iterative cycles of the second phase, refining our solutions as necessary until we were satisfied with the demonstrated improvements. As depicted in Figure 3.1, our simulation and implementation results led to further analysis and validation of improved solutions. The results of our research described in subsequent chapters were often the result of several iterations of the process.

3.2 Simulation Environment

Simulation is a particularly useful tool for networking research. First, it facilitates easy implementation of new algorithms and policies, allowing more rapid evaluation of a design space. Simulation can help to identify promising solutions which can often then be more carefully verified in an implementation. Second, a simulated environment is a controlled environment. Because of this, one can construct simulations that isolate the effects of certain parameters and algorithms on the overall performance. Also, evaluation of aggregate network performance is made easier because all network elements are made available through one interface. This is particularly important when studying the impact of an algorithm or policy on many nodes in a wide-area network, for example.

Third, in some cases, building an experimental implementation (such as a LEO satellite network or other large scale systems) is infeasible.

We performed most of our simulation studies using the UCB/LBNL network simulator known as *ns*, now widely used as part of the VINT project [8]. *ns* is a event-driven simulator originally derived from the REAL network simulator [72]. The simulator has an object-oriented architecture, and simulation objects are typically implemented as *split objects*: partly in C++, and partly in MIT's Object Tcl (OTcl) [141]. Such objects exist simultaneously in both language realms, and functionality can typically be added in either language (generally, functionality that requires per-packet processing is best implemented in C++, while more infrequently processed code is more flexibly implemented in OTcl). The state between the split implementation is made consistent through the use of *bound* instance variables, in which any changes to such variables in one language are immediately visible in the other. *ns* is a particularly strong choice for TCP research, since many TCP variants (Tahoe, Reno, NewReno, Vegas, etc.) are standard parts of the simulator. *ns* has also been used extensively for multicast routing and transport protocol research. Until recently, *ns* did not focus on providing detailed simulations of the link and physical layers, but UCB's BARWAN and CMU's Monarch research groups have contributed support for Local Area Networks (LANs), wireless channel error models, and wireless ad-hoc routing protocols. [20, 69].

Although we defer some details of our simulation extensions to later chapters, we briefly describe three enhancements we made to *ns*: (i) HTTP traffic generator, (ii) implementation of the Satellite Transport Protocol (STP), and (iii) LEO satellite network extensions.

3.2.1 HTTP Traffic Generator

TCP performance is well-known to be highly sensitive to the presence of other traffic in its path. In particular, the timing of packet losses due to congestion can cause the throughput to vary dramatically. When simulating TCP, it is necessary to load the foreground communications path with a realistic model of background traffic, so that performance can be accurately assessed in a realistic environment. We implemented, along with Emile Sahouria, an HTTP traffic generator for *ns*. This traffic generator was used to provide background Web-like traffic for both TCP and STP simulations, as described in the following chapters.

The HTTP traffic generator works as follows. Client and server traffic sources emulate the request and response traffic processes from typical Web browsers and servers. The client object first initiates a variable length request; after a random processing time, the server responds with a random number of connections of varying length. After a random viewing time (referred to as "think time"), the client then issues another request. Empirical distributions dictate all of the above random quantities. These distributions have been derived from traces of HTTP traffic taken by Bruce Mah on local area networks at the University of California, Berkeley, during the 1995-96 timeframe [79].

3.2.2 Satellite Transport Protocol

We implemented the data transfer mechanisms of the Satellite Transport Protocol (STP) in *ns* to test the large file transfer performance of the protocol. This simulation model then was used as a basis of the STP kernel implementation. We did not perform simulations of short-lived TCP or STP connections; instead, we relied on analysis and experiments with the actual implementations.

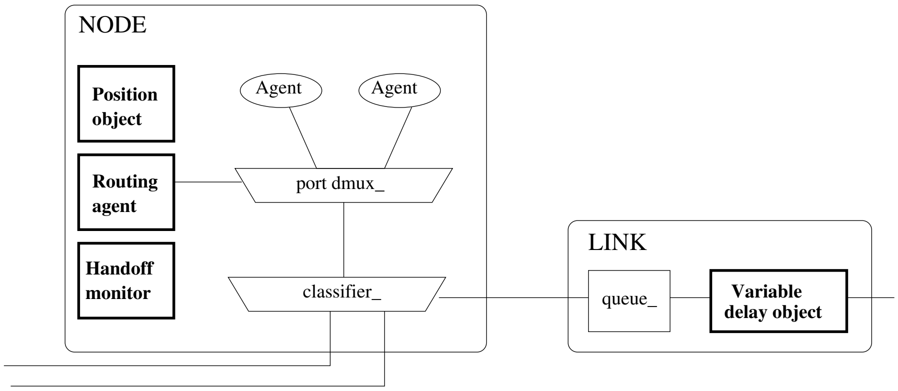


Figure 3.2: Extensions to the ns simulator (new elements listed in bold type).

3.2.3 LEO Satellite Network Extensions

As mentioned above, we selected *ns* as the basis for our simulation experiments because of its extensive support of Internet routing and transport protocols, and because our research group has been active in the ongoing development of the simulator. However, *ns* was not initially designed to support terminal mobility or dynamic topologies. Consequently, we were forced to introduce several new components into the simulator to more faithfully model LEO networks. Along the way, we attempted to make sure that our extensions were fully compatible with other aspects of the simulator, so that future researchers may investigate LEO networks along other lines (such as multiple access).

Figure 3.2 illustrates the major additions to the simulator.[2] We first introduced a spherical coordinate system, and added a *position object* to each network node. This position object can be given an initial coordinate and an equation which describes its trajectory through the coordinate system as a function of time. We centered the spherical coordinate system at the Earth's center, with the z-axis aligned with the Earth's rotation axis. This alignment simplified the description of polar orbits and trajectories for Earth terminals. The *link delay object*, which previously returned a fixed propagation delay, was changed to return a value based on the instantaneous positions of the two nodes at the end of the link.

The largest piece of coding involved link handoffs, because *ns* previously did not permit links to be dynamically detached and reattached to different nodes. Furthermore, we needed to introduce *handoff agents* to govern the handoffs. These agents are responsible for monitoring for opportunities to take down, bring up, or handoff links. Various policies for performing the handoffs can be implemented; we implemented asynchronous and synchronous handoffs as described above in Section 2.2.1. Finally, we implemented dynamic, distributed routing agents in each node for experiments with distributed routing described below in Chapter 6.

The default routing code in *ns* uses an all-pairs shortest path algorithm to compute new routes for each node in the simulator whenever the topology changes. This algorithm is useful to

[2] Since *ns* evolution is on-going, the exact structure of these enhancements that will be added to the publicly available simulator is subject to change.

populate routing tables initially for static topologies, but is very computationally expensive when applied to dynamically changing topologies because it has complexity of roughly $O(n^3)$. To speed up our simulations, we implemented single-source shortest path algorithms and configured the simulator to optionally compute routes on demand (whenever a packet needed to be sent), which yielded a run-time performance improvement of up to two orders of magnitude.

3.3 Experimental Testbed

We used the wireless testbed infrastructure of the BARWAN project at Berkeley. The BARWAN project was based on the vision of building future mobile information systems as a heterogeneous collection of *wireless overlay networks*. For example, a user may have a choice of connecting to an in-room infrared network, an in-building wireless LAN, a regional wide-area packet network, or even a satellite system. The research goals of this project were to tackle the problem of network access heterogeneity in such an environment. BARWAN solved problems associated with routing and handoffs within and between access networks, proxy-based application support, reliable transport over wireless channels, Web transport, and service location. References and a thorough overview of the project can be found in [20].

3.3.1 Experimental Machines and Software

Much of our experimental work involved machines on our local area networks, and most of the implementations involved changes to the networking code on the end hosts. We developed and experimented with modified TCP code and new Satellite Transport Protocol code on PCs running BSD/OS UNIX, version 3.0, from Berkeley Software Design, Inc. The networking stack in BSD/OS 3.0 resembles the code in the 4.4BSD-Lite distribution,[3] sometimes referred to as the “Net/3” release [127] and the source of many widely used systems like NetBSD and FreeBSD. This TCP/IP code has been developed and used over many years and is considered a stable source. Our experimental network consisted of 10 and 100 Mbit/s Ethernet segments joined by BSD/OS-based routers.

We used the `sock` program from Stevens to generate traffic for our experiments [127]. `sock` accesses the TCP/IP stack via standard system calls based on the well-known *sockets* Application Programming Interface (API) [127]. As described in later chapters, we sometimes used `sock` to simulate the transfer of large files, and at other times we used `sock` functions within another traffic generation program driven by traffic trace data. `sock` was trivially extended to support our STP experiments, as STP offers the same API as TCP, but via a system call with a different protocol number than TCP’s.

We used the network trace tools `tcpdump` and `tracelook` quite frequently in our packet trace analysis. `tcpdump` was written by Jacobson, Leres, and McCanne; it uses the *BSD packet filter* [86] to put a network interface into promiscuous mode so that all traffic on the interface can be observed. `tracelook` is a Tcl/Tk program written by Greg Minshall for graphically viewing the output of a tcpdump TCP tracefile.

Some of our experiments involved emulating the transmission characteristics of satellite channels. Rather than use a sophisticated satellite channel emulator, we used modified Ethernet

[3] The 4.4BSD-Lite distribution refers to the April 1994 version of the source code for a common reference implementation of TCP/IP developed by the Computer Systems Research Group at the University of California, Berkeley.

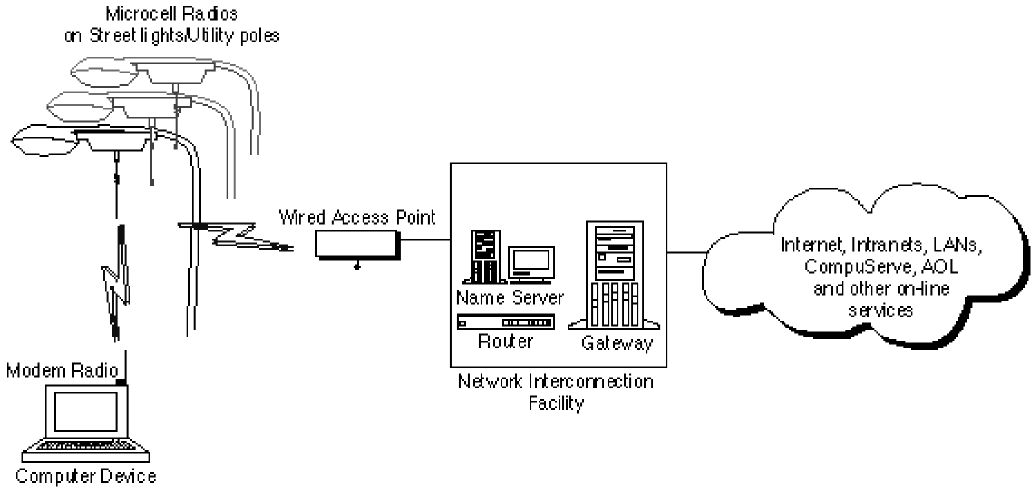


Figure 3.3: Architecture of the Richochet packet radio network (Source: Metricom, Inc.; used with permission).

device drivers (for BSD/OS) developed by Venkat Padmanabhan, a member of our research group. These drivers could buffer outgoing packets for a user-configured delay and also emulate a constrained bandwidth channel by imposing an additional delay based on the packet length and the emulated bit rate of the channel. We were also able to impose a random packet drop rate on the traffic through these drivers, which could be used to emulate a channel with a random, uniform bit error ratio. For transport protocol research over geostationary satellite channels, these drivers provided sufficient emulation, because the effects of a more precise modeling of the transmission characteristics of satellite channels are dwarfed by the dominant congestion-induced losses on Internet paths.

3.3.2 Ricochet Packet Radio Network

The Richochet packet radio network, deployed by Metricom, Inc., covers the Bay Area metropolitan region, as well as a number of other cities and airports in the United States. The system covers the region with telephone pole-top radios, and packets are routed through the radio network to one of several gateways to the Internet. The system uses frequency-hopping spread spectrum in the 915 MHz ISM band. The radios are half-duplex, meaning that they cannot simultaneously transmit and receive data. The radios also use a form of geographic routing to reach the nearest gateway; each radio is configured with its latitude and longitude, which it is able to announce to its neighbors, as well as the coordinates of a nearby gateway, and the radios route traffic to the neighboring radio that minimizes the geographic distance to the gateway. Figure 3.3 illustrates a modem that attaches to the serial port of a computer; the Point-to-Point protocol (PPP) [126] is used as an IP link layer between a computer and the gateway.

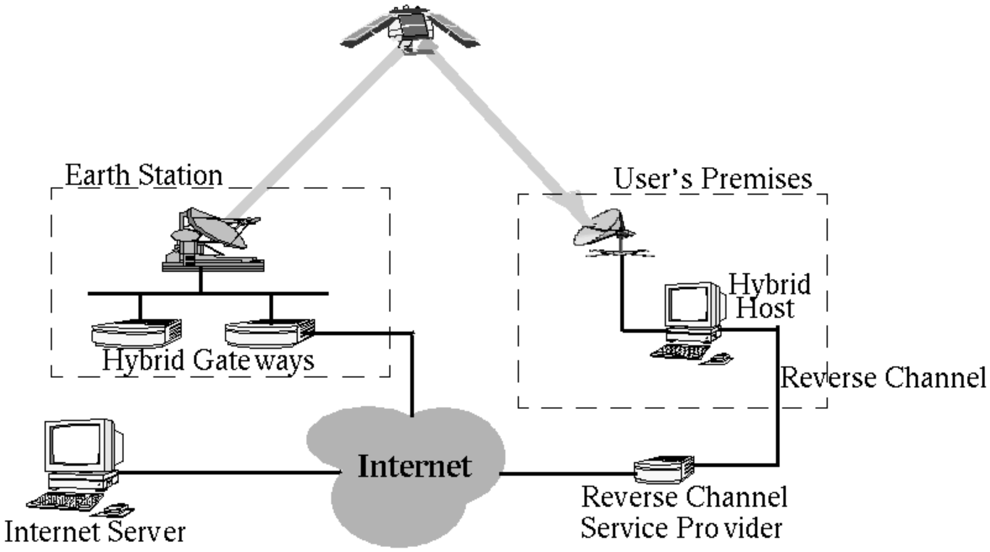


Figure 3.4: Architecture of the DirecPC satellite system (from [100]; used with permission).

3.3.3 DirecPC Satellite System

We used the DirecPC satellite system, developed and operated by Hughes Network Systems, for satellite experiments involving actual satellite channels. The DirecPC system is a hybrid Internet access system consisting of a high-speed unidirectional satellite broadcast channel and a return path accessed via a conventional Internet Service Provider (ISP). The system is based on the asymmetric traffic characteristics of typical end users, who use much more bandwidth inbound than outbound. Routing is performed by "tunneling" (encapsulating) outbound packets so that they are routed directly to the DirecPC network. From there, the packet is decapsulated and the inner, original, packet is sent onward to the destination Web site, but with a source address corresponding to the DirecPC network. In this manner, the packet can avoid being dropped by anti-spoofing filters commonly found in ISP networks, and the response traffic is naturally routed to the uplink gateway. Because of the asymmetric path, the end-to-end latency is around 400 ms, which is smaller than the 600 ms typically found on two-way geostationary satellite channels. Figure 3.4 provides an overview of the system.

To further our experiments, we placed a BSD/OS machine within the DirecPC network at their uplink facility in Germantown, MD. This enabled us to directly access the satellite uplink without traversing the Internet (which would have corrupted our forward data flow). We were able to remotely download our modified networking code to this machine.

3.4 Summary

In this chapter we have described our methodology and research infrastructure at a high level, while deferring detailed descriptions of experiments and simulations until later chapters when such details can be put in the proper context. We present the detailed results of our research in the next three chapters by first starting with the question of improving TCP performance over GEO satellite links.

Chapter 4

TCP Performance over Satellite Links

The TCP/IP protocol suite has become the lingua fraca of data communications, and the TCP protocol is used for most communications that require guaranteed, end-to-end reliability. Unfortunately, the performance of TCP is often degraded when the end-to-end path includes a satellite link. In this chapter, we focus on the problem of improving TCP performance over satellite links.

In recent years, the subject of TCP over satellite links, and more generally over wireless links, has been a fruitful research area. Nevertheless, the performance of modern TCP implementations over satellite links is still disappointing. In the Internet, satellite links are often used in the configuration depicted in Figure 4.1. In this configuration, the satellite provides access to the wired Internet. Furthermore, the cost of satellite transponder access generally dictates using the satellite link in an asymmetric bandwidth configuration (with more bandwidth allocated in the direction to the clients), or even in a hybrid unidirectional (broadcast) configuration with a telephone return path [100]. This kind of configuration has not been treated thoroughly in the literature, and in considering the configuration shown in Figure 4.1, we were led to consider the following research problems:

1. Satellite TCP connections for which a portion of the connection traverses the wired Internet are subject to severe throughput degradation if the packets flow through a queue that is being congested by connections with a short round-trip time (RTT). Can this bias against long RTT connections be overcome by simple changes to the congestion avoidance algorithm in end hosts?

2. In the current Internet, there exists a wide variety of TCP implementations with various options that interact in different ways. What is the best combination of (standard) TCP options and implementation guidelines for use over satellite channels?

3. How much performance advantage can be gained by "splitting" a TCP connection at a gateway located at the satellite terminal equipment connected to the wired Internet, thereby shielding the satellite subnetwork from the rest of the Internet?

4. In the case of split connections, how much further improvement could be gained by using a transport protocol specifically optimized for the satellite environment?

In this chapter, we focus on the first three questions raised above, and defer the fourth to Chapter 5. The first two questions mainly relate to improving various aspects of TCP's intertwined

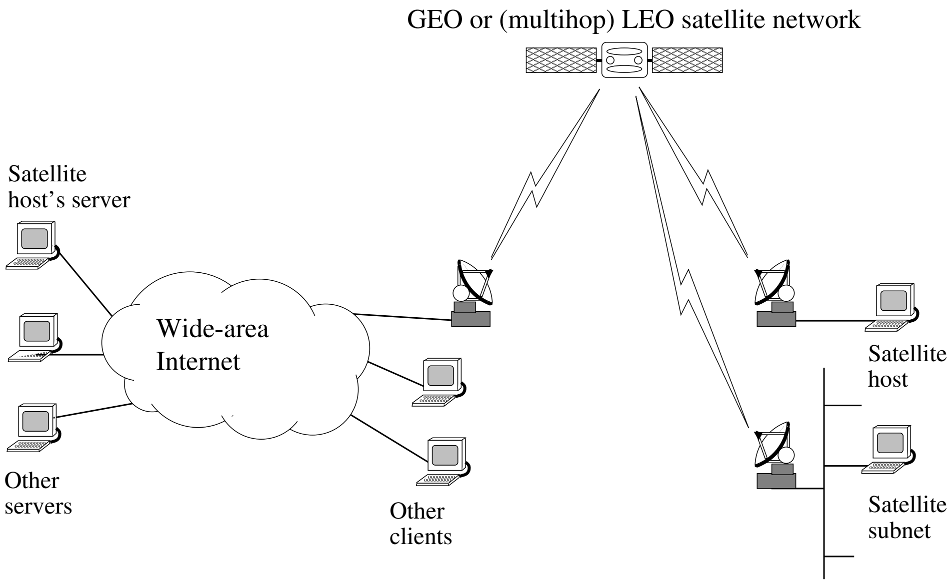


Figure 4.1: Example of a broadband satellite network in which a satellite-based host communicates with a server in the Internet.

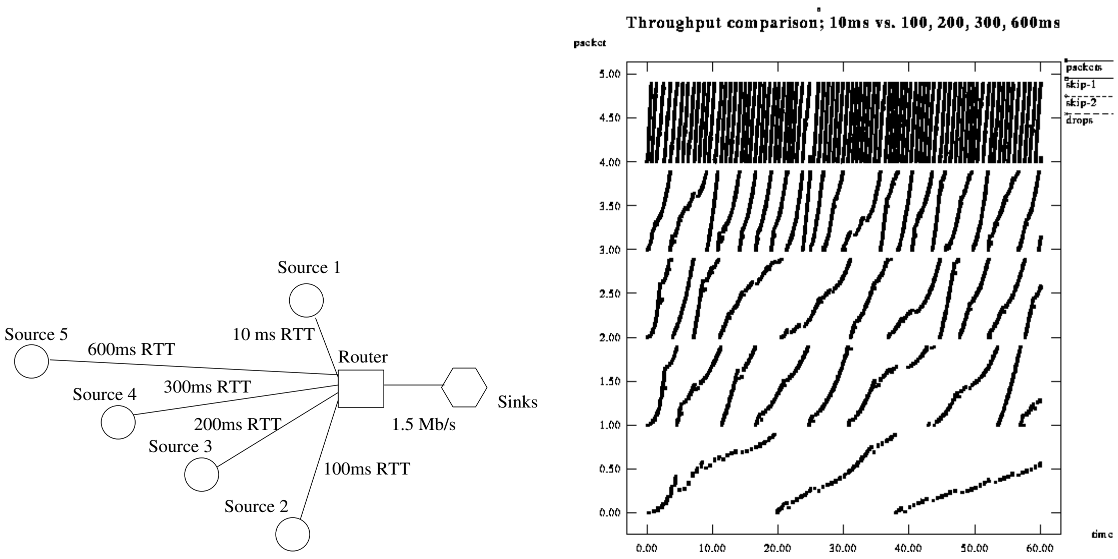
congestion control and *loss recovery* mechanisms. To address the first question, we investigate the potential improvement of changing the end-to-end behavior of the congestion avoidance algorithm. Next, we investigate the file transfer behavior of different variants of TCP SACK and present a standards-conformant algorithm that achieves high performance in a satellite environment. Finally, we address the third question by investigating the potential benefit of splitting the end-to-end connection at a gateway.

4.1 TCP Fairness in a Heterogeneous Environment

4.1.1 Introduction

The fairness problem in TCP is rooted in its congestion avoidance mechanism, which we described above in 2.1.1. The congestion avoidance phase is sometimes referred to as "additive increase and multiplicative decrease," because, in the absence of congestion, segments are added to the window over time, while in the presence of congestion, the window is halved (or multiplied by one half).

The "additive increase and multiplicative decrease" algorithm in TCP parallels a similar algorithm in the DECnet protocol [66]. Chiu and Jain showed that this algorithm leads to fair allocations of network bandwidth even though it operates in a distributed manner [28]. However, their analysis presumes that all connections in the network share the same additive increase rate and multiplicative decrease factor. In TCP, the multiplicative decrease factor (1/2) is the same for all connections, but the policy of an additive increase of one segment per round trip time (RTT) does



(a) Congested network topology. (b) TCP sequence number evolution vs. time.

Figure 4.2: A demonstration of the unfairness of the current TCP congestion avoidance algorithm. The connections, from top to bottom, have RTTs of 10, 100, 200, 300, and 600 ms respectively.

not provide a uniform increase in the rates of TCP connections with different RTTs[1]. In particular, connections with long RTTs open their window more slowly than those with short RTTs. And if a mixture of such short and long RTT connections share a bottleneck link, severe unfairness is inevitable as the short RTT connections grab the available bandwidth well before the long RTT connections have a chance [54].

Figure 4.2 illustrates an example of this problem by showing simulation results for a 60 second trace of TCP connections over the illustrated topology. In Figure 4.2, the evolution of the sequence number is plotted for 5 connections (from top to bottom, with RTTs of 10, 100, 200, 300, and 600 ms, respectively) sharing the same bottleneck link. The sequence number in this simulation is on a per segment basis, and the plots wrap after every 90 segments. The long RTT connections do not obtain an allocation close to their fair share of the bottleneck link, and their overall throughput performance suffers drastically.

To combat the bandwidth inequities that result from heterogeneous RTTs, Floyd proposed a modification to TCP's window adjustment algorithm that counteracts the RTT bias. In this section, we elaborate Floyd's "Constant-Rate" algorithm [39] with a thorough investigation of the performance achievable by both universally and selectively (i.e., incrementally) deploying a TCP with a modified window increase policy in the congestion avoidance phase of the connection.

Floyd [39] developed a fairly general characterization of window increase algorithms that facilitates the discussion of fairness. Although TCP maintains its send window in units of bytes, we find it more convenient herein to discuss it in units of segments. A key assumption is that a number of segments approximately equal to the send window size is sent every RTT; this is generally true for long RTT connections. Let c be the increase (in segments) in the size of the send window for a

[1] In the following, we distinguish between the overall congestion avoidance $algorithm$ and the $policies$ implemented in this algorithm such as the multiplicative decrease factor of 1/2 and the additive increase of one segment per RTT.

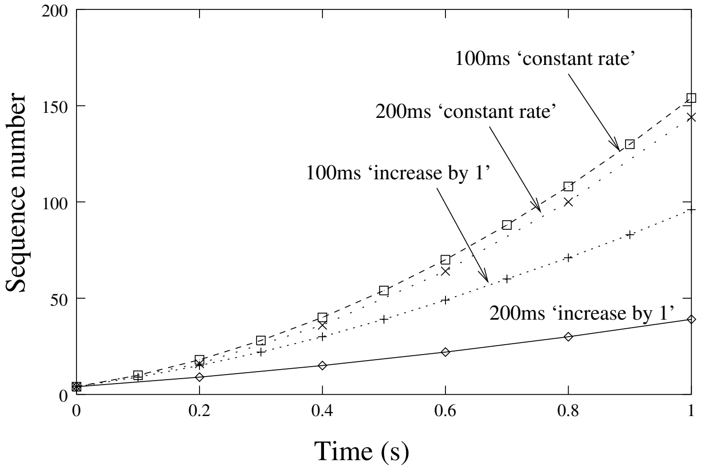


Figure 4.3: Simulated sequence number evolution for connections with different RTTs under both the constant rate and increase-by-one policies.

connection in one round trip time RTT. Therefore the window grows at a rate of c/RTT segments per second when additive increase is in effect. In conventional TCP, $c = 1$. Floyd refers to the standard TCP policy as an “increase by 1” policy.

If one were to scale the window growth rate c/RTT by RTT, the effect would be to build the window at a constant rate of c segments per second, independent of the RTT. However, the window growth rate does not equal the growth rate in data transmission. As [41] points out, it results in a “linear-in-RTT” bias in the sending rate. Because each connection can send a full window’s worth of segments each RTT, shorter RTT connections achieve greater throughput over a common time interval. To fully remove the bias, we must change the additive increase to $c * RTT$ segments per second; i.e., a factor of RTT^2 faster than the original algorithm. Floyd defines such an increase as a *Constant-Rate (CR) increase policy*, since it can roughly be interpreted as causing the rate of segment transmission to increase at a constant rate.

Figure 4.3 demonstrates the behavior of two of these policies for two connections with different round trip times. The figure plots the equation

$$P_{c_i,RTT}(t) = \sum_{k=0}^{\lfloor t/RTT \rfloor} (cwnd + kc_i),$$

for the different RTTs and policies c_i. This equation describes the number of segments sent ($P_{c_i,RTT}(t)$) assuming a window of segments is sent each RTT and there are no losses, where c_0 is equal to 1 (the standard policy), and c_1 implements the constant rate policy (and hence is a different value for each RTT). While only a rough approximation, the graph confirms the shape of the increase rate curve for each policy.

4.1.2 Methodology

We used the UCB/LBNL Network Simulator "ns" [2] to evaluate TCP performance. In addition to using the standard ns modules, we ported the HTTP traffic generator module from Bruce Mah's INSANE simulator[3], which gave us the ability to add a mix of realistic background traffic to our simulations.

Our study focused on large file transfer performance. While short HTTP transfers and Telnet connections over long RTT paths are also subject to performance degradation during periods of congestion, this degradation is due more to the fundamental latency of long RTT connections than to problems with congestion avoidance. Additionally, HTTP protocol implementations are migrating towards "persistent-HTTP" and longer duration TCP connections. We also did not assume the implementation of fair scheduling and TCP-friendly buffer management that can isolate flows or classes of flows from one another (e.g., as discussed in [131]), or pricing structures that might give network providers incentives to protect the throughput of paying customers. In short, we assumed an environment similar to the present day Internet, with the addition of Random Early Detection (RED) queues [44], and the latest in standardized TCP improvements (Selective Acknowledgments (SACK) [83] and large window enhancements [67]).

Performance Metrics

A number of metrics for quantifying fairness have been proposed but no single metric has common acceptance [39]. In this paper, we consider a "fair share per link" metric; i.e., if there are n flows through a bottleneck link, each flow has the right to $1/n$th of the capacity of that bottleneck link. Jain's metric of fairness [28] is applicable in this context. For n flows, with flow i receiving a fraction b_i on a given link, the fairness of the allocation is defined as:

$$Fairness \equiv \frac{(\sum_{i=1}^{n} b_i)^2}{n * (\sum_{i=1}^{n} b_i^2)}.$$

This metric ranges continuously in value from $1/n$ to 1, with 1 corresponding to equal allocation for all users. Utilization is another important metric, since high fairness is of little use if the link capacity is grossly underutilized. Utilization is defined herein as the number of original bits (i.e., not counting retransmissions) successfully transferred over a link during some time interval divided by the product of link rate and that time interval; this is often called "goodput."

$$Utilization \equiv \frac{(segments\_acked) * (segment\_size)}{(link\_rate) * time}.$$

Topologies

We explored a number of test configurations that allowed us to isolate selected behavior of both the standard and our proposed window adjustment policies. We selected the topologies illustrated in Figure 4.4; similar test configurations have previously been used by the research community to study the effects of congestion. The first test configuration (Topology 1) was used to

[2] http://www-mash.cs.berkeley.edu/ns/

[3] http://http.cs.berkeley.edu/~bmah/Software/HttpModel/

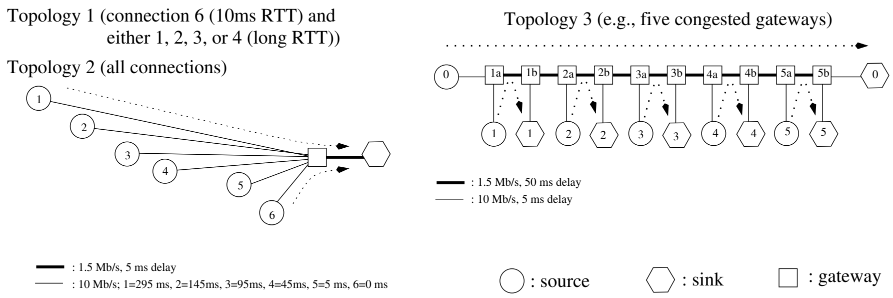


Figure 4.4: Three simulation topologies.

study the effects of two long duration connections, one with a short RTT (10 ms) and one with a long RTT (100-600 ms), sharing a single bottleneck link. The second test configuration (Topology 2) was used to examine the effects of many competing connections over a single bottleneck link. The six connections have RTTs of 10, 20, 100, 200, 300, and 600 ms. The third configuration (Topology 3) was used to examine the effects of long RTT connections that must also traverse a number of network hops populated by short RTT connections. This topology is very similar to the one previously used by Floyd to study the CR algorithm [39]; the number of congested gateways could vary between 1 and 10 (the figure illustrates 5 congested gateways).

Configuration Details

We studied the performance of two different TCP variants described above in Section 2.1.1: TCP NewReno, and TCP SACK. We note that other researchers have detected problems with using TCP Reno (a version of TCP that does not perform adequately when multiple drops occur in a window of data) in combination with congestion avoidance mechanisms that try to add more than one segment per RTT [17]; therefore, we avoided such implementations. We also examined two different queueing schemes: traditional "first-in, first-out" (FIFO) queueing, and Random Early Detection (RED) with packet discard.[4] In our simulations, data packet sizes were fixed at 1000 bytes, and the bottleneck link speed was 1.5 Mb/s. We examined a range of queue sizes from 4 to 50 packets, but in the data that follows, we concentrate on a RED queue size of 50 packets with a "minimum threshold" of 20 packets and a "maximum threshold" of 40 packets; all other RED parameters were set to the ns defaults. 20 packets in this case is approximately 100 ms at our output line rate.

[4] RED queues operate by computing an exponentially weighted moving average of the queue size. When the average queue size is below some minimum threshold, the queue does not drop any packets. When the average queue size is between the minimum and maximum threshold, the queue probabilistically drops incoming packets according to an algorithm described in [44]. When the average queue size exceeds the maximum threshold, the queue drops every incoming packet. The instantaneous queue depth can exceed the maximum threshold if the average queue depth is below the maximum threshold.

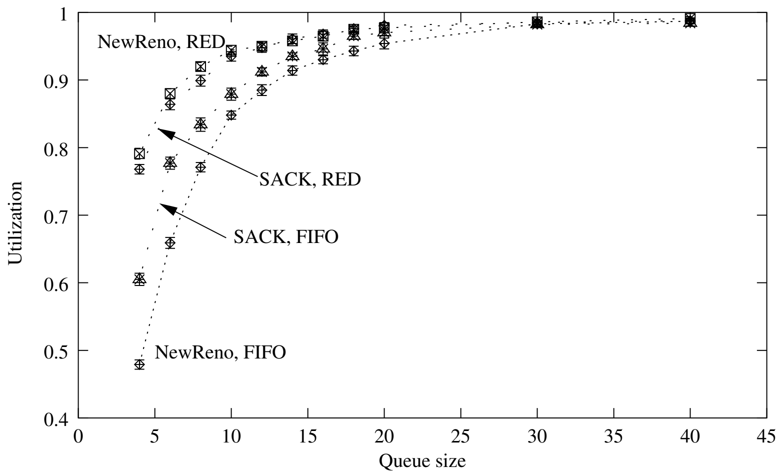


Figure 4.5: Benchmark performance results using Topology 2: Utilization vs. queue size.

Data Analysis and Presentation

TCP throughput in an environment containing random traffic can be quite variable, because small changes in initial conditions can cause wide variations in resulting behavior. Therefore, we computed the utilization and fairness of a particular configuration as follows. We first ran enough independent simulations such that the sample standard deviation of each connection's throughput was within 5% of its sample mean (this generally required around fifty runs). We then used these sample means to compute the fairness and utilization of a given topology. In the remainder of Section 4.1, if the experimental data does not explicitly list error bars, the reader may assume that the sample standard deviation is within 5% of the value listed. In the following subsections, we first provide some benchmark data, followed by an analysis of the Constant-Rate policy, followed by experiments aimed at selectively increasing the aggressiveness of a long-delay TCP connection.

4.1.3 Benchmark Results

To calibrate our simulation studies, we established a set of benchmark performance results for each of our test topologies. Though used principally to gauge the efficacy of our proposed policies, the benchmark data itself reveals some interesting effects. In this section, we examine benchmark performance data from Topology 2 in Figure 4.4.

Figures 4.5 and 4.6 plot the utilization and fairness of the standard TCP window adjustment policy for Topology 2. The error bars on these figures represent 99% confidence intervals, and are very small. We show results from the combination of two TCP variants, NewReno and SACK, with two queueing disciplines, FIFO and RED. In this representative data set (and in our other benchmark data), the following trends are evident:

- The utilization of the bottleneck link improves with increased queue size, because, under congestion, large queues keep the link busy as they drain out while TCP sources back off

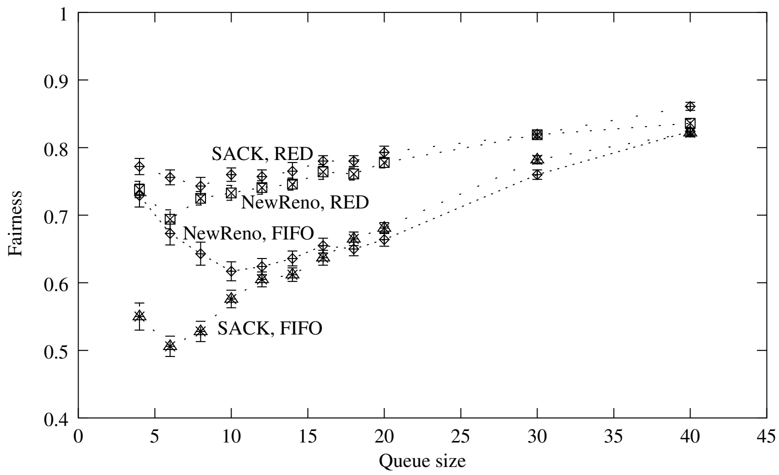


Figure 4.6: Benchmark performance results using Topology 2: Fairness vs. queue size.

their sending window. Additionally, larger queues absorb bursts of packets and prevent coarse timeouts, which cause long idle periods.

- Network fairness is poor in almost all cases, except when queue sizes are very large. In general, the two short delay connections obtained roughly 50% of the bandwidth, and the background WWW traffic consumed 20% of the bandwidth. The remaining 30% was split unequally among the 4 long RTT connections, with the longest connection receiving only about 24 to 64 kb/s (2 to 4%) for TCP NewReno with FIFO queueing. While even larger queue sizes may help further, they would also introduce more significant delay variability.

- In general, when queue sizes are reasonably large and when all TCPs use SACK instead of NewReno, the network fairness is marginally better. This is most likely due to SACK's superiority in recovering from multiple drops in a single window. Since multiple congestive losses in a single window are more likely to occur in a connection with a long RTT, the use of SACK helps such connections. However, the use of SACK by all TCP connections does not, by itself, remove the bias against long RTT connections.

In general, RED queues perform much better in terms of utilization and fairness than do FIFO queues. However, we found that the use of RED and SACK alone, without modifications to TCP sending behavior, still leaves much room for improvement in fair bandwidth allocation. RED queues equalize the bandwidth of flows with similar RTTs, but do not do so for flows with heterogeneous RTTs, as pointed out in [44]. In the remainder of this section, we focus on enhancing the performance of TCP implementations that use SACK and networks using RED queues.

4.1.4 Performance of the Constant-Rate Policy

In this section, we describe the case in which each foreground and background TCP connection uses a Constant-Rate (CR) policy. In TCP implementations, an additive increase to the TCP variable snd_cwnd (send congestion window) of approximately one segment per RTT, assuming an acknowledgment (ACK) is received for each segment, is effected by executing the following pseudocode upon receipt of a new ACK:

```
snd_cwnd = snd_cwnd + 1/snd_cwnd.
```

In this manner, the TCP connection gradually adds to its congestion window at the rate of approximately one segment per RTT; this approach to building the congestion window reduces transmission burstiness [66]. To implement a CR policy, we can modify the window increase algorithm to account for the RTT bias:

```
snd_cwnd = snd_cwnd + (c*rtt*rtt)/snd_cwnd,
```

where c is the constant that controls the rate. This policy causes an additive increase in the throughput rate that is the same for all connections. After initial experiments, we observed that the second term of the above equation could lead to very bursty send patterns, which led to increased losses. For example, if the RTT is large and the value of snd_cwnd is small, each ACK can trigger the transmission of several segments. To avoid this behavior, we bounded the increase per ACK by 1 segment; i.e.:

```
snd_cwnd = snd_cwnd +
min((c*rtt*rtt)/snd_cwnd, 1 segment).
```

With this constraint on the sender's behavior, the TCP connection is never more bursty than a TCP connection in slow start. Another approach, with which we did not experiment, would be to smooth the sending of several segments across a longer time period.

One question previously raised by Floyd is how to pick the proper value for the constant c. One way to think of the value of c for CR connections is how the aggressiveness of the CR connection would compare to that of a standard TCP connection with a certain RTT. For example, if $c = 100$, the value of the RTT that makes the numerator equal to 1 in our pseudocode above is 100 ms. Therefore, an environment in which $c = 100$ would have connections that were about as aggressive as normal TCP connections with 100 ms RTTs. We chose to experiment with a range of values, between $c = 4$ (as aggressive as standard 500 ms connections) and $c = 1600$ (25 ms).

In addition to varying the constant c, we experimented with several other variations in an effort to identify which types of environments were suitable for the CR policy:

- TCP NewReno vs. TCP SACK,
- RED vs. FIFO gateways,
- bottleneck queue lengths (or RED maximum queue thresholds) from 4 to 50 packets, and
- TCP RTT timer granularity of 500 ms (standard in many TCP implementations) vs. 10 ms.

As mentioned above, Topology 3 conforms closely to one with which Floyd experimented [39], and we were able, when using a similar value for the CR constant ($c = 4$), to confirm their results that CR can substantially improve the fairness of connections traversing multiple gateways when all connections use a CR policy. However, we also observed the following general trends in our data:

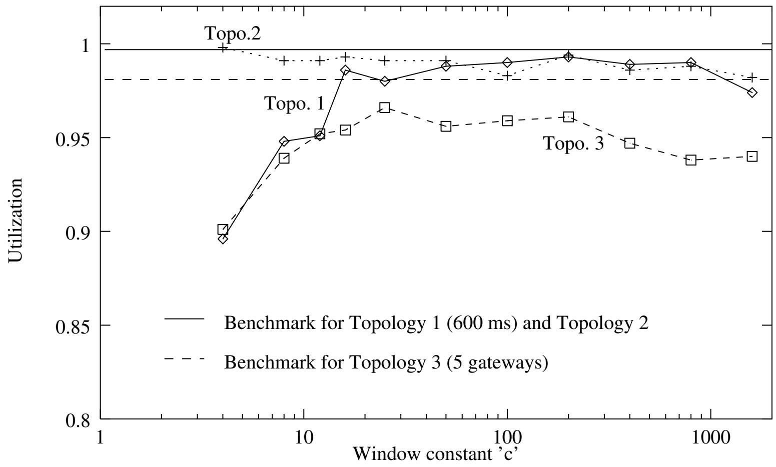


Figure 4.7: Utilization vs. window constant of TCP SACK with fine-grained RTT estimates over topologies with bottleneck RED queues.

- Deep RED queues appear to be a prerequisite for good performance of the CR policy. Performance when FIFO queues or short queues were used was very inconsistent, in the sense that there was often no value of c that simultaneously yielded high fairness and high utilization. Moreover, we could not determine strong correlations between the value of c and the fairness and utilization metrics we were using; i.e., the performance was highly sensitive to the particular simulation topology.

- TCP SACK and fine-grained RTT timers were the next most important indicators of good CR performance. The use of SACK helps TCP recover from losses more quickly, which leads to improved and more consistent performance. Also, many existing TCP implementations use a coarse estimate of the RTT, which impairs the ability of our modified congestion avoidance algorithm to determine the true RTT of the connection.

Figures 4.7 and 4.8 plot the utilization and fairness performance of TCP SACK over bottleneck RED queues when all connections, including background HTTP traffic, use the same CR policy, constant c, and RTT timer granularity of 10 ms. Topology 1 corresponds to the case in which the long RTT connection has a round trip propagation delay of 600 ms, while Topology 3 in this case corresponds to the topology with 5 congested gateways (the trace is taken from the first congested gateway). For comparison, we also plot as horizontal lines the utilization and fairness achieved when all TCP connections use the standard algorithm (i.e., benchmarks). The data indicates that the fairness can be substantially improved if all connections adopt the CR policy. However, the utilization suffered for small c when there were only two foreground connections in the topology (Topologies 1 and 3). When statistical multiplexing was in full effect (Topology 2), both fairness and utilization were near optimal for small values of c. Additionally, in Topology 3, although the fairness improved substantially, equal allocations were not obtained by the CR policy because the long RTT connection is also traversing multiple congested gateways.

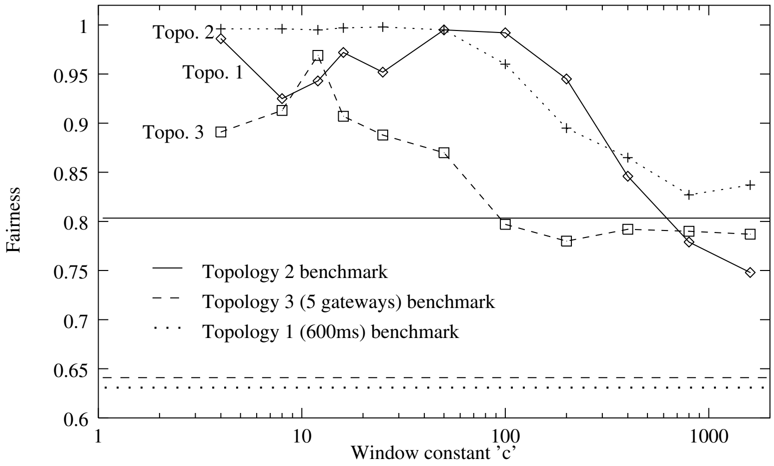


Figure 4.8: Fairness vs. window constant of TCP SACK with fine-grained RTT estimates over topologies with bottleneck RED queues.

In these experiments, as c became larger, connections became more aggressive, to the point that the bound in our policy of adding 1 segment per ACK was in effect nearly all of the time. Consequently, since the CR policy was no longer being applied, the unfairness reappeared. In general, we observed that the fairness properties were best when the value of c was below 100. However, if too few connections are using the link, such as in Topologies 1 and 3, such a small value of c can lead to lower utilization. Because it is difficult in practice for a given connection to determine the number and type of connections against which it is competing, we conclude the following negative result: a good choice of the constant c cannot be determined with high confidence on an operational basis.

Not only does the CR policy appear difficult to manage in a distributed network, we also found it susceptible to the presence of TCP connections operating under the standard policy. For example, Figure 4.9 illustrates the fairness performance when a single additional connection using the standard window increase policy was introduced into each of the topologies (and also included in the fairness computation). Although this additional connection slightly improved link utilization, much of the fairness improvement due to CR was lost when this competing connection was introduced, as it not only used a disproportionate share of the bandwidth itself but also acted as a "trailblazer," improving the performance of short RTT connections that were using the CR policy by a disproportionate amount. We also observed similar performance degradation if no extra file transfers were introduced, but instead the HTTP background traffic (20% of the bottleneck link rate, on average) used the standard policy. Similar effects (passive connections competing with more aggressive connections using the standard congestion avoidance algorithm) are also responsible for the poor performance of TCP "Vegas" in a long RTT environment characterized by a mix of heterogeneous TCP implementations [147].

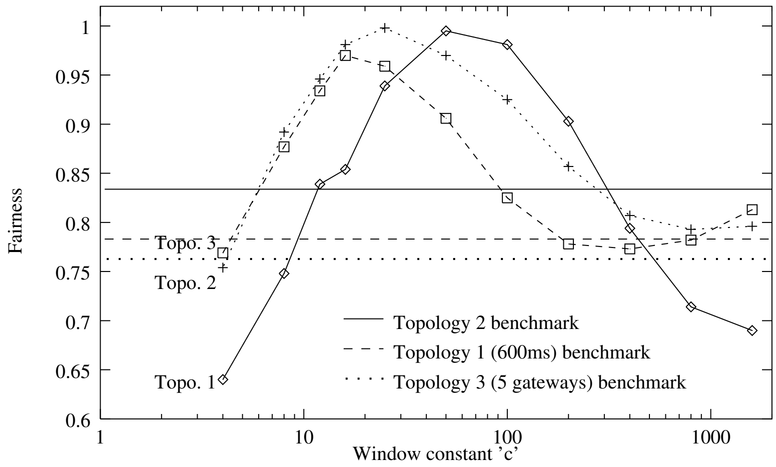


Figure 4.9: The sensitivity of the proper selection of the constant c when the effects of a standard TCP connection are added to the performance shown in Figure 4.8.

4.1.5 Selectively Modifying the Additive Increase Policy

We next investigated whether the throughput of an individual long RTT connection could be improved by modifying the additive increase policy of only the long RTT connection. We were interested in two questions:

• Can an individual connection improve its own throughput by becoming more aggressive during additive increase?

• If so, how does the individual connection's more aggressive behavior affect the performance of other (unmodified) connections using the same path?

To study the first question, we experimented with an "increase-by-K" (IBK) policy rather than the standard "increase-by-one," again limited by a maximum increase of one segment per ACK. In other words, we used the following pseudocode in our implementation:

```
snd_cwnd = snd_cwnd +
min((K/snd_cwnd),1 segment).
```

For example, by setting $K = 2$, we built the window by roughly 2 segments per RTT.[5] Again, the increase is bounded by 1 segment per ACK, but this is the general trend.

Figure 4.10 illustrates fairness results, calculated over the entire simulated network, for the case in which only one connection in the simulated topology used the IBK policy. In particular, we enabled the IBK policy on the longest RTT connections in each of the three topologies, and then repeated the experiment by enabling the IBK policy on only the 300 ms connection in Topology 2. In the graph, the value of $K = 1$ (left-most data points) corresponds to the normal (benchmark) case. We observed that the long RTT connection was able to steadily improve its performance over that of the benchmark case by increasing K across the range of values we considered. This resulted

[5]If delayed acknowledgments are being used, this is akin to correcting the window growth penalty that is due to delayed acknowledgments.

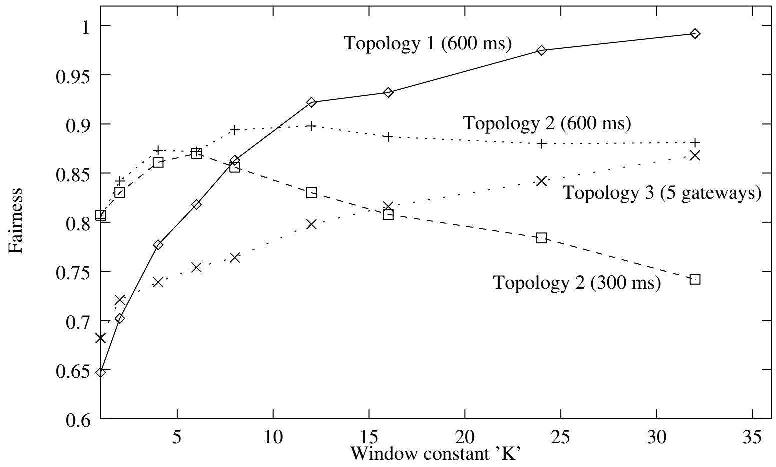


Figure 4.10: Improvement in fairness vs. window constant due to the IBK policy.

in improved fairness in all topologies for small values of $K > 1$. For larger values, even though the throughput of the long RTT connection continued to improve, fairness actually decreased in some topologies as the more aggressive connection began to take more bandwidth than its fair share.

In Figure 4.10 we plotted the performance of TCP SACK over RED queues, and found that there was no limit to the improvement that a more aggressive connection could obtain for itself. We repeated the experiment for TCP NewReno over FIFO queues, and found that connections could increase their own performance, independent of the topology, by using a value of K of up to 4 or so. However, for higher values of K, performance degraded, because the sending behavior became too bursty for the FIFO queues to successfully absorb.

Because the performance improvements result from increasing TCP's aggressiveness, we should be concerned that this can have a negative impact on other peer connections. Remarkably, we found that, in every case we examined, the average fairness index *always* improved, and the average utilization held relatively constant, when the more aggressive connection used a modest value of K (less than 8 or so). This improvement occurred regardless of whether TCP SACK or NewReno was used, or whether FIFO or RED queues were present. In fact, the majority of the redistributed bandwidth came from connections that were already using more than their fair share. The effect on other connections was similar to what they would have experienced had the long RTT connection actually been a connection with a somewhat shorter RTT.

For example, Table 4.1 provides an example of the magnitude of the performance gains achievable. For the value $K = 4$, we tabulate the utilization, fairness, and throughput of the four longest RTT connections in Topology 2. In the first column are results from when each connection used the standard policy, the second column shows the results from when only the 300 ms connection used an IBK policy, and the third column is from when only the 600 ms connection used an IBK policy. This table illustrates that the more aggressive connections did not seriously harm the throughput of the peer connections. The throughput gains (highlighted in bold font) are substantial; in many cases, throughput increases (and hence reductions in user-perceived latency) exceeded

	benchmark	300 ms	600 ms
Utilization	0.99	0.99	0.99
Fairness	0.807	0.861	0.873
	Throughput (kb/s)		
600 ms	**68.1 (0.9)**	59.6 (0.6)	**138.7 (2.0)**
300 ms	**116.2 (1.1)**	**240.8 (2.8)**	109.1 (1.1)
200 ms	156.6 (1.7)	139.8 (1.5)	151.6 (1.1)
100 ms	217.1 (2.1)	188.1 (1.9)	207.0 (1.8)

Table 4.1: Effect of the IBK policy on throughput (Topology 2, K = 4). 99% confidence intervals are shown in parentheses.

100% when RED queues were used, and 50% when FIFO queues were traversed.

We next investigated whether the performance gains were sustainable when multiple connections become more aggressive by examining this case with Topology 2. We found that the aggressive connections were able to simultaneously improve their own performance, although their relative performance gains were not what they would have achieved had they been the only aggressive connection. Of course, if every connection adopted an IBK policy, the fairness situation would be back to the standard case, so there must be some RTT threshold beyond which connections can become more aggressive if such a policy is to work in practice. Finally, we experimented with the case in which congestion was induced in both directions of data transfer. This had the effect of disrupting the ACK stream to some extent, but did not significantly affect our main results.

4.1.6 Implementation Issues

We have already discussed some minor implementation changes to the code segment which builds the congestion window. Throughout the discussion, we implicitly assumed that the TCP connection had an accurate estimate of its RTT. In practice, this is not the case. TCP does maintain a smoothed round trip time ($srtt$), but because of the timer granularity of 500 ms in TCP, this value is not very accurate. It is, however, rather easy to improve the RTT accuracy through use of the TCP timestamps option [67]. A sending TCP implementation can put a more accurate timestamp in the TCP timestamp field, which is merely reflected by the receiver; such a technique is suggested for TCP Vegas [16]. However, it is important not to base the retransmission timer value on this accurate timestamp, because TCP fast retransmit and fast recovery rely on the $srtt$ variable being somewhat larger than the actual RTT. We experimented in ns with running the TCP timer granularity at 1ms instead of 500 ms, and found that the more accurate $srtt$ value caused coarse timeouts to trigger before fast recovery could be accomplished.

One practical issue is that modifications to the sending algorithm of an implementation have little use if the implementation is a client of a large data transfer rather than the source of the data. However, it is possible to indirectly modify the sender's behavior by actions taken at the receiver. For example, by sending more ACKs back to the sender, the receiver can "speed up" the sender; e.g., if one were to receive a segment of 1000 bytes, sending ACKs 1:250, 251:500, etc. would quadruple the window build rate. We did not experiment with this technique, and note that it is of limited utility when the reverse channel is constrained and cannot handle the additional ACK traffic.

Another alternative to increasing the aggressiveness of a single connection is to run multiple connections in parallel, coordinated by the application or some type of session manager. This technique, sometimes called "striping", has been in use for some time by satellite operators and WWW browser software. One advantage of this approach is that it overcomes small offered windows by the receiver. This technique, however, can impact the network more than our approach since multiple slow starts are launched into the network simultaneously. Recent results in which congestion window state is shared across the multiple connections can potentially combat the problem [11]. In general, the use of striping without the constraints outlined in [11] are not favorably viewed by the research community.

4.1.7 Summary

In this section, we have presented the results of our investigation of simple changes to TCP's congestion avoidance algorithm in an effort to improve its fairness properties. While we found that the Constant-Rate (CR) policy could improve fairness dramatically, we faced two practical difficulties that would likely prevent universal deployment of this scheme in its current form: i) the proper selection of a constant is dependent upon the network topology and the number of peer connections and is therefore difficult to determine in a distributed manner, and ii) the fairness benefits of the CR policy can be confounded by competing connections using standard congestion avoidance, thereby making it disadvantageous to deploy CR in an existing heterogeneous environment. However, when we instead made only certain long RTT connections slightly more aggressive, we were always able to improve network fairness while keeping bottleneck link utilization relatively constant by using an increase-by-K (IBK) policy. Interestingly, the effects on other unmodified connections that were sharing the bottleneck link were similar to what they would have experienced had the modified connection actually been a connection with a shorter RTT.

Our results indicate that it may be beneficial for long RTT connections (running TCP SACK) to become slightly more aggressive during the additive increase phase of congestion avoidance. While our data set is not comprehensive enough to allow us to advocate a particular policy at this time, the IBK policy for small values of K (such as 2 or 4) may significantly improve the throughput while not significantly impacting other flows. Such a policy could be invoked in practice when the TCP implementation detects that the connection has an RTT above a certain threshold (for example, connections traversing a GEO satellite link have a much larger RTT– at least 500 ms– than terrestrial connections). A TCP receiver could even induce the sender into an IBK policy by acknowledging data in smaller chunks. Determining appropriate values for K as a function of RTT, as well as determining the accuracy and resolution required of TCP's RTT estimates, could be the focus of future work.

4.2 End-to-End TCP Performance over Satellite Links

4.2.1 Introduction

In this section, we quantify just how well state-of-the-art TCP implementations perform in a satellite environment composed of one or more satellites in geostationary orbit (GEO) or low-

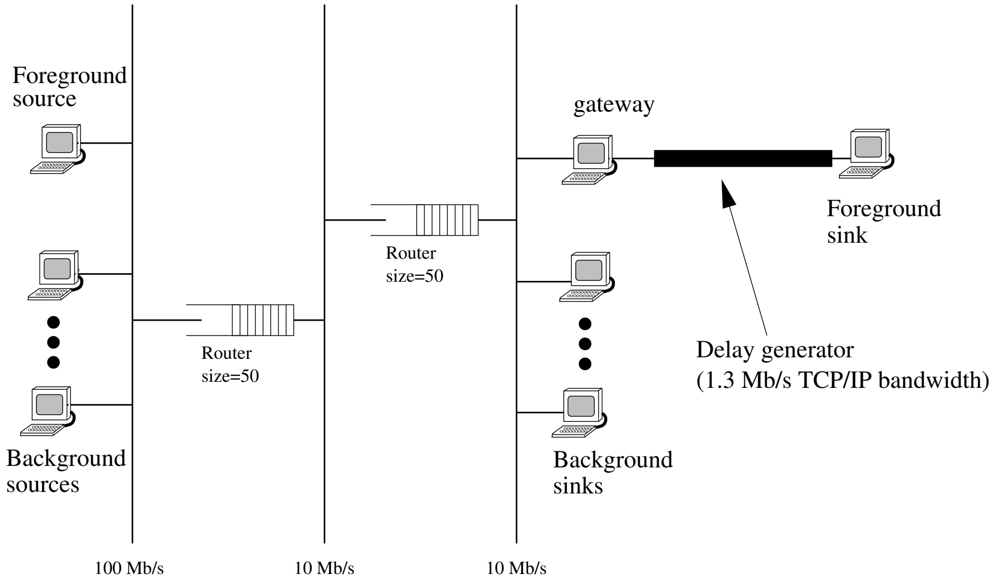


Figure 4.11: Configuration for network experiments.

earth-orbit (LEO), particularly when the satellite connection forms only a part of the end-to-end connection, as shown in Figure 4.1. We focused on two types of workload found most commonly in the Internet: large file transfers, and short Web connections.

Our assumptions about future satellite network characteristics are shaped by projections of future commercial systems (e.g., Teledesic [130], Spaceway [38]) that will offer Internet connections at up to broadband (tens of Mb/s) data rates via networks of LEO or GEO satellites (or hybrid constellations). Users may contact other hosts in either the satellite network or the wide-area Internet. We discussed some of our assumptions about the transmission and congestion characteristics of the end-to-end path using such satellite systems in Section 2.1.2. In short, we assume future satellite networks characterized by low BERs, potentially high degrees of bandwidth and path asymmetry, high propagation delays (especially for GEO based links), and low internal network congestion. These assumptions were used to drive our protocol design and performance analyses described in the rest of this chapter.

4.2.2 Methodology

This experimental methodology pertains to the remainder of this chapter and also to the results presented in Chapter 5.

Experimental Setup

Our experiments were conducted using hosts, running BSD/OS 3.0 UNIX, connected to Ethernets in a local-area subnet at Berkeley. The TCP implementations on these machines are derived from 4.4BSD-Lite (also known as Net/3 [144]), with modifications to support our experiments.

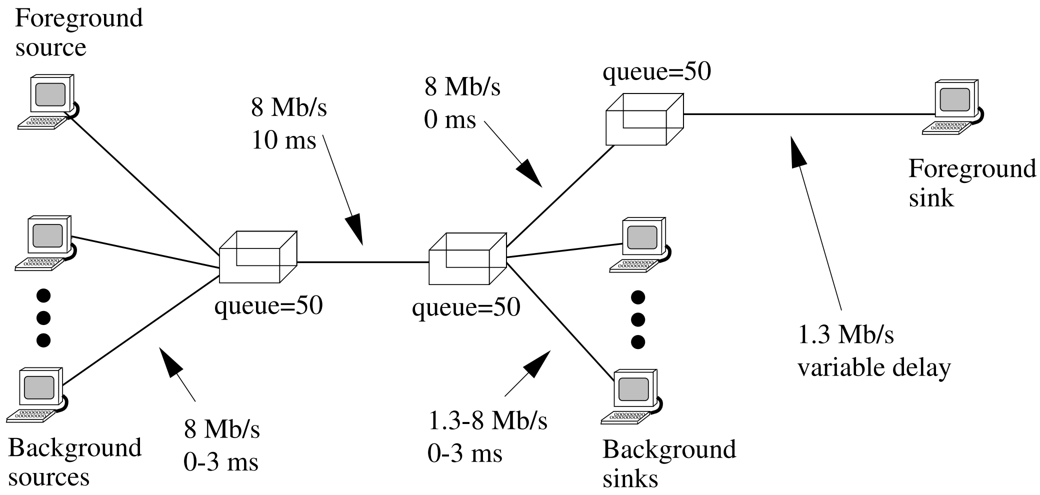


Figure 4.12: Configuration for simulation experiments.

We configured the receivers to offer the largest window possible (240 KB) to the senders. For the experiments, traffic sources were connected to a 100 Mb/s Ethernet, and traffic sinks were on a 10 Mb/s Ethernet separated by a 10Mb/s transit Ethernet segment. Figure 4.11 illustrates the experimental topology. To generate traffic, we used a combination of the `sock` program [127] for bulk file transfers and a HTTP traffic generator for testing of "4K slow start" and T/TCP. This traffic generator generated small file transfers according to empirical distributions drawn from Bruce Mah's HTTP traces [79]. We implemented STP in the BSD/OS UNIX kernel.

For investigating satellite transport protocol performance, it is usually sufficient to experiment with delay and error simulators rather than with detailed emulators of the transmission channel. To emulate satellite links, we used modified device drivers that delayed sending a packet onto the Ethernet for a deterministic amount of time. These drivers can also constrain the maximum rate at which a host can send data. We modeled GEO satellite links by a constraint of 1.3 Mb/s of TCP/IP bandwidth (i.e., approximately T1 rate at the physical layer), on a 600 ms RTT link. LEO satellites were modeled by a constraint of 1.3 Mb/s with a fixed RTT in the range of 40-400 ms [49]. Our links had no bit errors or variation in propagation delay, which, while not representative of all satellite links, exemplifies the common case.

In addition to controlled experiments performed in our local environment, we also describe experiments in Chapter 5 involving two commercial networks in our wireless networking testbed. We used a network based on a direct broadcast satellite (the Hughes DirecPC system, which covers the contintenal US), and a packet radio network (the Metricom Ricochet system, deployed in the San Francisco Bay area). For DirecPC experiments, we sent data from a computer located at the DirecPC uplink center at Germantown, MD over the satellite link to a multi-homed host on one of our subnets. We used the wide-area Internet to return acknowledgments to the traffic source. To emulate a normal user experience with the DirecPC system, we constrained the return link to be bandwidth limited to 50 Kb/s to simulate a modem connection. Although not a satellite network, the Ricochet network offers a challenging environment for transport connections, includ-

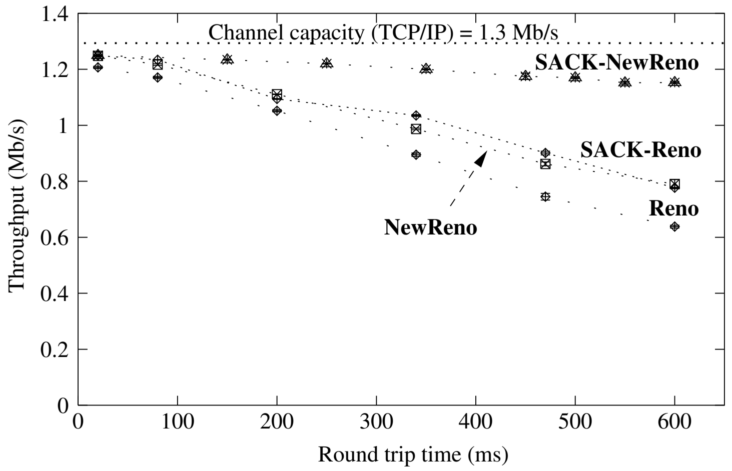


Figure 4.13: Throughput performance of TCP SACK NewReno, TCP SACK Reno, TCP NewReno, and TCP Reno over an experimental path with a TCP/IP bandwidth of 1.3 Mb/s and no transmission errors. Data points represent the sample means from 20 independent transfers of 10 MB each. In this and subsequent figures representing a large number of experimental results, error bars represent 95% confidence intervals.

ing asymmetry and large latencies; we used this network only for testing of the STP protocol as described in Chapter 5. In these experiments, a wired host at Berkeley communicated with a host on the Ricochet network using the packet radio network in both directions.

Simulation Configuration

We used *ns*, described above in Section 3.2, to test simulated topologies that matched our experimental setup. We aligned the TCP modules to match our implementations, and wrote a STP simulation module to closely emulate the implementation used in the experiments. We also used a background HTTP traffic generator, similar to that used in the experiments, to lightly load the network topology and to break up any TCP phase effects [43]. Our simulation topology, which conformed closely to the experimental setup, is shown in Figure 4.12.

4.2.3 Performance for Large File Transfers

TCP is the dominant protocol for file transfers (FTP) in the wide-area Internet. In this section, we describe simulations and experiments used for characterizing file transfer performance over satellite links.

To maintain high throughput for large file transfers, the TCP congestion window must be large. This implies that the congestion avoidance and loss recovery mechanisms are very important in determining performance. In this section we examine the performance of four variants of TCP loss recovery and congestion control, which we first introduced in Section 2.1.1:

- **TCP Reno** The unmodified TCP implementation in our BSD/OS 3.0 operating system is commonly known as TCP Reno. Many modern TCP implementations are largely based on this version of TCP. Of the satellite-friendly TCP extensions described above, BSD/OS 3.0 supports window scale and path MTU discovery.

- **TCP NewReno** TCP "NewReno" is a collection of bug fixes and refinements for how TCP Reno handles the fast recovery phase of congestion avoidance. Our TCP NewReno implementation is similar to TCP Reno except that it avoids false fast retransmissions [60], multiple window reductions in one window of data [36], and constrains the burstiness of the sender upon leaving fast recovery [36]. Specifically, it implements the "Less Careful, Slow-but-Steady" variant of NewReno described in [42].

- **TCP SACK-Reno** Reno congestion avoidance algorithms may be combined with the SACK option for loss recovery to form TCP "SACK-Reno."

- **TCP SACK-NewReno** Likewise, this corresponds to TCP NewReno congestion avoidance with the SACK option for loss recovery.

It is important to emphasize that all of the above implementations would be regarded as conformant to the TCP standards; in practice, many more variants of TCP exist.

For our file transfer experiments, we repeatedly transferred 10 MB files across our testbed while varying the latency of the emulated satellite channel. The file transfers lasted at least 60 seconds, allowing the low throughput of the initial slow start phase to be amortized across the lifetime of the connection. In the simulations, we added a number background HTTP traffic generators to the topology so as to introduce low levels of cross traffic (approximately 10% of the forward throughput of the channel). These traffic generators did not by themselves congest the forward path; the TCP losses were periodically self-induced by the greedy nature of the congestion avoidance mechanism of the persistent file transfers. In the experiments, which were conducted on operational networks during early morning periods of light network activity, the low amounts of live traffic on the networks and the variable processing delays of the hosts sufficed to add variability to the experiments.

Behavior of Several TCP Variants

We plot the results of these experiments in Figure 4.13. In all of our figures, throughput is defined as "application-level" throughput. For values of RTT less than 100 ms, the performance is relatively high for all four variants. However, for GEO delays (600 ms) and for LEO delays greater than 100 ms, the difference in performance for different TCP implementations is quite evident. By analyzing packet traces in both the simulations and the experiments, we determined that the main distinction between the implementations was in their behavior immediately upon leaving the slow start phase of congestion avoidance. It is critical that TCP transition from slow start to congestion avoidance in a smooth manner, with a congestion window close to the bandwidth-delay product of the path. We found the performance of SACK-NewReno congestion avoidance to be the best; in this case, when a slow start overshoot occurs, the protocol cuts its window in half once and smoothly moves to congestion avoidance after recovering all losses. There is little penalty for using a high-bandwidth, high-latency GEO satellite link in this case. When SACK was used without NewReno enhancements (SACK-Reno), we observed that the slow start termination, which is characterized

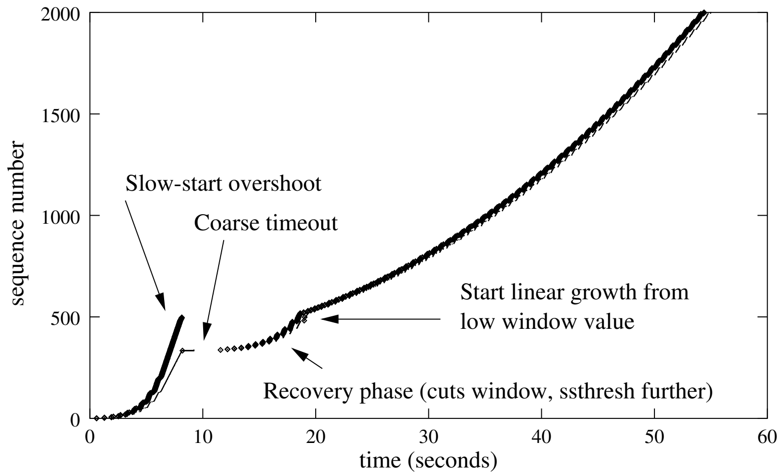


Figure 4.14: Typical performance, using a standard BSD TCP (Reno) implementation, of a large file transfer over a GEO satellite channel.

by several bursts of packet losses, resulted in the implementation cutting its congestion window in half several times, rather than just once. As a result, TCP was forced to rebuild its window linearly from a very low value. The performance of NewReno without SACK was similar but for a different reason. In this case, the slow start overshoot resulted in similar bursty patterns of losses, but since NewReno, unlike SACK, can only recover one loss per RTT, it spent a large portion of time recovering from the slow start losses. Finally, TCP Reno rarely avoided a retransmission timeout and multiple reductions in its window after the first slow start, resulting again in slow window growth.

A closer look at the behavior of these different TCP variants is informative. Figure 4.14 illustrates a "time-sequence" plot of an individual connection– the initial 60 seconds of a large file transfer using an unmodified BSD/OS UNIX TCP implementation (TCP Reno without SACK) over the topology illustrated in Figure 4.11. Two plots are overlaid– the evolution of the sender's sequence number, and the evolution of the acknowledgments received (the trace of the sender's sequence number generally lies to the left of the acknowledgment trace and is marked by larger points). The connection initially starts in slow-start, and although the connection takes several seconds to make noticeable progress, within the first 10 seconds the connection has already overshot by a wide margin the capacity of a router along the path, resulting in many packet drops (not necessarily contiguous in the sequence space). The implementation performs fast retransmission, but since many packet losses have occurred, the implementation invariably is forced to recover with a coarse timeout because it does not interpret the arrival of a partial acknowledgment (that is, an acknowledgment that does not cover all of the data that was outstanding at the time of the retransmission) as a sign that the next unacknowledged packet is missing. The timeout cuts the window to one segment and the slow start threshold in half; normally this would allow the sender to rapidly ramp up after the timeout to half of its previous window (that caused congestion). However, because the receiver's buffer has many out-of-order packets (and holes to fill), as the post-timeout

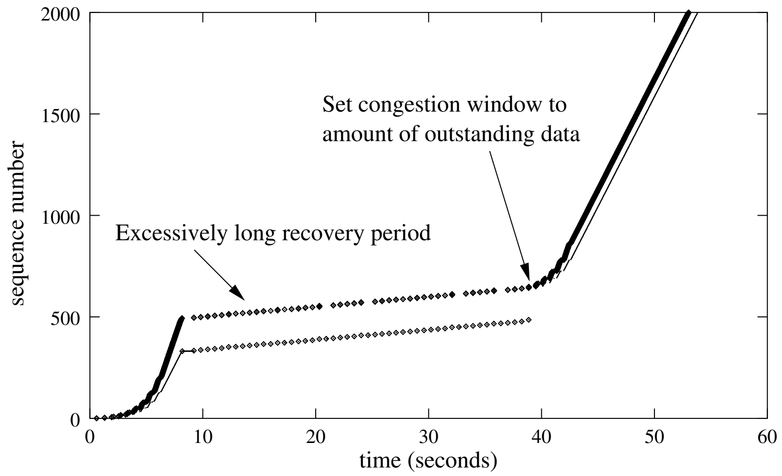


Figure 4.15: Correct NewReno behavior, using a modified BSD TCP implementation, of a large file transfer over a GEO satellite channel.

TCP sender starts to send more data, it can receive a number of duplicate acknowledgments that push it in and out of fast retransmission again (these are sometimes called "false fast retransmissions" [60]), each time cutting the window and slow start threshold in half. The result is, by the time that the sender has recovered from all of the original losses, it has a very low congestion window and slow start threshold value, and is forced to build its window linearly from a very low value, resulting in poor throughput.

TCP NewReno was devised to correct this oversight in the TCP Reno implementation; it defines a "recovery phase" that ends when all of the packets that were outstanding when the first loss was detected are acknowledged. Figure 4.15 illustrates the typical performance of this algorithm. No timeouts occur during the recovery and the window is not reduced multiple times for the same burst loss. However, since the recovery takes one round trip time for each gap in the sequence space to be recovered, the result is a TCP connection that takes over half a minute to recover from a single burst of losses. For this reason, as pointed out by Floyd [42], it may be beneficial to prevent this behavior from occurring by forcing TCP to take a timeout if it requires too many round trips to recover. Second, in our experiments with this algorithm as specified by [60] and [36], we noticed undesirable behavior that occurred at the end of the recovery phase. This behavior (immediate reentry into a burst loss situation) can be seen in Figure 4.16, and it is due to a burst of packets that can occur at the end of recovery. This burst occurs if, during window inflation of the recovery phase, the transmission of new segments was constrained by the receiver's offered window (because the "window inflation" step of TCP Reno can result in very large artificial windows being generated). As a result, when the hole is plugged in the reassembly buffer and the TCP sender resets its congestion window upon receipt of the acknowledgment, it is eligible to immediately send many segments. The solution to this problem, as shown in Figure 4.15, is to constrain the congestion window at the end of recovery to be no larger than the amount of outstanding data at that time (plus one segment, to allow a new transmission). This proposal was first described in [76].

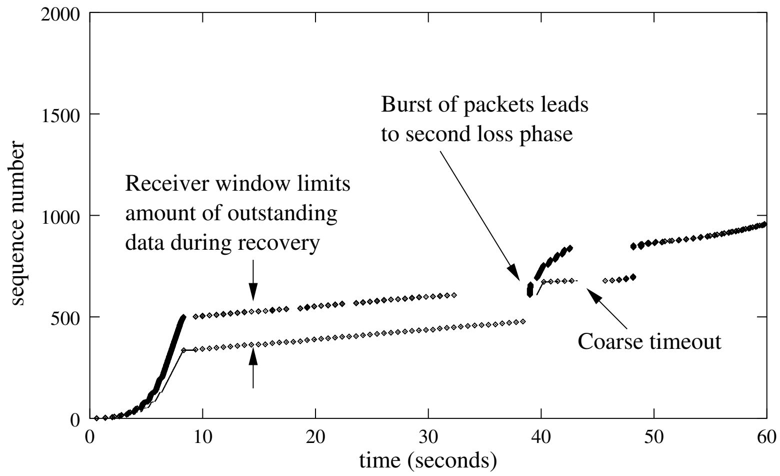


Figure 4.16: Incorrect NewReno behavior, using a modified BSD TCP implementation, of a large file transfer over a GEO satellite channel. This behavior is due to a burst of packets transmitted at the end of recovery.

The best behavior is obtained by combining both the SACK and NewReno algorithms, as illustrated by Figure 4.17. In this case, TCP recovers very rapidly from bursty losses because the extra information present in the SACK option gives the TCP sender a more complete picture of what is missing. Figure 4.18 illustrates this recovery in more detail; the burst of losses is recovered in less than two seconds (that it requires more than a round trip delay is due to queueing delays that have built up), and a large window is preserved for the subsequent connection to use in linear growth phase.

Finally, we illustrate in Figure 4.19 the close correspondence between simulation and experimental results for file transfers. In the remainder of this section, we present only our experimental results since our simulation results were generally in close agreement. The TCP implementations of the *ns* simulator are very realistic, to the point that bugs found in common implementations can also be enabled in our simulations.

Effect of a Competing Connection

The above experiments are appropriate to model connections entirely within a satellite subnetwork, but do not accurately portray conditions found when using the satellite network to access sites on the wired Internet, where competition for bandwidth from many different connections (with shorter round trip delays) can lead to network congestion and unfairness in bandwidth allocation. For our next experiments, we added a single, large-window persistent connection from a background source to a background sink in the same direction as the foreground file transfer. In our topology, this caused the first router in the network to occasionally become congested. Note that this background connection does not traverse any portion of our emulated satellite subnet. The results in this case are strikingly different. It only takes one low delay (in this case, 20 ms RTT) connection to drastically reduce the achievable throughput for SACK-NewReno, as shown in Figure 4.20. This

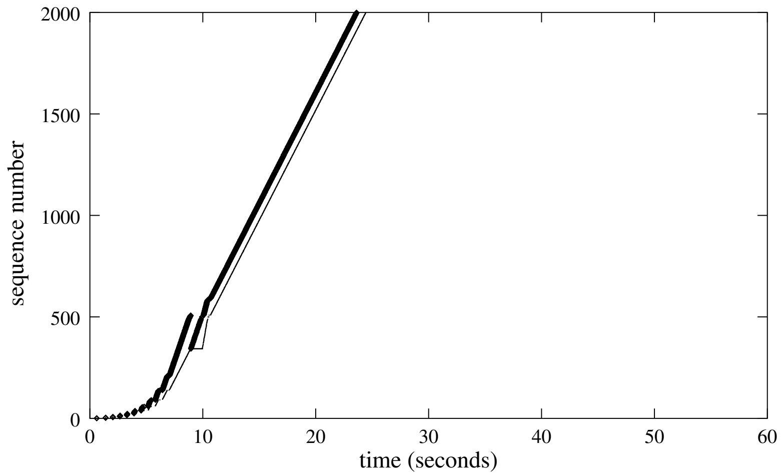


Figure 4.17: Correct SACK behavior, using a modified BSD TCP implementation (including NewReno loss recovery), of a large file transfer over a GEO satellite channel.

is the TCP fairness problem identified earlier in this chapter. TCP's fairness properties can be the first-order determinant of how well a large-window satellite TCP connection can do in the wide-area Internet. Even though the satellite connection was successful in avoiding timeouts in almost all of the transfers, the window reductions due to recurring fast retransmits substantially reduced the throughput. The throughput is also much more variable under these conditions, as represented by the error bars. The main problem is that the connection with the long RTT is too sluggish to rebuild its window and push data through the congested queue before it takes another loss.

In summary, we observed that TCP SACK with NewReno congestion avoidance is able to sustain throughputs at close to the bottleneck link rate even for GEO-like delays. This is because TCP is able to amortize the low throughtput of the initial window build across a longer period of high throughput. However, our data illustrates that the use of SACK alone is not sufficient to enable high performance. Specifically, NewReno helps to avoid coarse timeouts and multiple window reductions, while SACK accelerates the loss recovery phase. Specific details of our SACK-NewReno implementation can be found in Appendix A. Finally, the result we would like to emphasize, which agrees with our analysis in Section 4.1, is that it only takes very moderate levels of congestion in the wide-area Internet to drastically impair the performance of even well-configured TCP connections.

4.2.4 Performance for Web Transfers

Besides file transfers, most of the rest of the TCP traffic in the Internet is driven by Web transfers. Such connections are very different from file transfers. Typically, an Web client issues a small request to a server for an HTML (HyperText Markup Language) page. The server sends the initial page to the client on this first connection. Thereafter, the client launches a number of TCP connections to fetch images that fill out the requested page or to obtain different pages. Each item

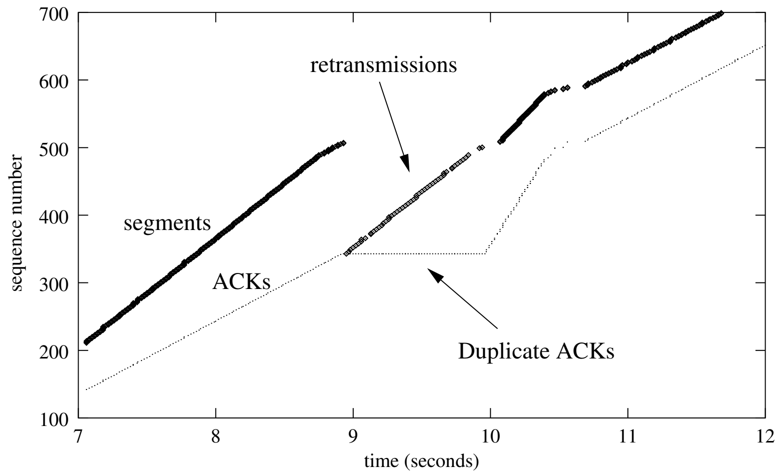


Figure 4.18: Closeup of rapid SACK recovery of multiple losses.

on the page requires a separate connection.[6] Many common Web browsers allow a user to operate multiple (typically, four) TCP connections in parallel to fetch different image objects. Basically, the data transfer model is "client request, server response."

Using standard TCP, any connection requires a minimum of two RTTs until the client receives the requested data (the first RTT establishes the connection, and the second one is for data transfer). As the RTT increases, the RTT can become the dominant portion of the overall user-perceived latency, particularly since average Web server response times are much smaller than one second [51]. Two mechanisms described in Section 3 attempt to alleviate the latency effects of TCP for short connections. The first, T/TCP, does away with the initial handshake (RTT) of the connection. The second, 4KSS, allows the TCP server to send up to 4380 bytes in the initial burst of data. If the size of the transfer is no more than 4380 bytes, the transfer can complete in one RTT. By using some simple analysis, we can quantify the beneficial effects that these TCP mechanisms have on the user-perceived latency.

Figure 4.21, adapted from a similar figure in [57], illustrates the latency in a hypothetical three segment reply using standard TCP. We make the following assumptions:

- We do not model server response times or segment transmission times. We assume an environment in which the RTT is the dominant latency in the transfer.[7] Server response times and segment transmission delays are a constant offset to the latencies we calculate; i.e., the same offset must be added no matter what version of TCP we are considering.
- We assume no packet losses and a fixed RTT. Therefore, these latencies are the best case.
- We do not model some of the bugs that have appeared in early HTTP implementations and that

[6] We will discuss shortly a modification to this approach, known as Persistent-HTTP (P-HTTP), which reuses the same TCP connection for multiple items.

[7] This is not always true in practice. Even for fast links, server responses can take several seconds, but on average, the server response time is much less than a second [51].

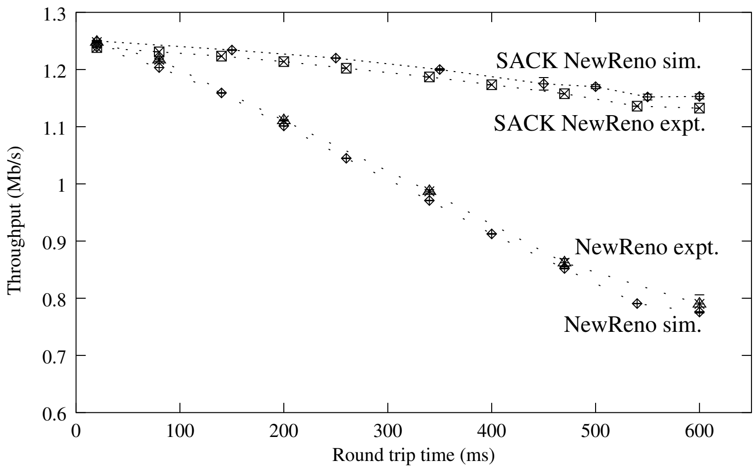


Figure 4.19: Agreement between simulation and experimental results for TCP SACK and TCP NewReno.

are discussed in [57], under the assumption that they will gradually disappear. For example, one quite prevalent bug allows the connection to start with an initial congestion window of two segments [128].

With these assumptions in mind, consider Figure 4.21, in which dashed lines denote control packets and solid lines indicate data packets. The first RTT is consumed by a SYN exchange, after which the client issues an HTTP GET request. Upon receiving and responding to this request, the server at this point has a congestion window of one segment. Assuming that the TCP implementation implements delayed acknowledgments (delayed ACKs) of up to 200 ms [127], the client on average will acknowledge this data after 100 ms. Upon receiving the acknowledgment, the congestion window grows to 2, and the server sends the second and third segments, followed by a FIN, which closes its half of the connection. The client must close its own half of the connection, but we do not model this delay since it does not contribute to user-perceived latency. Therefore, the total amount of TCP-related latency is 3 RTTs + 100 ms in this case. Using either T/TCP or 4KSS would reduce the latency to 2 RTTs, and using both mechanisms would reduce it to a single RTT.

We used HTTP traces to compute probability mass functions (pmfs) for the number of bytes transferred per HTTP connection. We then computed the average TCP latency for all of these file sizes, based on a simple analysis of how the congestion window builds over time. Because some transfers were very long, we eliminated those over 100 segments (only 2-4% of the data set, in general). For these cases, it is more realistic to consider them as large file transfers. Our trace data was gathered from two different user populations. The first, collected by Mah in 1995 [79], comes from a well connected Berkeley subnet. The second set, collected by Gribble in 1997 [51], comes from Berkeley residential usage over dial-up modems. By using this trace data with our model, we estimated the minimum, median, and mean latency effects of TCP on user-perceived latency. For GEO networks, we modeled the RTT as a fixed 600 ms, and for LEO networks we assumed a RTT of 80 ms. To verify the analytical results, we also performed measurements using similar pmfs to

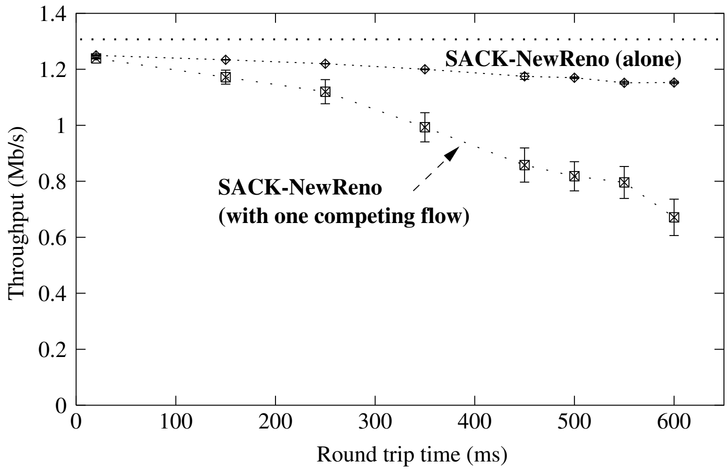


Figure 4.20: The effect of a single competing short-delay connection on the satellite connection's throughput. The competing connection was a persistent file transfer using TCP SACK NewReno with a nominal 20 ms RTT between a background source and sink in the experimental topology.

drive a TCP traffic generator in our experimental topology, and we recorded the latency experienced along with the file size for each file transfer. For the experiments, we did not cull the large transfers from our trace data. The experimental setup captured the effects of not only the propagation delay but also the processing delays in real end systems.

In Table 4.2, we present the results from an analysis of the data set provided by Mah [79]. The first three columns of data list the minimum, median, and mean TCP transfer times required, according to the analysis of the trace file and assuming a maximum segment size of 1500 bytes. These values were calculated by first determining the TCP related latency for a connection of a given size, and then by weighting these latencies according to the pmfs derived from the trace data. The fourth column lists experimental results corresponding to this data set. These values are the mean (and 95% confidence interval) of 1000 independent transfers, in which the size of the transfer was generated randomly according to the pmfs drawn from the trace data. The last four columns are similar to the first four, except for the use of a maximum segment size of 500 bytes. This data indicates that the use of either T/TCP or TCP with 4KSS improves mean latency by a small amount, but the combination of both options yields an improvement by a factor of two to three. The relative improvement is similar whether GEO or LEO networks are assumed (because the analysis is based on RTT). Because the mean latencies using the assumed LEO network are already rather small, the improvements due to TCP optimizations are less likely to be perceived by users. The data set provided by Gribble [51] contained slightly larger transfers, on average, but the same trends in TCP latency were present.

Finally, the most recent version of the HTTP specification (version 1.1 [37]) recommends that servers and clients adopt the persistent connection and pipelining techniques known as "persistent-HTTP" (P-HTTP) [102]. Rather than using separate TCP connections for each image on a page, P-HTTP allows for a single TCP connection between client and server to be reused for

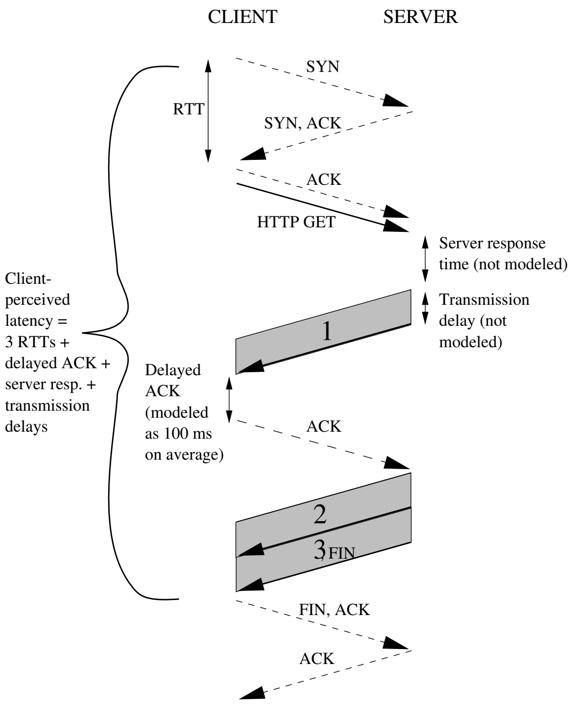


Figure 4.21: TCP latency of a 3 segment server reply using standard TCP.

multiple objects. The shift to P-HTTP offers a tradeoff in performance for satellite connections. On the one hand, P-HTTP is potentially much more bit-efficient than HTTP with standard TCP, because connections are not set up and torn down as frequently (the connection establishment costs are identical to those of T/TCP [57]). However, in terms of latency, the use of T/TCP and multiple, concurrent connections may yield faster Web page loads under some scenarios. The capability of many Web browsers to support multiple, concurrent connections is an example of a general technique known as "striping," which has been a strategy for transport protocol improvement known to satellite network operators for some time, and which has most recently been studied in the context of FTP [6]. Because TCP and HTTP optimizations such as T/TCP, and TCP with 4KSS do not yield major performance improvements for most users of the Internet [57], it is unclear whether they will see deployment. In fact, Padmanabhan recently studied the potential benefit of not using P-HTTP but instead reverting back to multiple, concurrent TCP connections that share congestion window and other state information [99].

In summary, for connections using GEO satellite links, TCP optimizations such as T/TCP and 4KSS, especially when used together, can yield a reduction of two to three times in in user-perceived latency and can also reduce the bandwidth overhead of HTTP connections. For LEO satellite links, optimizations to reduce the number of unnecessary control packets are desirable, but optimizations to reduce latency will not have as perceptible of an effect for users because propagation delays are smaller. However, since such optimizations benefit only a small user community, it is possible that they will not see widespread deployment.

	Geostationary orbit (600 ms RTT)							
	1500 byte segments				500 byte segments			
	minimum	median	mean	expt. mean	minimum	median	mean	expt. mean
Standard TCP	1.2	1.9	1.9	2.0 (0.1)	1.2	2.5	2.5	2.6 (0.1)
T/TCP	0.6	1.2	1.2	1.4 (0.1)	0.6	1.8	1.7	2.0 (0.1)
TCP "4KSS"	1.2	1.2	1.4	1.6 (0.1)	1.2	1.8	1.7	1.9 (0.1)
T/TCP "4KSS"	0.6	0.6	0.8	1.0 (0.1)	0.6	1.2	1.1	1.3 (0.1)
	Low-earth orbit (80 ms RTT)							
Standard TCP	0.16	0.34	0.31	0.37 (0.02)	0.16	0.42	0.42	0.55 (0.02)
T/TCP	0.08	0.16	0.17	0.28 (0.02)	0.08	0.24	0.25	0.47 (0.02)
TCP "4KSS"	0.16	0.16	0.18	0.25 (0.01)	0.16	0.24	0.23	0.31 (0.01)
T/TCP "4KSS"	0.08	0.08	0.10	0.16 (0.01)	0.08	0.16	0.15	0.23 (0.01)

Table 4.2: TCP latency effects on HTTP transfers for GEO and LEO satellite connections. Trace data is taken from [27]. All latencies are in seconds. For the experimental results, 95% confidence intervals are shown in parentheses.

4.3 Split TCP Connections

Although TCP can work well over even GEO satellite links under certain conditions, we have illustrated that there are cases for which even the best end-to-end modifications cannot ensure good performance. Furthermore, in an actual network with a heterogeneous user population, users and servers cannot all be expected to be running satellite-optimized versions of TCP. This has led to the practice of "splitting" transport connections. This concept is not new; satellite operators have deployed protocol converters for many years. In this section, we describe how TCP connections may be split at a satellite gateway, identify some drawbacks to split connections, and quantify how much improvement can be obtained.

4.3.1 Split Connection Approaches

The idea behind split connections is to shield high-latency or lossy network segments from the rest of the network, in a manner transparent to applications. TCP connections may be split in a number of ways. Figure 4.22 illustrates the most general case, in which a *gateway* is inserted on the link between the satellite terminal equipment and the terrestrial network. On the user side, the gateway may be integrated with the user terminal, or there may be no gateway at all. The goal is for end users to be unaware of the presence of an intermediate agent, other than improved performance. From the perspective of the host in the wide-area Internet, it is communicating with a well-connected host with a much shorter latency. Over the satellite link, a satellite-optimized transport protocol can be used.

TCP may be split in the following ways:

- **TCP spoofing** In this approach, the gateway on the network side of the connection prematurely acknowledges data destined for the satellite host, to speed up the sender's data transmission [146]. It then suppresses the true acknowledgment stream from the satellite host, and

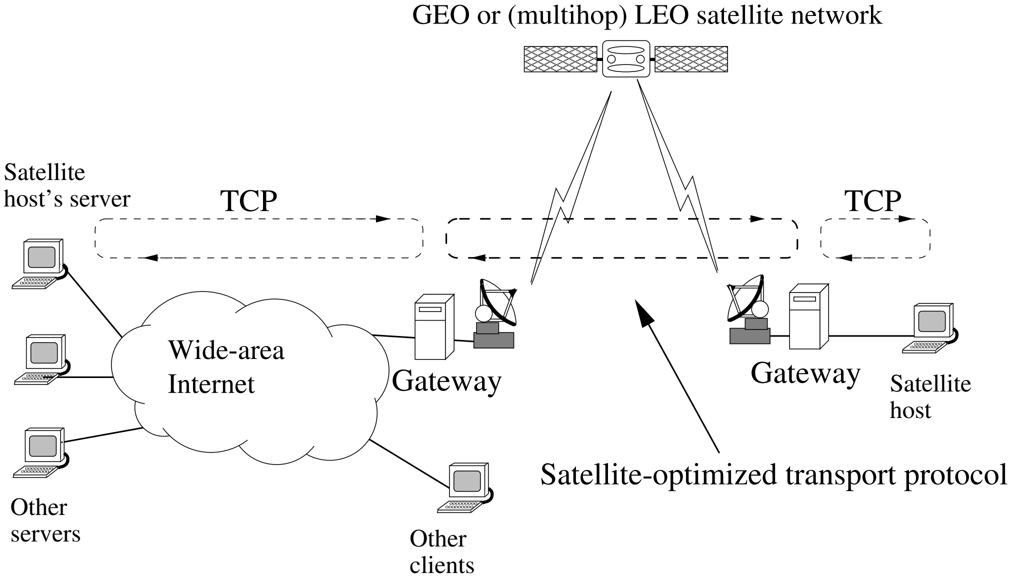


Figure 4.22: Future satellite networking topology in which a satellite-based host communicates with a server in the Internet through a satellite protocol gateway.

takes responsibility for resending any missing data. As long as the traffic is primarily unidirectional, TCP datagrams are passed through the gateway without alteration. In the reverse direction, the same strategy is followed. No changes are needed at the satellite client.

- **TCP splitting** Instead of spoofing, the connection may be fully split at the gateway on the network side, and a second TCP connection may be used from the satellite gateway to the satellite host. Logically, there is not much difference between this approach and spoofing, except that the gateway may try to run TCP options that are not supported by the terrestrial server. Modern firewall implementations often perform a type of TCP splitting (such as sequence number remapping) for security reasons.

- **Web caching** If satellite-based Web users connect to a Web cache within the satellite network, the cache is effectively splitting any TCP connection for requests that result in a cache miss. Therefore, Web caching not only can reduce the latency for users in fetching data from the Web, it has the side benefit of splitting the transport connection for cache misses.

Furthermore, when the TCP connection is fully split at a gateway or cache, it is possible to use an alternative protocol for the satellite portion of the connection. While this requires the use of a satellite gateway or modified end-system software on the satellite host's side, this approach may provide better performance by improving on TCP's performance in ways not easily achieved by remaining backward compatible with existing implementations. Set-top boxes or other user terminal equipment may provide a natural point for the implementation of protocol conversion (back to TCP, if necessary) on the satellite host's side of the connection.

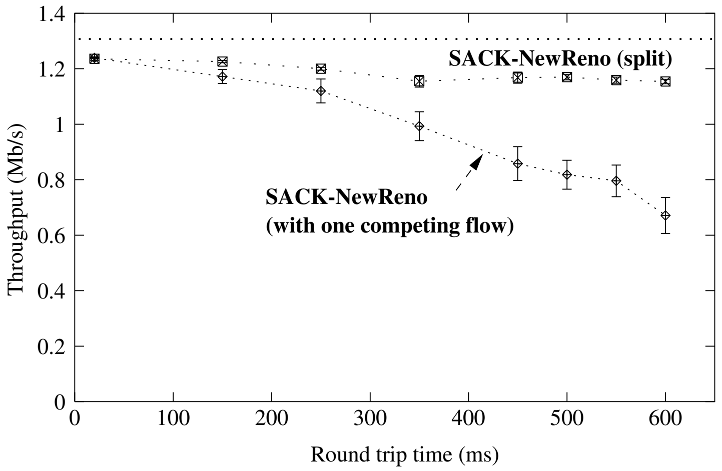


Figure 4.23: Forward throughput performance of split TCP in the presence of a short-delay competing connection. TCP SACK NewReno with large windows was used on both connection portions.

In all three approaches, the amount of per-connection buffering required at the gateway is roughly 2-3 times the bandwidth-delay product of the satellite link or the Internet path, whichever is smaller. The computing resources required to support a large set of users (approximately 200-500 KB of memory per active connection, plus processing) can form a significant portion of the hardware requirements of a satellite Internet gateway. In addition, although persistent-HTTP connections will reduce the number of connections that need to be set up and torn down, they will also drastically lower the duty cycle of each TCP connection, requiring the gateway to keep resources allocated for idle connections. However, it is important to emphasize that if Web caches or other proxies are already part of the satellite network architecture, there would be no need for extra equipment to support transport-level gateways.

Besides the resource consumption noted above, split connections are not without other hazards. First, from an architectural standpoint, a split TCP connection that is not explicitly associated with a proxy or a cache breaks the end-to-end semantics of the transport layer. Although approaches for TCP improvement over local area wireless links, such as Berkeley's "snoop" protocol [10] and "mobile TCP" [21], can preserve end-to-end semantics, it is more difficult to do so in the satellite environment because combating the fairness problem relies on early acknowledgment of data. However, steps can be taken to ensure that the connection does not close normally unless all data has been received; for example, the gateways can allow the FIN segment of TCP to pass end-to-end. Furthermore, higher layer protocols typically have mechanisms to restart a transport connection if it prematurely fails. Second, gateways introduce a single point of failure within the network, and require all traffic for a given connection to be routed through them (i.e., there can be no alternate packet routing). Third, protocol conversion gateways are ineffective if IP-level encryption and authentication protocols are operating on a link, although they can still function normally if the encryption and authentication is performed at the transport layer. In the case of IP-level security, the transport gateway must be included as part of the "trust infrastructure" to operate. Typically,

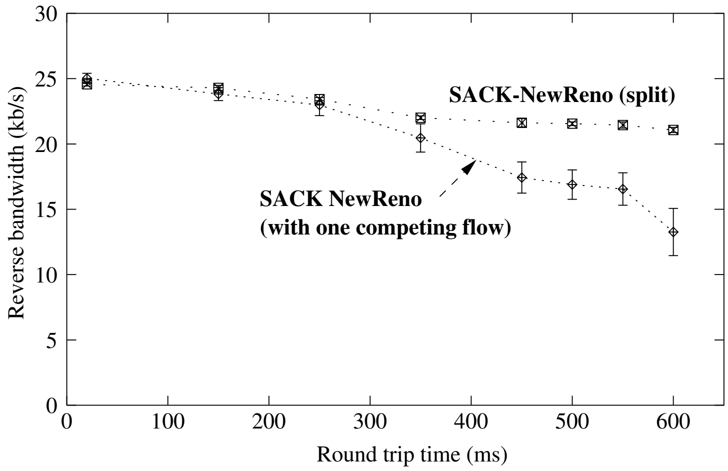


Figure 4.24: Reverse channel utilization of split TCP in the presence of a short-delay competing connection. TCP SACK NewReno with large windows was used on both connection portions.

however, if a satellite network is used to provide “last-mile” access to a large diverse set of users, transport-level security protocols will be used instead of IP-level security; in this case, protocol gateways can operate correctly.

4.3.2 Split Connection Performance

In Figure 4.23, we illustrate the performance gains achievable when the TCP connection is split at the gateway between the satellite network and the Internet, under the same conditions as shown in Figure 4.20 (a competing short delay connection in the Internet). We replotted the relevant data from Figure 4.20 for comparison. Note that the presence of the gateway allows the split connection to compete for bandwidth in the wide area and obtain its fair share. However, as shown in Figure 4.24, the reverse channel usage required for this TCP connection is roughly 20 Kb/s. This usage scales linearly with the forward throughput, and for 1000 byte segments, is roughly 2% of the forward throughput achieved. For bandwidth-constrained reverse channels as will be the case in most satellite systems, this sets an upper bound on the forward throughput achievable if TCP relies on a stream of acknowledgments to clock out new data. This suggests that it would be useful either to make modifications to TCP to reduce its reverse channel usage (such as using modifications to handle TCP asymmetry [11]) or to use a protocol over the satellite portion of the connection that uses less bandwidth. We investigate the latter possibility in the next chapter.

4.4 Summary

In this chapter, we have investigated the performance of IP-compatible transport protocols over satellite links from several perspectives. Our main results are as follows:

i) We observed little degradation in TCP performance for connections with RTTs in the range of future LEO systems (40-200 ms), although we did not investigate potential problems due to large RTT variations. However, maintaining good TCP performance over GEO latencies (or long LEO paths) is challenging.

ii) While we found that the Constant-Rate (CR) policy could improve TCP fairness dramatically, we faced two practical difficulties that would likely prevent universal deployment of this scheme in its current form. First, the proper selection of a constant is dependent upon the network topology and the number of peer connections and is therefore difficult to determine in a distributed manner. Second, the fairness benefits of the CR policy can be confounded by competing connections using standard congestion avoidance, thereby making it disadvantageous to deploy CR in an existing heterogeneous environment.

iii) When we instead made only certain long RTT connections slightly more aggressive, we were always able to improve network fairness while keeping bottleneck link utilization relatively constant by using an increase-by-K (IBK) policy. Interestingly, the effects on other unmodified connections that were sharing the bottleneck link were similar to what they would have experienced had the modified connection actually been a connection with a shorter RTT.

iv) If the right TCP options are used and congestion is light, TCP can work well for large file transfers even over GEO links. In particular, in our large file transfer experiments with TCP SACK plus NewReno congestion control, average throughput decreased by no more than 10% when the RTT was increased from 20 ms to 600 ms. However, we showed that even low levels of competition from short delay flows (in the form of cross-traffic in the wide-area Internet) significantly degrades the satellite connection's performance.

v) Concerning the latency due to HTTP exchanges, we found that the use of both T/TCP and modified slow start performed much better than either option used separately, and could cut the average TCP-related latency by a factor of two to three for GEO links.

vi) We showed that the performance problems due to mis-configured TCP or network congestion can be alleviated by splitting the TCP connection at a gateway within the satellite subnetwork. Even with congestion in the wide-area Internet, the end-to-end connection is still able to maintain high throughput.

TCP has proven to be a very robust protocol in a variety of network environments. However, this chapter has illustrated that obtaining good performance using standard end-to-end TCP connections is very challenging in a GEO satellite environment. For file transfers, the best performance results that we obtained were based on splitting the connection at a gateway, where the long round trip delay of the satellite portion of the path can be isolated from the portion of the connection that traverses the Internet. For short transactions, we found that the best performance requires TCP enhancements (T/TCP, 4KSS) that are not implemented in the current Internet– again leading us to consider split connection solutions. Given that one decides to split TCP connections at a satellite gateway, it is natural to ask what protocol (either modified TCP or an entirely new protocol) should be used over the satellite portion of the split connection. In the next chapter, we attempt to improve on our performance results even further by designing a satellite-optimized transport protocol that outperforms even the split TCP configuration described above in Section 4.3.

Chapter 5

Satellite Transport Protocol

In the previous chapter, we illustrated the performance advantage gained by splitting a TCP connection at a gateway within an access satellite network. Given such a split connection, however, there is no requirement to actually use TCP within the satellite network. Moreover, for connections completely internal to a satellite network, transport protocols other than TCP are possible. In this chapter, we describe a new transport protocol optimized for asymmetric satellite access networks. The main difference between this protocol and TCP is in the way data is acknowledged. The designers of TCP chose to use a steady stream of data acknowledgments as a pacing mechanism to clock out new data; the implicit design tradeoff was to simplify the protocol implementation at the cost of extra traffic in the network. For broadband geostationary satellite systems, bandwidth is at a premium– therefore, our choice is to reduce the traffic load on the backchannel as much as possible, at the possible expense of more complicated implementations. Fortunately, we can avoid significant increases in complexity by changing the basic data transfer mechanism of the protocol. Our protocol, which we dubbed the "Satellite Transport Protocol" (STP), outperforms TCP in satellite environments characterized by high bit error ratios, asymmetry, or widely varying round trip times. STP can be used in two ways: i) as the satellite portion of a split TCP connection, and ii) as a transport protocol for control and network management traffic within a satellite communications network. This chapter describes the overall protocol design, followed by simulation and experimental results.

5.1 Design Goals

In Section 2.1.3, we introduced the background material relating to the development of STP. In summary, STP is an outgrowth of the ATM-based link layer known as SSCOP. While SSCOP was mainly intended for networks with large bandwidth-delay products such as broadband wide-area ATM networks, many of the same design principles of that protocol help greatly in the satellite environment. In this section, we describe our design requirements for an optimized satellite transport protocol and discuss how we modified the core protocol mechanism of SSCOP to develop STP.

5.1.1 Basic Design

To understand STP, it is perhaps easiest to contrast its operation with that of TCP. Like TCP, STP provides a reliable, byte-oriented streaming data service to applications. We designed STP to offer the same API as does TCP, and to operate over an IP-based network. The transmitter sends variable-length packets to the receiver, storing the packets for potential retransmission until the receiver has acknowledged them. However, STP's automatic repeat request (ARQ) mechanism uses selective negative acknowledgments, rather than the positive acknowledgment method of TCP. Packets, not bytes, are numbered sequentially, and the STP transmitter retransmits only those specific packets that have been explicitly requested by the receiver. Unlike TCP, there are no retransmission timers associated with packets.

One of the main differences between STP and TCP, and one that offers an advantage for asymmetric networks, is the way in which the two protocols acknowledge data. TCP acknowledgments are data-driven; the TCP receiver typically sends an ACK for every other packet received. While this is beneficial for accelerating window growth upon connection startup, it results in a large amount of acknowledgment traffic when windows are large. In STP, the transmitter periodically requests the receiver to acknowledge all data that it has successfully received. Losses detected by the receiver are explicitly negatively acknowledged. The combination of these two strategies leads to low reverse channel bandwidth usage when losses are rare and to speedy recovery in the event of a loss.

Packet Types

STP has four basic packet types for data transfer (we ignore, for now, the additional packet types needed for connection setup and release). The *Sequenced Data (SD)* packet is simply a variable length segment of user data, together with a 24 bit sequence number and a checksum. SD packets that have not yet been acknowledged are stored in a buffer, along with a timestamp indicating the last time that they were sent to the receiver. No control data is included in the SD packets; instead, the transmitter and receiver exchange *POLL* and *STAT(us)* messages. Periodically, the transmitter sends a POLL packet to the receiver. This POLL packet contains a timestamp and the sequence number of the next in-sequence SD packet to be sent. The receiver responds to the POLL by issuing a STAT message which echoes the timestamp, includes the highest in-sequence packet to have been successfully received, and contains a list of all gaps in the sequence number space. The STAT message is similar in concept to a TCP selective acknowledgment, except that the STAT message reports the entire state of the receiver buffer (rather than the three most recent gaps in a SACK). Since each STAT message is a complete report of the state of the receiver, STP is robust to the loss of POLLs or STATs.

The fourth basic packet type is called a *USTAT (unsolicited STAT)* packet. USTATs are data-driven explicit negative acknowledgments, and are used by the receiver to immediately report gaps in the received sequence of packets without waiting for a POLL message to arrive. This allows the POLL and STAT exchange to be run at a low frequency (typically two or three per RTT when the RTT is large). In a network in which sequence integrity is guaranteed or highly likely, a USTAT can be sent upon any reception of a packet numbered beyond the next expected. If resequencing by the network is possible, USTATs can be delayed until there is a high probability that the missing packet was not reordered by the network. However, if the USTAT is sent too early there is only the small

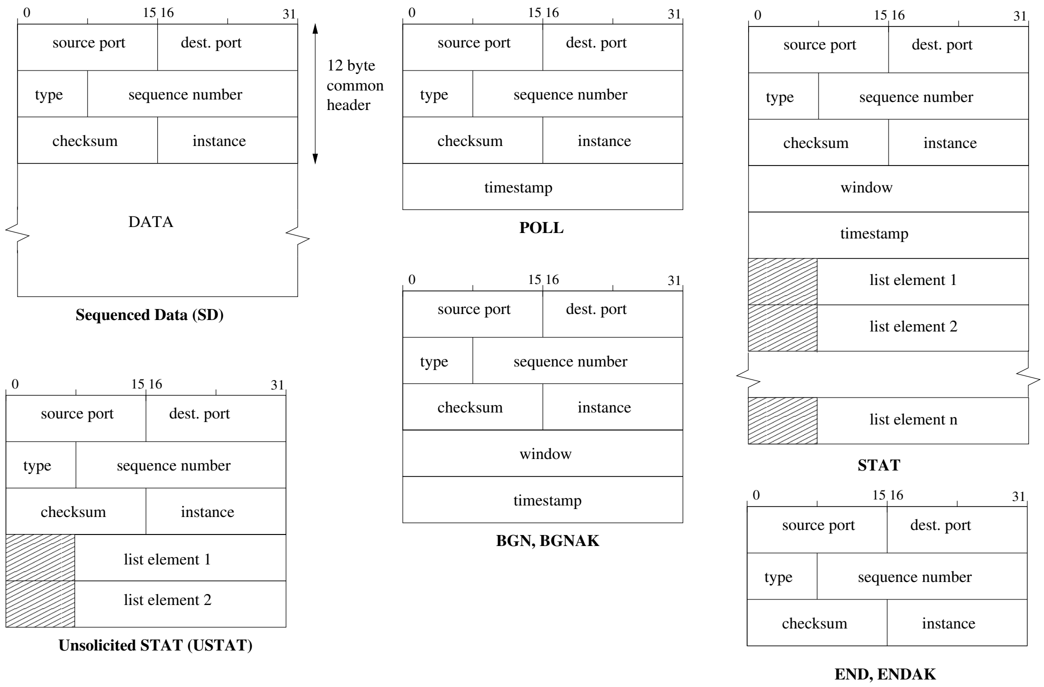


Figure 5.1: Illustration of the main packet formats in STP.

penalty of a redundant retransmission. USTATs are the primary form of negative acknowledgment, and STATs recover all second-order losses.

Figure 5.1 illustrates the key packet types used by STP. The fields are byte-aligned and the data portion of packets is 32-bit aligned. Above, we discussed the use of the SD, POLL, STAT, and USTAT PDUs. The BGN and BGNAK are used to open a connection, and the END and ENDAK are used to close a connection. Each packet contains a 12-byte common header that includes the source and destination port numbers (as in TCP), a type field, a 24-bit sequence number, a 16-bit checksum, and an instance number (to distinguish between different connections that may happen to use the same port numbers). Certain packets are permitted to be concatenated for the purpose of conserving the number of packets transmitted; for example, an SD and a POLL packet may be concatenated, in which case the type field is the logical "OR" of the SD and POLL values, and the POLL timestamp precedes the data.

Bulk Data Transfer Operation

The basic operation of STP can best be illustrated by an example. For simplicity, Figure 5.2 only illustrates one direction of data transfer and assumes that sequence integrity of transmissions is preserved. In the example, the transmitter sends a series of consecutively numbered packets. After packet (SD) #4 is sent, a POLL packet is sent (due to either the expiration of a POLL timer or a threshold on the number of new packets sent). The POLL tells the receiver that the next message to be sent is #5, so the receiver knows that it should have received packets 0 through 4. In this

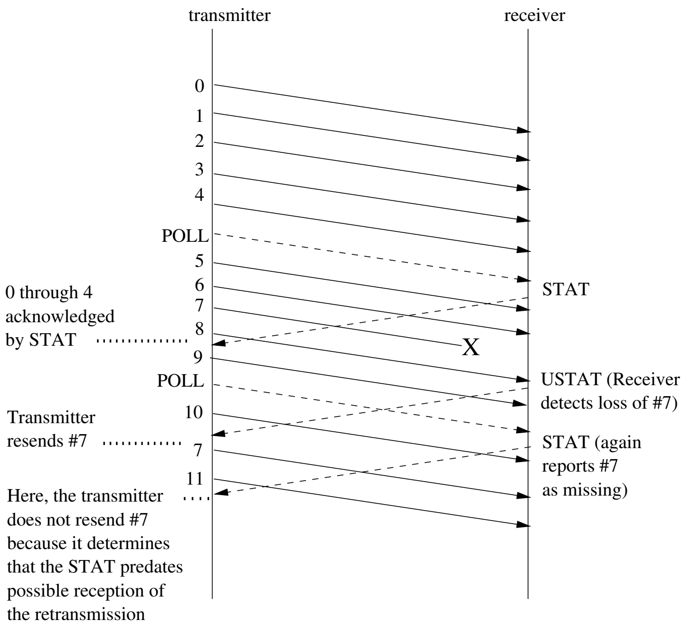


Figure 5.2: Basic STP operation for bulk data transfer.

case, since they have all been received, the receiver returns a STAT packet acknowledging all data up to and including packet #4. After sending the POLL, the transmitter continues with packets 5 through 9. However, packet #7 is lost. The receiver detects this loss upon receipt of packet #8 and immediately requests retransmission of #7 with a USTAT packet. Before this USTAT is received at the transmitter, the transmitter agains sends a POLL packet. Upon reception of the USTAT, the transmitter immediately resends #7, continues on with new data transmission, and then receives a STAT packet again reporting #7 as missing. However, the timestamp in the STAT packet allows the transmitter to determine that the retransmission has not yet had an opportunity to reach the receiver, thereby avoiding an unnecessary retransmission. If #7 had again been lost, the next STAT message would have stimulated a second retransmission.

The key to the performance of the protocol is the frequency with which the transmitter polls the receiver. If the bit error rate is high or the sender and/or receiver are using small windows (either due to small socket buffer sizes or a congestion window that has not yet opened up to a large value), it is advantageous to poll frequently, perhaps three or more times per round trip time (to most quickly recover from losses and open up the window). However, if the above conditions are not met, then it is safe to poll once per round trip time or less. This is because, under low loss conditions, the *USTAT* message provides the first-order recovery mechanism for losses. By polling infrequently, the best savings on the bandwidth usage on the return channel can be realized.

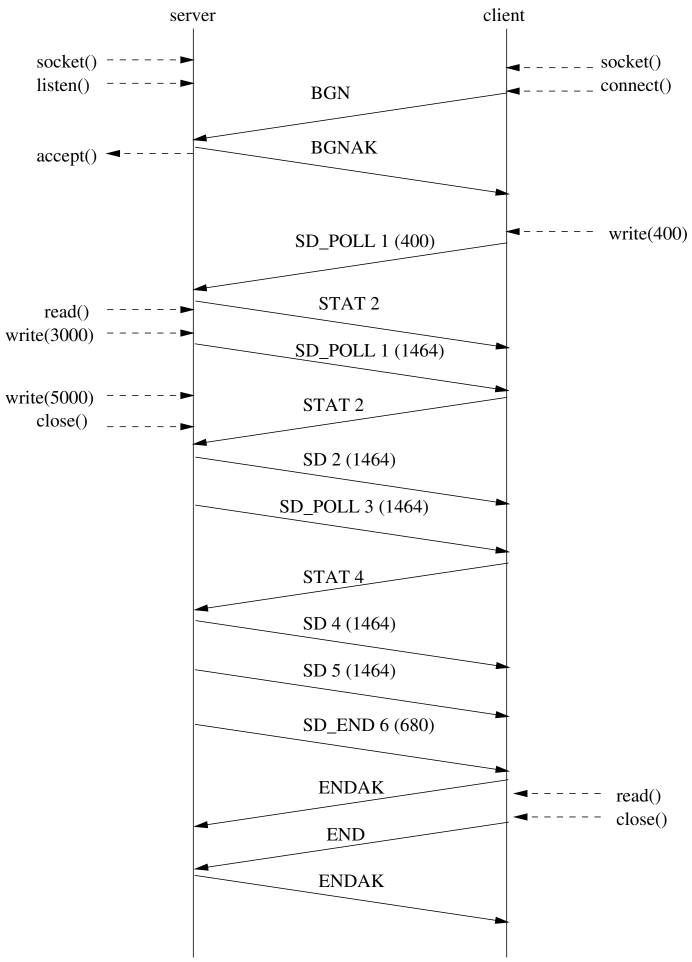


Figure 5.3: Basic STP operation for short transactions.

Short Data Transactions

Figure 5.3 illustrates a hypothetical data transaction in which the client (the initiator of the connection) writes 400 bytes of data to a server and receives an 8000 byte response. The example illustrates typical system calls that would be used by the application. The connection uses TCP-like window control, so that the congestion window builds by one packet for each packet acknowledged. The server is listening on a particular port, and the client connects to that port by issuing a `connect()` system call, which causes STP to send a BGN packet. This exchange of BGN and BGNAK coordinates the sequence numbers to be used by both sides and establishes the window sizes in each direction. The client then writes 400 bytes to the socket, which stimulates an SD to be sent to the other side. In this case, the client is configured to POLL with the first burst of data, so the actual packet sent is a concatenation of an SD with a POLL. The server responds to the POLL by issuing a STAT, reporting the next sequence number (#2) that the server expects

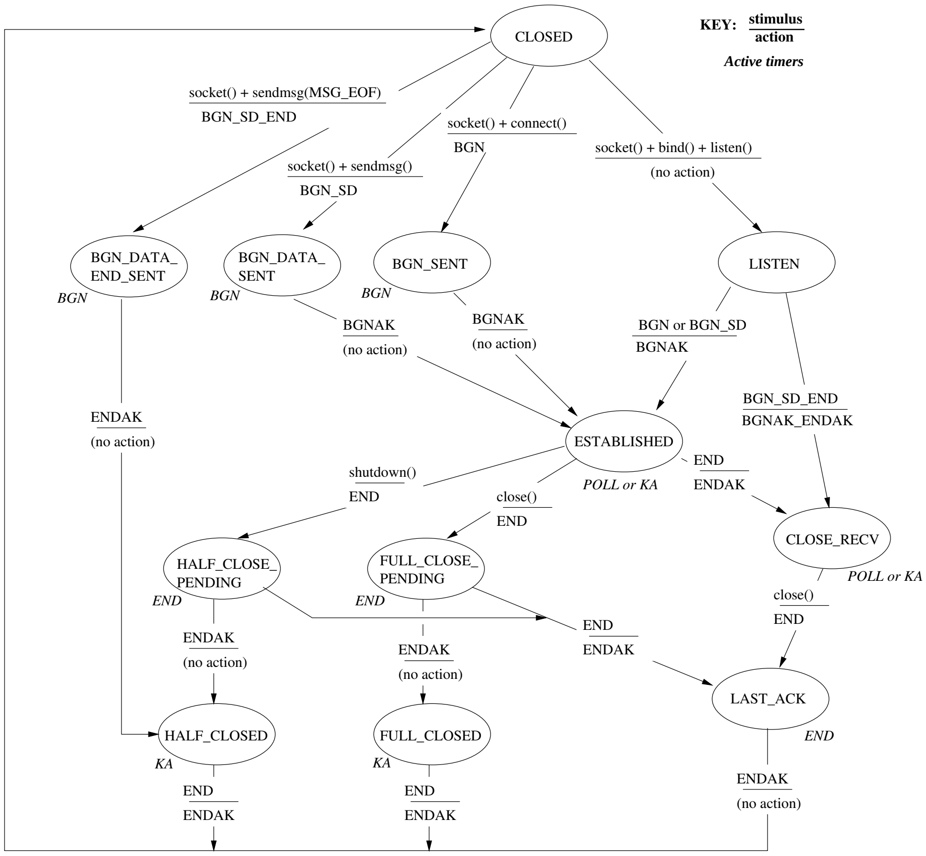


Figure 5.4: Overview of STP state transitions.

to receive. The server then writes 8000 bytes of data to the socket, and it requires six packets to complete the transfer. Upon successful completion of the data transfer, both sides exchange END messages to close down their respective halves of the connection.

State Machine

Figure 5.4 illustrates the state machine associated with typical STP connections, as well as the common stimuli and responses that effect the transitions (packets are labeled in capital letters, while socket calls are listed in small case). This diagram illustrates a number of additional packet types (BGN, BGNAK, END, ENDAK) used for connection control. In much the same way that a TCP segment can be overloaded to perform more than one function (for example, a TCP *SYN* flag may be combined with a TCP *ACK* flag), these STP connection control packets can be combined with data (e.g., BGN_SD packet for fast connection opening) or with other connection control packets (e.g., BGNAK_ENDAK). Using the basic client-server model of data communications, a connection is instantiated by one side (server) putting a socket into the *LISTEN* state, and the other side (client) initiating the connection by sending a BGN packet. In STP, since we allow the protocol to skip the initial handshake waiting for a BGNAK, we define a number of states (BGN_SENT, BGN_DATA_SENT, and BGN_DATA_END_SENT) that capture the state of the client before it receives a BGNAK. The normal data transfer state is the *ESTABLISHED* state. Applications can either close both directions of communications by issuing a `close()` system call, or can shutdown the write side of the connection by issuing a `shutdown()` system call. The remainder of the states illustrated involve the closing of the connection by both sides.

Also listed in Figure 5.4 are the timers active in each state. STP has four main timers: the *BGN* and *END* timers used to guarantee receipt of the respective packet types of that name, and the *POLL* and *KEEPALIVE* timers, active during data transfer. If data is outstanding, the *POLL* timer will be running; otherwise, the *KEEPALIVE* timer will (very infrequently) poll the peer to ensure that the connection is still up.

5.1.2 Our Protocol Modifications

In Section 5.1.1 above, we described the core data transfer mechanism of STP, which follows closely the basic operation of the SSCOP protocol. However, SSCOP cannot operate over connectionless networks for a number of reasons. In [59], we have described how STP builds on the basic SSCOP design through several protocol additions. In this section, we highlight three of the most important differences between STP and SSCOP; namely, the addition of a hybrid window/rate congestion control mechanism, a fast connection start that avoids unnecessary handshaking, and the piggybacking of a POLL message on a data segment.

Congestion Control

The SSCOP specification included no flow or congestion control mechanism. For data transfer in a distributed packet-switched network, some mechanism is needed to adapt to changing network conditions. The TCP congestion control mechanism, in which each connection adjusts its sending rate based upon implicit feedback from the network (the dropping of packets), has two main problems when applied to STP. First, TCP relies on a property known as ACK-clocking: the arrival of an ACK triggers departures of new packets, which helps to smooth out the transmission of packets to a degree of burstiness that the network can accept. In STP, since ACKs (STATs) are only sent periodically, another technique to smoothly send data is required. Second, it is unlikely that congestion control in a satellite network will operate in a completely distributed manner with no bandwidth constraints. The solution that we adopted is based on modifications to TCP's flow control. In particular, we designed a mechanism that adapts to the amount of rate control desired in the network.

We start with the basic TCP algorithm and describe operation when there is no network rate control. The protocol maintains a congestion window, which is set to an initial number of segments and which is guaranteed never to exceed the window offered by the receiver. The protocol then undergoes slow start by increasing its congestion window by one packet for each ACKed packet; i.e., it follows rules for TCP slow start. The congestion avoidance algorithm is also similar. However, slow start is never reentered since there are no timeouts. The protocol increases its window or enables new retransmissions only upon receipt of a STAT or USTAT message. Therefore, at every reception of a STAT or USTAT, the transmitter counts how many transmissions are enabled, and schedules them to be sent uniformly over the estimated RTT of the connection. The estimated RTT is computed from the timestamp of a received STAT and the current time, and we perform a low pass filtering across several samples to obtain the *delayed send* timer.

Next, consider the case where a minimum and maximum sending rate are imposed by the network. The technique described above easily generalizes to this case by constraining the allowable values of the timeout interval for the *delayed send* timer. If a hard upper bound exists on the sending rate, retransmissions can be counted among the packets scheduled to be sent. The required granularity of the *delayed send* timer depends on the granularity of the rates enforced by the network and on the access speed of the network.[1] Additionally, the granularity of the timer may be relaxed to reduce the overhead of interrupts in the protocol processing. In our implementation, we used timers with a granularity of 10 ms, which corresponds to a timer "tick" in BSD-derived systems. There exists a tradeoff between the granularity of the delayed send timer and the smoothness with which data is submitted to the network; as the data transfer rate grows, more and more packets are emitted at once when this timer expires. In our experiments, the speed of our network interface cards was more of a limiting factor than this timer granularity.

Handshake Avoidance

SSCOP originally had hooks placed in the protocol specification to allow the standardization of a "fast connection start," but the mechanism was never completed. We added this feature to STP as follows. Data is allowed to be sent with a BGN packet, in anticipation of a connection acceptance by the peer host. In addition, depending on the initial value of the window (if window control is being used in a network), SD and POLL segments may also be sent before an acknowledgment of the BGN packet is received. Therefore, both the T/TCP reduced handshaking and policies such as the 4KSS may easily be implemented. Connection sequence numbers help to distinguish different connections in much the same way as in T/TCP. Although the use of T/TCP connection handshaking in the wide-area Internet is deprecated due to denial-of-service concerns, in a network where the satellite service provider controls both of the endpoints, such concerns are mitigated.

Piggybacked POLL

Finally, a fundamental design principle of SSCOP was the separation of data and control flow. SSCOP was designed for an ATM environment, in which a POLL message fits into a single cell and occupies a small amount of switch buffering. For this reason, POLL messages or ACK information is not piggybacked on SD segments, although the mechanism was seriously considered during SSCOP development. In the Internet, however, most IP routers place buffer limits on the number of packets received, not on the size of such packets, so a POLL segment actually takes up as much buffer space as full data segment. Because of this, we noticed in our initial experiments an effective reduction of usable buffer space along the forward data path. Therefore, we experimented with piggybacking POLL messages on outgoing data segments if both types of segments were scheduled to be sent around the same time. This modification helped greatly, reducing the number of standalone POLL segments by about an order of magnitude, leading to substantial improvement at the small cost of defining an additional packet type. Moreover, piggyback POLLs can be used to efficiently and quickly trigger STAT responses when the windows are too small to justify periodic POLLing, such as the initial period of data transfer on a congestion-controlled connection.

[1] It may be possible to relax the required granularity of this timer if the MAC layer also performs traffic smoothing.

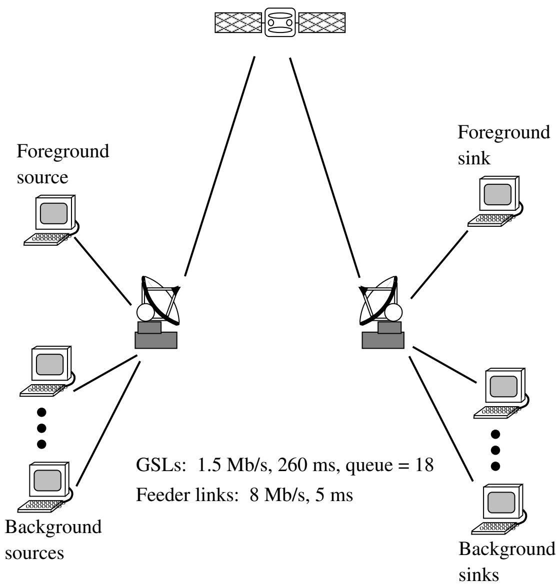


Figure 5.5: Simulation configuration for GEO topology. Simulations involved measuring the performance of file transfers between "foreground" hosts against background Web-like traffic generated by "background" hosts.

5.2 Simulation Results

We implemented the data transfer procedures of STP in the UCB/LBNL network simulator *ns*, described in Chapter 3. We were interested in comparing the performance of STP and TCP by examining the performance of persistent connections (i.e., long file transfers) over simulated satellite network topologies. In this section, we describe our simulations, present the results of a comparison of STP and TCP performance with respect to bulk data transfers, and present performance results for STP connections in a high BER environment.

5.2.1 Simulation Configuration

Topologies

Figures 5.5 and 5.6 illustrates the two simulated topologies with which we experimented. The first topology is configured to emulate a 1.5 Mb/s channel over a geosynchronous (GEO) satellite channel with a one way delay of 260 ms. Counting the propagation delays of the feeder links, the total RTT experienced by a user is 532 ms, excluding queueing and transmission delays. The queue sizes were set to approximately 10 percent of the outgoing line rate (a commonly used rule-of-thumb for queue sizes in practice). The second topology illustrates a hypothetical low-earth-orbit (LEO) configuration. The access links to the satellite have a one-way propagation delay of 5 ms and a bit rate of 2 Mb/s, and the intersatellite links have a propagation delay of 10 ms and a bit rate of 100Mb/s. The topology is similar to transcontinental connections across proposed broadband LEO

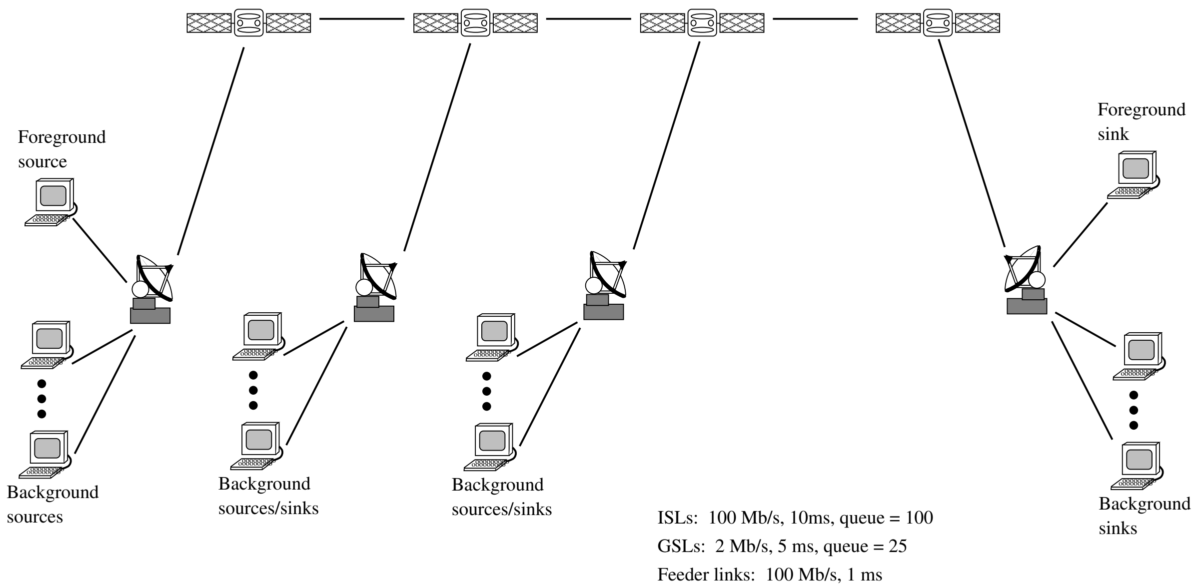


Figure 5.6: Simulation configuration for LEO topology. Simulations involved measuring the performance of file transfers between "foreground" hosts, across four LEO satellite hops, against background Web-like traffic injected at various points in the topology.

systems.

TCP Configuration

We tested two TCP variants: TCP with selective acknowledgments (TCP SACK) and TCP Reno, which corresponds to a current reference TCP implementation. For the TCP simulation configuration, we used *ns* defaults for all parameters except for the offered window size, which we opened up to a large value to avoid artificially constraining the sender, and the coarse timeout interval of 500 ms, which is set by default in *ns* to the non-standard value of 100 ms. Since delayed acknowledgments are standard practice in current implementations (to reduce the reverse channel bandwidth), we configured the TCP sinks to send delayed acknowledgments (typically, an ACK is sent for every two segments). For STP, we set the threshold on duplicate acknowledgments before a USTAT is sent to three (as in TCP), and we configured the STP sender to send roughly three polls per RTT, since such a polling interval has been found to offer high performance for SSCOP [58].

Congestion Control

To permit a fair comparison of the basic protocol performance, we implemented a window-based congestion control policy in STP identical to that of TCP; namely, additive window increases of one packet per RTT and a multiplicative decrease by one half during the congestion avoidance phase, and slow start. For both the GEO and the LEO topologies, we added background Web-like traffic which occupied, on average, about 10% of the bottleneck links. This traffic emulated actual WWW usage based on empirical distributions of actual traffic traces. The purpose was not to heavily congest the network (the greedy behavior of the congestion avoidance mechanisms of the foreground connections guaranteed that congestion would frequently occur) but to add variability to the simulation runs to break up any phase effects. In the GEO topology, the background traffic was sent in the same direction as the foreground transfer, creating an occasional bottleneck at the queue at the ingress of the satellite network. In the LEO topology, the background traffic was sourced from multiple ground stations in different spot beams, creating an occasional bottleneck at either the ingress or the egress of the satellite network. In some situations, discussed below, we balanced the traffic load in each direction by creating a persistent TCP SACK connection in the reverse direction along with the same amount of simulated WWW traffic, so as to cause periodic congestion losses in the reverse path as well.

5.2.2 Bulk Transfer Performance of STP

Tables 5.1 and 5.2 present the results of an average of 200 simulation runs, each 60 seconds long, over the GEO and LEO topologies, respectively. We chose 200 runs of each configuration in order to get the small confidence intervals listed in the tables. We compared the performance of STP, TCP SACK, and TCP Reno, first with foreground traffic only and then with bidirectional traffic. Since no bit errors were introduced on the links, all losses were due to congestive losses induced by the congestion avoidance mechanisms. We used a fixed packet size of 1000 bytes (including TCP/IP or STP/IP overhead), corresponding to the *ns* default. The performance is compared in terms of forward throughput achieved ("goodput"), forward bandwidth efficiency (ratio of goodput to total TCP/IP data transferred in the forward direction) and reverse channel bandwidth usage. We observed the following:

- TCP SACK and STP both significantly outperform TCP Reno in the forward direction, especially over a long delay path. This is because (as described in Section 4.2.3 above) both TCP SACK and STP are able to recover multiple losses in a window's worth of data within a single RTT.
- Both TCP SACK and TCP Reno use much more bandwidth than STP on the reverse channel for returning ACKs. For all of the simulations averaged together, STP required roughly 5 Kb/s, while TCP Reno needed 17 Kb/s and TCP SACK used 21 Kb/s.
- For one way traffic, STP outperforms TCP SACK in the GEO case, in terms of throughput in the forward direction. This is largely because of TCP SACK's delayed acknowledgments, which cause the congestion window to open up more slowly than if every segment were acknowledged. If the TCP receiver were to acknowledge every segment, TCP would slightly outperform STP since STP acknowledgments are slightly delayed relative to the times that packets arrive. Doing so, however, would double the usage of the reverse channel bandwidth, which is already high with delayed acknowledgments. Another benefit to STP in a high bandwidth-delay product environment [2] with high losses is that it reports the complete state of the receiver with every STAT.

[2] "Bandwidth-delay product" refers to the total number of bits that can be "in the pipe" during one round trip time of the connection.

	Throughput (Kb/s)	95% conf. int. (+/-)	Fwd. efficiency	95% conf. int. (+/-)	Reverse bw. (Kb/s)	95% conf. int. (+/-)
One way traffic, GEO						
STP	672.3	20.8	0.938	0.001	2.2	0.03
TCP SACK	594.9	20.5	0.945	0.001	13.2	0.3
TCP Reno	296.5	15.2	0.928	0.003	6.7	0.3
Two way traffic, GEO						
STP	595.7	20.2	0.933	0.008	2.0	0.03
TCP SACK	388.4	13.9	0.939	0.001	8.9	0.3
TCP Reno	259.2	12.0	0.922	0.003	6.0	0.2

Table 5.1: Performance comparison of STP, TCP SACK, and TCP Reno over the simulated GEO topology.

- In the LEO case, TCP SACK slightly outperforms STP in the forward direction. We believe that this is due to STP slowing down its sending rate in response to a queueing backlog (which increases its RTT). This is actually a nice self-regulating property of the congestion control algorithm, since the sending rate is inversely proportional to the measured RTT of the connection.

- STP outperforms TCP SACK to a greater extent when there is two way traffic in the system. We believe that this is due to the effect of ACK compression, which disrupts TCP's self clocking behavior. STP is relatively insensitive to these effects, and in fact it seems that much of the reduction in its performance is due to the presence of the reverse (TCP SACK) connection's acknowledgments in its forward path buffers, effectively decreasing its usable buffer space.

- TCP SACK and STP have comparable performance in terms of forward bandwidth efficiency. Although STP has less per-packet overhead, the overhead of the three POLL messages per RTT must be amortized across data; therefore, the efficiency improves as the throughput improves.

- The difference in performance between the protocols was reduced in the LEO case as compared to the GEO case. There is less of an advantage in using TCP SACK instead of TCP Reno when the RTT is smaller.

In summary, when using the standard TCP flow control policies in identical simulated satellite environments, STP generally outperformed TCP SACK and TCP Reno in terms of throughput while using much less bandwidth in the reverse channel. We did not explore possible improvements via further fine-tuning of the protocols.

5.2.3 STP Performance in a High BER Environment

We next examined the performance of STP in a rate-controlled environment in which the transmitter was only constrained by a maximum sending rate, but in which the BER was varied

	Throughput (Kb/s)	95% conf. int. (+/-)	Fwd. efficiency	95% conf. int. (+/-)	Reverse bw. (Kb/s)	95% conf. int. (+/-)
One way traffic, LEO						
STP	1668.2	13.8	0.961	0.001	8.5	0.05
TCP SACK	1715.1	14.5	0.957	0.001	37.4	0.3
TCP Reno	1552.9	11.6	0.957	0.002	33.5	0.2
Two way traffic, LEO						
STP	1440.5	13.8	0.960	0.001	8.0	0.04
TCP SACK	1154.2	12.6	0.958	0.001	25.5	0.3
TCP Reno	975.4	15.5	0.957	0.001	21.3	0.3

Table 5.2: Performance comparison of STP, TCP SACK, and TCP Reno over the simulated LEO topology.

from 10^{-4} to 10^{-7}. The modification to STP's flow control to permit this operation is simple. The delayed send timer can be regularly scheduled to expire at the rate at which packets are allowed into the network. If retransmissions are queued, they are sent with highest priority in the scheduled slot; otherwise, a new data packet is sent. We did not impose rate control on the traffic in the reverse channel. Figure 5.7 illustrates results for a 1 Mb/s connection (1 Mb/s available transport bandwidth), again using the GEO topology shown in Figure 5.5, but for which bit errors were randomly inserted on the link. Again, we configured the IP packet sizes to be 1000 bytes for data traffic. Since the STP/IP overhead per packet is 32 bytes, the usable throughput is constrained to be 968 Kb/s at best. Figure 5.7 illustrates that the selective retransmission mechanism provides high efficiency even as the BER degrades substantially, and that the reverse channel bandwidth also rises as the BER increases (due to the increased use of the *USTAT* message), as shown in Figure 5.8. As the BER degrades further, good performance can be maintained by using smaller packets (since with a BER of 10^{-4}, the packet loss rate is 55% with 1000 byte packets). We did not compare STP with TCP in this case since TCP has no facility for explicit rate control.

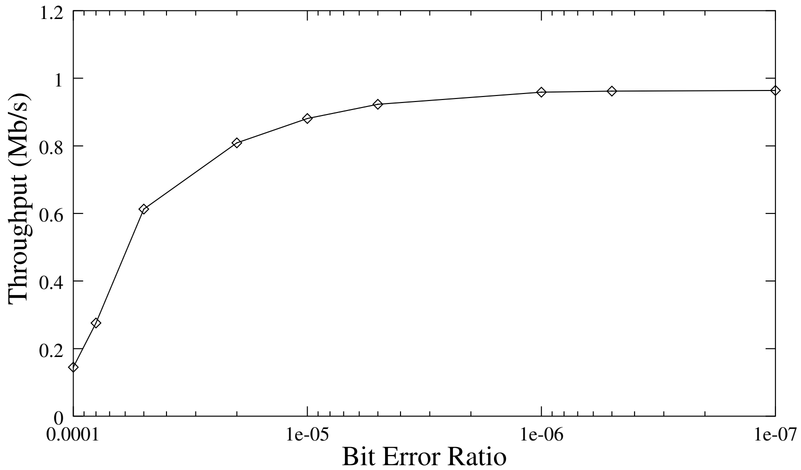


Figure 5.7: Simulation results of the forward throughput performance of STP on a 1 Mb/s channel with a variable BER.

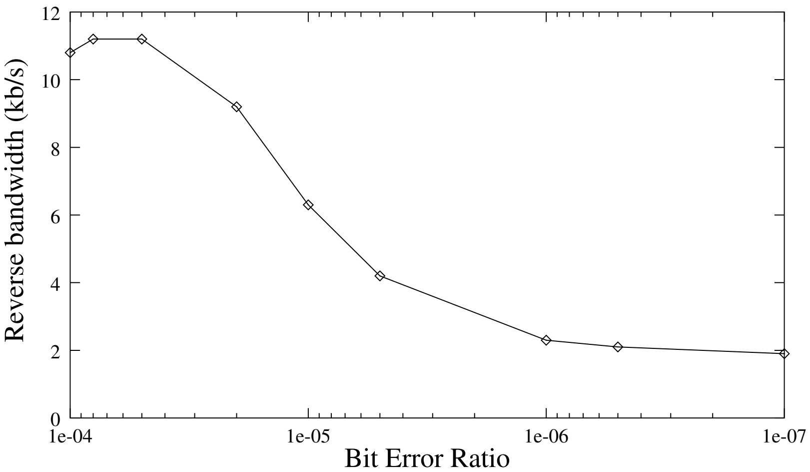


Figure 5.8: Reverse channel bandwidth requred for a large STP file transfer as a function of BER (simulation results corresponding to Figure 5.7).

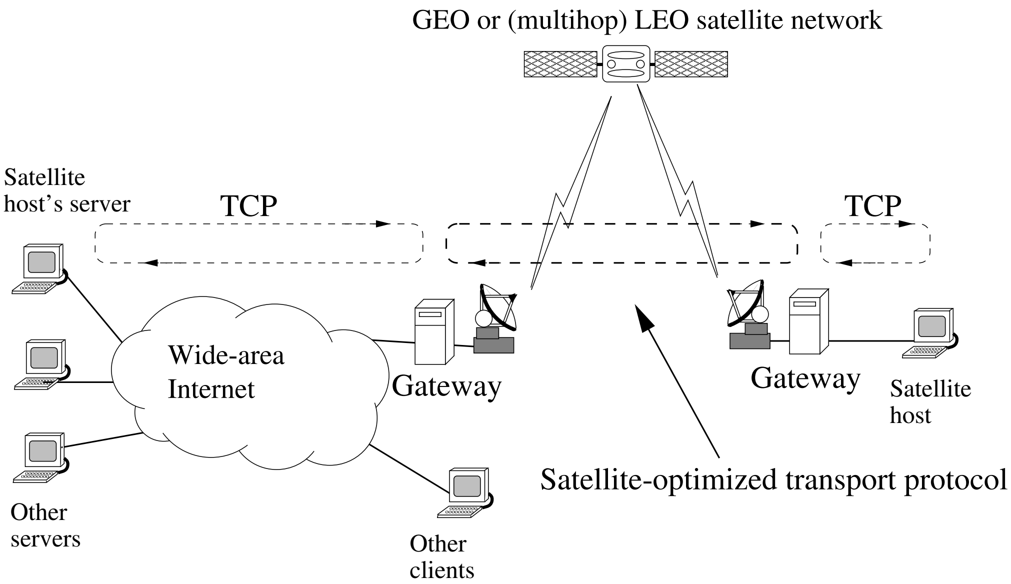


Figure 5.9: Future satellite networking topology in which a satellite-based host communicates with a server in the Internet through a satellite protocol gateway.

5.3 Experimental Results

In the previous chapter (Sections 4.2 and 4.3), we described a series of network experiments aimed at characterizing the performance of end-to-end and split TCP connections in a satellite environment. In this section, we continue that series of experiments with a look at the performance of STP under similar conditions. The reader is referred back to those sections for a description of the methodology used in those and the following experiments. This overall progression from analysis to simulation to experiment is part of our basic research methodology introduced in Chapter 3.

Figure 5.10 plots the difference in file-transfer performance between split STP and split TCP (SACK plus NewReno) when there is competing short-delay traffic in the wide-area Internet. For reference, we reproduce Figure 5.9 above, previously illustrated as Figure 4.22 in Chapter 4, as a description of the networking topology used in these experiments. To permit a fair comparison between the two protocols, we implemented in STP the identical slow start, congestion avoidance, and exponential backoff algorithms found in TCP (the main difference is that STP uses byte counting, rather than ACK counting, to build its congestion window). In practice, depending on the bandwidth management employed in the satellite network, other congestion control mechanisms may perform better. The TCP data is reproduced from Figure 4.23, discussed previously in Section 4.3. Figure 5.10 illustrates that STP achieves approximately the same forward throughput as TCP, because the forward throughput is primarily governed by the congestion avoidance policy. Again we found that for long RTTs, STP's throughput is slightly smaller than TCP's because the STP congestion control mechanism, in smoothing the transmission of new data over the estimated RTT of the connection, effectively makes the control loop longer. We found the bandwidth overhead in the forward direc-

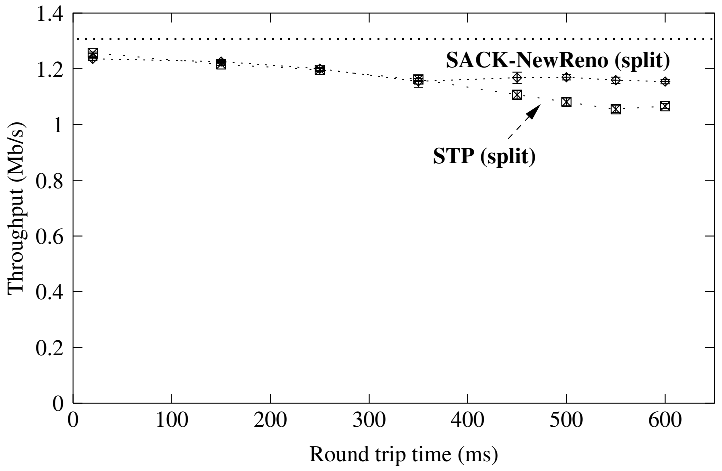


Figure 5.10: Comparison of forward throughput performance of split TCP and split STP. For fair comparison, both TCP and STP used identical congestion control policies.

tion to be slightly lower for STP than for TCP, since the per-segment overhead reduction in STP data packets more than compensates for the POLL traffic. In the reverse direction, illustrated in Figure 5.11, STP uses much less bandwidth than TCP. STP's reverse channel usage linearly decreases with the RTT, since we configured the polling frequency to be three times the estimated RTT of the connection. Under this configuration, therefore, the amount of return bandwidth required is roughly independent of the forward throughput. In both Figures 5.10 and 5.11, 95% confidence intervals are plotted as error bars on the data, although the error bars are difficult to notice because they are very small.

We also examined the performance of STP versus that of TCP and T/TCP for short transfers. There is an inherent tradeoff between the user-perceived latency of the connection and the amount of bandwidth used to return ACKs. To complete the connection as fast as possible, data must be ACKed regularly and quickly, but this leads to more packets sent on the reverse channel. For long file transfers when the window and buffers are large, data can be ACKed less frequently. In our STP design, when the congestion window was low (below some threshold value), we configured the STP transmitter to send the last packet of every data burst with a piggybacked POLL, and to suppress timer-driven POLL transmissions. When the window grew above the threshold, POLL transmissions were regularly scheduled. This led to frequent STAT messages (one per arrived data burst) at the beginning of connections, but also reduced the relative amount of POLL traffic in the forward direction and kept the latency low. The overall STP behavior is similar to that of T/TCP for short transfers, while for long transfers when the window is large, the reverse channel utilization is greatly reduced. In our experiments, we found that a window threshold value of approximately 10 times the segment size worked well.

Table 5.3 illustrates the relative performance of TCP, T/TCP, and STP in terms of both the average latency and average number of packets, when driven by a traffic generator based on the HTTP trace distributions of [79]. The data were collected from experiments on a local network in

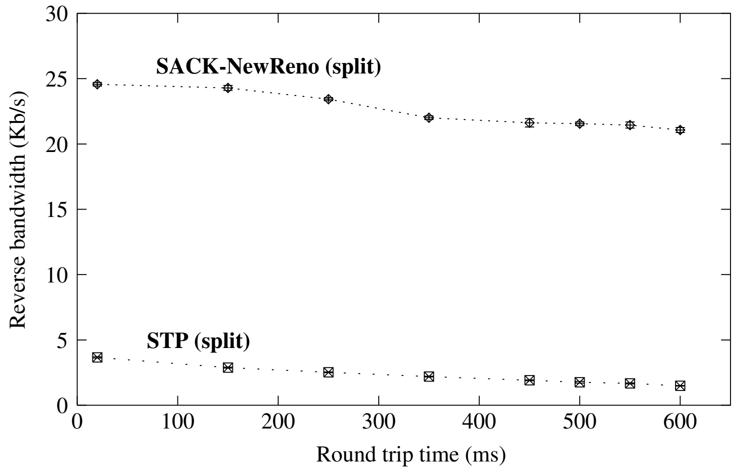


Figure 5.11: Comparison of reverse channel usage of split TCP and split STP, for the forward transfers illustrated in Figure 5.10.

which the device drivers of the hosts were configured to produce a RTT of 600 ms, and STP, T/TCP, and TCP implemented standard TCP congestion avoidance with an initial congestion window size of one. Each table entry is the average latency of 1000 independent runs with the given protocol. We observed that STP's performance was better than TCP's but slightly worse than T/TCP's, both in terms of average latency and average number of packets per connection. The reason that the number of packets required for an STP connection is higher than for T/TCP is because, as discussed above, for small values of the congestion window, the protocol "ACKs" (i.e., sends a STAT) more frequently than every other packet, to reduce latency. However, the reason that STP's latency is not consequently lower than T/TCP's is due to its traffic smoothing mechanism: packets eligible for transmission are not sent immediately but rather paced out over the estimated RTT. In short, this data illustrates yet another tradeoff in protocol design, this time between smoothing bursty data and reducing latency. For small transfers, STP behavior could be further tuned to more closely approximate T/TCP operation, although we did not experiment with this approach. Empirically, we have observed that Web browsers using STP over GEO-like emulated channels continue to operate with good performance for reverse channels with bandwidth as low as 1 Kb/s, while such a constrained backchannel renders conventional TCP unusable.

In addition to laboratory testing, we experimented with the performance of both TCP SACK-NewReno and STP in commercial networks. For these experiments, we we used the DirecPC satellite system and Ricochet packet radio networks (introduced in Section 3.3), both of which are high latency networks with asymmetric paths. The RTT over the DirecPC system and back through the Internet was roughly 375 ms over 12 hops. The base RTTs in the Ricochet system were roughly 350 ms, but because of the deep packet queues in the radio network, latencies could range as high as 15 seconds. In addition, 15 network hops were required between the wireless gateway and the machine at Berkeley. Table 5.4 provides experimental results from several file transfers over these systems. Both networks rely on the wide-area Internet for at least a portion of the traversed path.

	Avg. latency (s)	Avg. packets
TCP	2.0	12.3
T/TCP	1.4	7.3
STP	1.5	9.1

Table 5.3: Comparison of TCP, T/TCP, and STP performance for HTTP traffic. The results are averages of 1000 HTTP transfers, where the traffic generated was drawn from an empirical distribution based on traces described in [27].

	STP (Kb/s)	TCP (Kb/s)
DirecPC fwd.	480	370
DirecPC rev.	2.8	7.6
Ricochet fwd.	28.1	27.1
Ricochet rev.	0.6	0.9

Table 5.4: Results of file transfer experiments over the DirecPC DBS system and Ricochet packet radio network. The throughputs listed are the averages of 25 file transfers. The file sizes were 1 MB for DirecPC and 100 KB for Ricochet.

For the DirecPC network, the average forward throughput performance for STP is better than that of TCP, and STP also uses less than half of the reverse bandwidth required for TCP. Similarly, STP does better on average in the packet radio network. The packet buffers in this case are very deep, and STP's sending behavior was so smooth that we often observed extremely long queueing delays (15 seconds) built up in the network before STP took a loss due to buffer overflow. This behavior suggests that STP, when used in low bandwidth networks, should back off its window growth upon detection of lengthening RTTs. In addition, the fact that some transport protocols can induce this much queueing delay argues for the deployment of router-based congestion control mechanisms such as Random Early Detection (RED) [44] in packet radio networks [85].

5.4 Summary

In this chapter we have described the design and performance of STP, a satellite-optimized transport protocol that compares favorably with satellite-optimized TCP for certain environments. STP inherently incorporates many of the features that have been proposed or adopted as TCP options for improved satellite performance. STP also allows for the use of rate-based congestion control, and because the reverse bandwidth usage is roughly constant, STP is well matched to satellite networks which allocate fixed amounts of uplink bandwidth to users (such as those using TDMA multiple access). One drawback of using STP with a heterogeneous client population is the requirement that either the end host implement STP or the satellite network interface (such as a set-top box) convert the protocol back to TCP. However, many of the changes proposed as satellite-friendly TCP options also require client-side changes; particularly those dealing with TCP asymmetry. Finally, since STP provides the same reliable byte-stream service as does TCP, STP can be used internally within a satellite network by applications that are written to use TCP.

We experimented with simulation models and UNIX kernel implementations of STP. A key requirement of our tests was that the protocol performance be measured in an environment containing other competing connections sharing portions of the same network path. Under the same window-based congestion control policy as used in TCP, we found that STP data transfers could obtain roughly the same forward throughput as similar TCP transfers while using up to an order of magnitude fewer bytes in the reverse direction; the difference was most pronounced for long file transfers. For short Web-like transfers, we found that STP could achieve a performance better than conventional TCP and approaching that of TCP for Transactions. Our simulation results also highlighted that STP is less sensitive to congestion on the reverse path, and illustrated good throughput performance in environments characterized by BERs as low as 10^{-4}.

This chapter concludes our investigation of transport protocol issues in a GEO satellite environment. In the next chapter, we turn our attention to the problem of designing packet routing protocols for LEO satellite networks.

Chapter 6

Packet routing for LEO networks

In this chapter, we study the core packet routing design problem for LEO networks, focusing in particular on the potential for simplifying routing by using geographic-based network addresses. In Section 6.1, we first provide a high-level overview of those characteristics of LEO networks that are relevant to packet routing, and discuss why and in what sense LEO packet routing is an interesting problem. Next, in Section 6.2, we describe the simulation environment that we constructed and justify our choices for the various simulation parameters we needed to configure. Our simulation models revealed some fundamental delay performance characteristics of LEO networks, which we illustrate in Section 6.3. These benchmark results are interesting in their own right but are also useful as a reference for comparing with our later results. In the remainder of the chapter, we focus on the potential benefit of embedding geographic location information within the network addresses of user terminals. After first introducing a cellular geometry in Section 6.4, we describe in Section 6.5 our attempts to construct a distributed routing protocol that makes packet forwarding decisions based on such geographic information. Finally, in Section 6.6 we examine the benefit of using geographic-based addresses in a network that uses centralized routing.

Throughout this chapter, we make frequent reference to the Iridium and (proposed) Teledesic satellite constellations (first introduced in Chapter 1 and described in more detail in Chapter 2), and use these topologies as the basis for our packet routing research. Iridium and Teledesic are just two examples of a particular class of LEO satellite constellation– other constellations designs are possible. Nevertheless, rather than explore the entire design space of possible satellite constellations, we have chosen to focus on the Iridium and Teledesic constellation topologies as examples of feasible LEO systems because they represent two designs that have been considered commercially viable from a frequency management (interference), orbital deployment, and economic perspective. The Iridium and Teledesic systems are described in [75] and [130], respectively.

6.1 Why is LEO Packet Routing an Interesting Problem?

LEO networks are an interesting type of mobile network in that the nodes are moving rapidly with respect to the slow moving or fixed user nodes, causing frequent link handoffs. Despite the highly time-varying nature of the network topology, there are some simplifying properties. First, most of the topology changes of the satellite mesh itself (aside from equipment failures) can be predicted in advance. Second, the graph topology is somewhat regular and dense, leading to a multiplicity of similar routes to most destinations. Both of these simplifying properties can potentially be exploited by routing algorithms as we explore later in this chapter. Nevertheless, when compared with routing protocol design for terrestrially-based packet networks, there are several fundamentally different design objectives that complicate the design. First, we make the assumption that satellite hardware will continue to be mass and power constrained, thereby limiting the amount of on-board memory and processing. Although it is true that advances in electronics technologies will continue to make memory cheaper and less power-consuming in future years, the satellite payload is still a very power-constrained network node, with as much power as possible allocated to signal transmission. We therefore seek routing algorithms that are memory efficient and are not computationally intensive. Second, conservation of link bandwidth, particularly on the links between ground and satellites, is important because a loss of capacity for user traffic on these expensive links leads to a loss in revenue or higher service costs. Third, economic factors limit the number of satellites that can be deployed in a constellation, and consequently cause the coverage footprints of satellites to be stretched thin. For instance, the Iridium system, which uses 66 satellites, requires an elevation mask of 8.2 degrees at the edge of each satellite's coverage footprint [105], which is not very high above the horizon and could potentially lead to shadowing problems. Systems that guarantee double coverage, such as one described in [142] that leads to a Manhattan network topology, do not seem likely to be built.

For the above reasons, operating traditional distributed routing protocols and using traditional means of hierarchy are not likely to provide the best performance. Distance vector protocols have well known convergence problems in time-varying topologies, and while some of the shortcomings have been addressed over the years (such as the DUAL protocol [47]), the improvements come at a cost of complicating the protocol. Link state protocols converge much more rapidly upon topology changes, at the expense of a large amount of message traffic, higher protocol complexity, and routing computational overhead. Of course, either distance vector or link state protocols can be made to work in LEO satellite systems; the point is that because such protocols do not capitalize on the simplifying aspects of LEO network properties, one is likely to do better with more specialized protocols. Furthermore, area hierarchies as used in the current Internet are not as appropriate for a highly regular network topology with nodes under a single administrative control– where does one draw the area boundaries? Finally, a centralized routing system may be preferred in this environment for a number of reasons discussed later in this chapter.

In summary, the major challenge in the design of packet routing algorithms for LEO networks is coping with both a time-varying topology and constraints on key system resources, while trying to capitalize on certain (simplifying) properties of the network topology. We have relied heavily on simulations of LEO networks to explore this problem. Therefore, before presenting any results we will first describe our simulation model and the key parameters used therein. Next, we will illustrate some fundamental delay performance results in LEO constellations before focusing our attention in the remainder of the chapter on the following question: How can geographic location information about network nodes be used to simplify packet routing?

6.2 Simulation Model and Key Parameters

In this section, we describe in more detail how we modelled the behavior of the LEO constellations patterned after the Iridium and Teledesic constellations. LEO systems are complicated

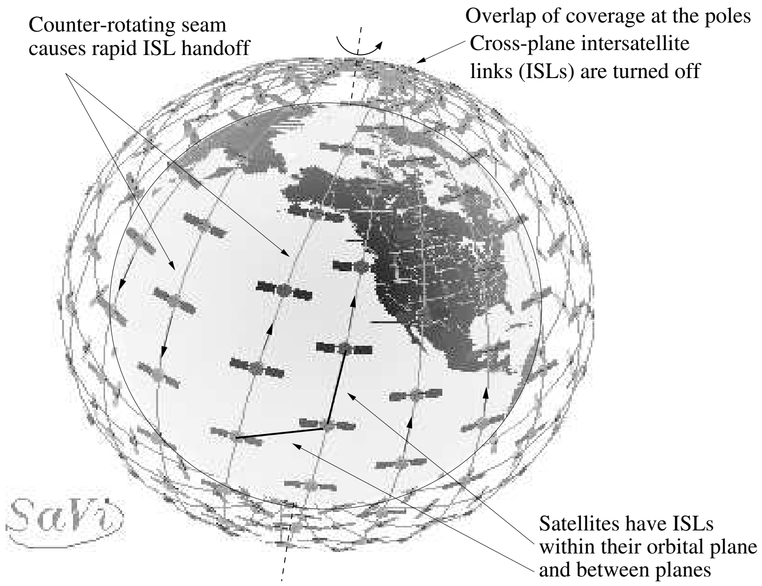


Figure 6.1: Example of a polar-orbiting satellite constellation (figure reproduced from Section 2.2).

to model because of the sheer complexity of the system and because many details of such systems are strongly hardware dependent and have not been discussed in the literature. We discuss in this section the many choices for simulation parameters that we made, why we chose the values that we did, and whether or not our results are highly sensitive to these choices.

Recall that in Section 2.2, we described some of the most important features of LEO constellations, and in Section 3.2, we introduced the simulation environment that we have constructed to study LEO routing. We preface the rest of the material in this chapter by briefly reviewing the key points discussed there. Figure 6.1 illustrates a possible configuration for a polar-orbiting LEO constellation (modelled after the Teledesic 288 satellite configuration). The satellites orbit the Earth in fixed circular planes while the Earth rotates underneath. Satellites communicate with one another using intersatellite communication links (ISLs). As the figure indicates, three types of ISLs can exist: interplane, intraplane, and cross-seam ISLs. Table 6.1 again summarizes key constellation parameters for both the Teledesic and Iridium systems. We should emphasize here that while Iridium has been designed for circuit switching at very low bit rates, in this chapter we are considering the use of the Iridium constellation design in a hypothetical broadband packet switching network. Also, as of this writing, the parameters describing the Teledesic constellation are likely to change.

Figure 3.2 in Chapter 3 illustrated the key components of our extensions to the ns simulator to enable it for LEO routing studies. We use a spherical coordinate system centered on the Earth's center, and inserted a *position object* in each node that describes the node's position as a function of time in this coordinate system. Links between nodes in the simulator have a dynamically varying propagation delay that is based on the instantaneous distance between two nodes– whenever a packet must be sent, both nodes at the endpoints are queried for their current position. Nodes also contain a *handoff monitor*, described in more detail below, that check for opportunities

	Iridium	**Teledesic**
Altitude	780 km	1375 km
Planes	6	12
Satellites per plane	11	24
Orbit inclination (deg)	86.4	84.7
Interplane separation (deg)	31.6	15
Seam separation (deg)	22	15
Elevation mask (deg)	8.2	40
Max. ISLs per satellite	4	8
Cross-seam ISLs	no	yes

Table 6.1: Key constellation parameters for the Iridium and Teledesic systems. Both systems are examples of polar orbiting constellations.

to enable, disable, and hand off links between nodes, and nodes also contain a *routing agent* for use in distributed routing (we also implemented a centralized routing genie for studying centralized routing).

Beyond the main topological parameters listed in Table 6.1, the following additional details help to more fully describe our models. With respect to the constellation configuration, we made the following two minor simplifications. First, we did not model the minimal orbital eccentricity found in the topologies; our orbits were purely circular. Second, we did not model any drifts in nominal satellite position with respect to the original constellation design, assuming instead that, where possible, the placement of satellites in adjacent orbits will be staggered so as to maximize ground coverage (i.e., in Teledesic, where satellites are nominally spaced at intervals of 15 degrees in each orbit, we offset the position of satellites in adjacent planes by 7.5 degrees). While such a staggering is optimal, it is unclear whether satellite operators will expend the fuel necessary to maintain this phasing (both Iridium and Teledesic plan to hold constant the relative positions of satellites within a particular orbit, but in the Teledesic system there are no guarantees of maintaining any phasing between satellites in different planes).

Iridium satellites are connected to their four nearest neighbors: two satellites in the same orbital plane, and one each in the adjacent planes. Satellites along the counter-rotating seam only have three active ISLs if cross-seam ISLs are turned off (in our simulator, we could also selectively enable cross-seam ISLs for the Iridium topology but generally experimented without them). It is only the cross-seam ISLs that require satellite handoffs, since the intraplane ISLs are static links, and the interplane ISLs only need to be deactivated and reactivated near the poles. The Teledesic system connects to eight nearest neighbors: the four closest satellites within the same plane, one satellite each in the two adjacent planes, and one satellite each two planes away. At the counter-rotating seam, only one ISL is active across the seam and the other is used to acquire the next satellite to be handed off to. We configured the GSLs to be full duplex links at 1.5 Mb/s (i.e., we considered a broadband version of the Iridium system), and the ISLs to be 155 Mb/s for Teledesic and 25 Mb/s for Iridium. The exact values of these bandwidths were not important since we were only considering a minimal amount of traffic. Figures 6.2 and 6.3 illustrate snapshots of satellite positions and active intersatellite links for Iridium and Teledesic, respectively. The plots were generated by

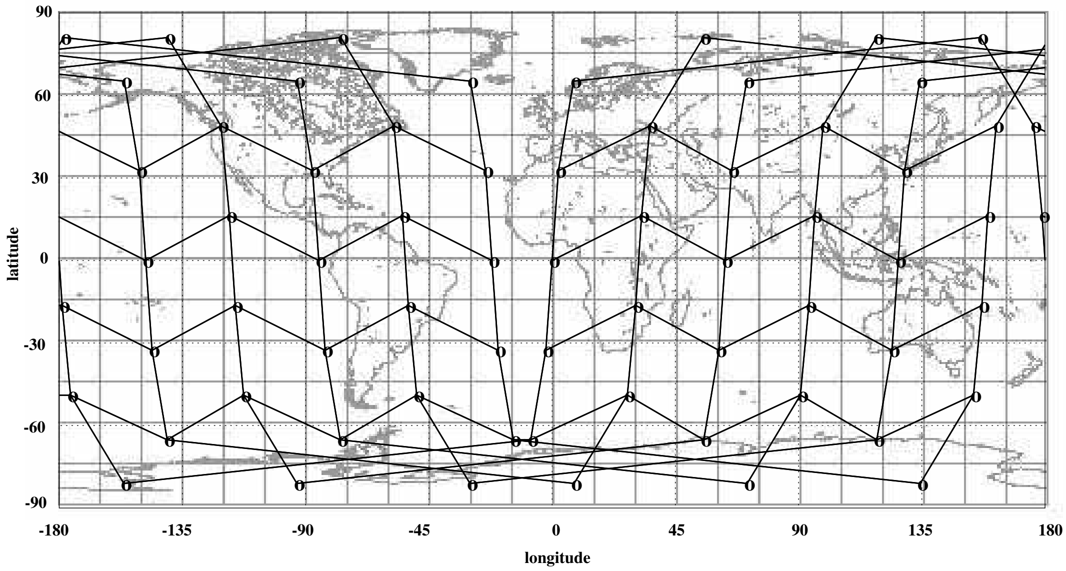


Figure 6.2: Snapshot of the Iridium constellation, illustrating active ISLs.

outputting satellite and link position information and then superimposing the data on a rectangular map projection obtained from the Xerox PARC Map Viewer. Note in the Iridium topology the lack of cross-seam ISLs and the absence of interplane ISLs in the high latitudes.

Various policies for performing handoffs between network nodes are possible– the exact choice of handoff mechanism is sensitive to the satellite hardware capabilities, and Iridium and Teledesic have not publicly revealed their techniques. We implemented both asynchronous and synchronous handoffs as described above in Section 2.2.1. Asynchronous handoffs between ground terminals and satellites work as follows. Each terminal periodically checks whether the satellite that is serving it has dropped below the elevation mask for the terminal. In our simulations, we performed this check every ten seconds, on average (we added a random dither to the timeout interval so that it would vary between five and fifteen seconds); we did not regard the exact value of this timeout parameter as being critical, although too small of a choice leads to slower simulations. Upon checking, if the terminal discovers that the current satellite has dropped below the elevation mask, the terminal searches for another satellite that is above the mask and connects to the first such one found. The technique of synchronous handoffs assumes that topology changes occur only at certain times– our simulator can also be configured such that all nodes perform a topology check synchronously (as we explore later in this chapter).

We next describe two simulation parameters that are highly dependent on the antenna steering capabilities. Interplane ISLs are deactivated whenever one or both satellites are above a given latitude threshold. We typically set this threshold to 70 degrees, since analysis by Werner indicates that Iridium should be able to maintain ISLs between 60 degrees north and south latitude [140], and a Motorola patent by Rahnema claims that an Iridium-like constellation is able to keep these links active up to 68 degrees latitude [116]. Although we conjecture that the denser Teledesic

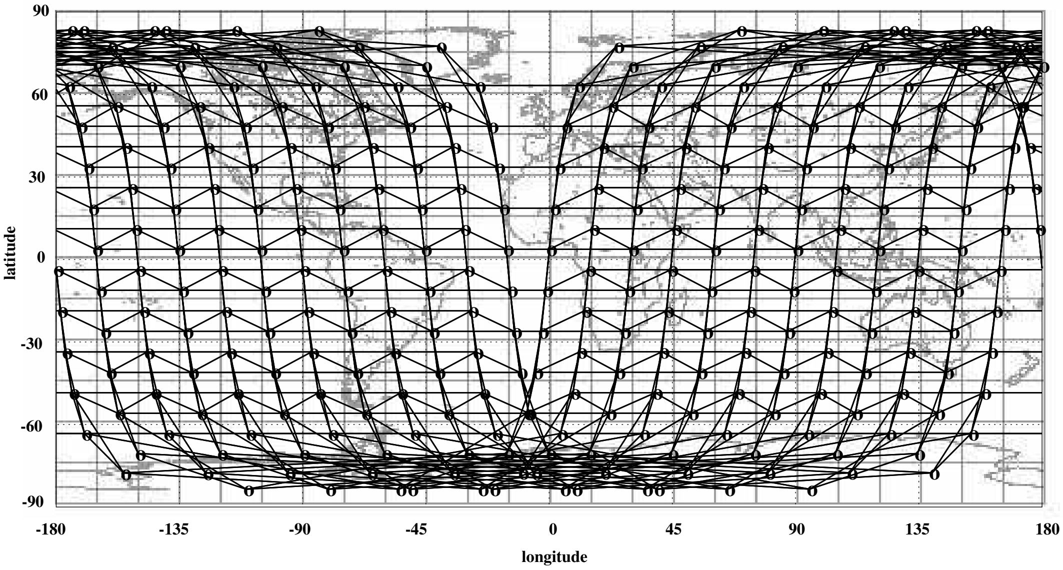


Figure 6.3: Snapshot of the Teledesic constellation, illustrating active ISLs.

constellation may be able to steer these beams beyond a 70 degree latitude, we have no evidence to support this. Handoff agents on board the satellites monitor for this occurrence as well (again, we check every ten seconds on average). Finally, cross-seam ISLs cannot be maintained near the points where the counter-rotating planes intersect; in our simulations, we deactivated these ISLs whenever the satellites were within eight degrees of longitude of one another. More information in the public domain about the antenna steering capabilities of ISLs is needed to make these parameter guesses more accurate. We will have more to say about these particular parameters when we discuss geographic-based distributed routing, but in general, we found that our results were not highly sensitive to these two parameters.

Since our studies were focused on fundamental routing and propagation delay performance measurements, we simplified our simulations (and dramatically improved simulation runtime) by not modelling additional delays due to multiple access contention, framing, and link layer protocols, nor did we consider queueing delays in the network due to heavily loaded links. We also did not model or experiment with link outages or errors due to terrain or sun outages, propagation impairments, or thermal noise. Our rationale for these simplifications was that, while investigating the potential for network load balancing through routing is a good candidate for future research, our simulations on the fundamental routing properties of LEO networks did not require the level of detail that would have resulted from modeling all of the parameters listed above. Nevertheless, the simulator allows for such additional models to be inserted for future research.

In summary, we have described a number of system parameters that have implications on routing architectures. We will elaborate more on the implication of some of these parameters later in this chapter.

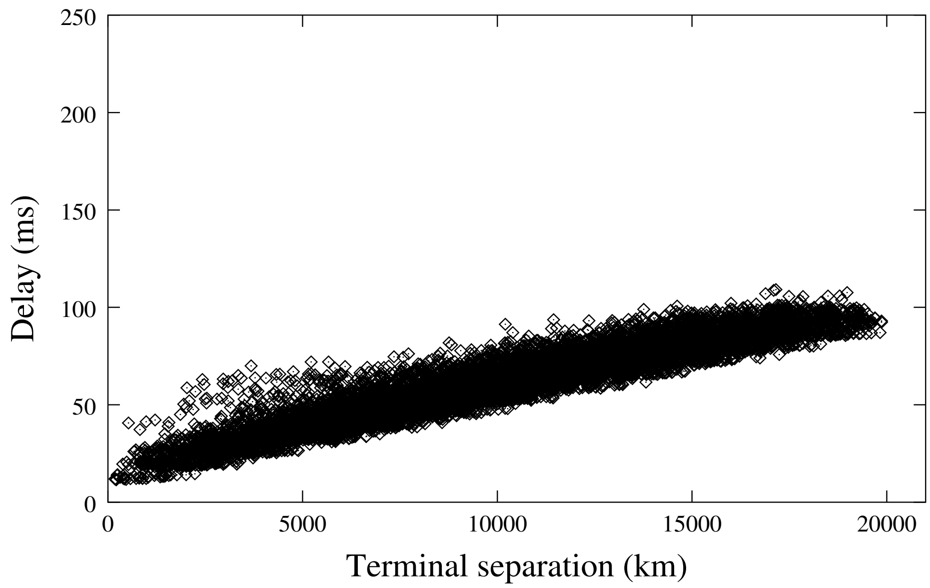


Figure 6.4: Scatter plot of the one-way delay experienced by 10,000 different pings between random locations on the Earth's surface, when global min-delay shortest path routing is used (Teledesic constellation configuration).

6.3 Basic Performance Results

The basic packet delay performance of modern LEO satellite constellations has never been thoroughly described in the literature. In this section, we quantify typical delay profiles that might be seen by users of future LEO networks. The results are useful in understanding the fundamental performance characteristics of such networks, and will also serve as benchmarks for our later evaluation of geographic routing.

6.3.1 Delay Profiles

One of the advantages of LEO systems over GEO satellites is the reduction in propagation delay between the Earth and satellite. Although the end-to-end latency can often be reduced from a quarter of a second to tens of milliseconds by using a LEO system, the delay in a LEO system is inherently variable. Our first experiments with our LEO network simulator were designed to study this delay variability.

Figure 6.4 is a scatter plot of the end-to-end delay experienced by 10,000 different single packet exchanges ("pings") using the Teledesic system. This simulation was designed to illustrate the range of end-to-end delays that users of these systems might experience. In the simulation, which ran for 20,000 seconds of simulation time, we repeated the following steps every two seconds. We first selected two points at random on the Earth's surface, and instantiated a link between each terminal and the first eligible satellite found (a satellite was considered "eligible" if it was above the terminal's elevation mask). We then configured one of the terminals to send a packet

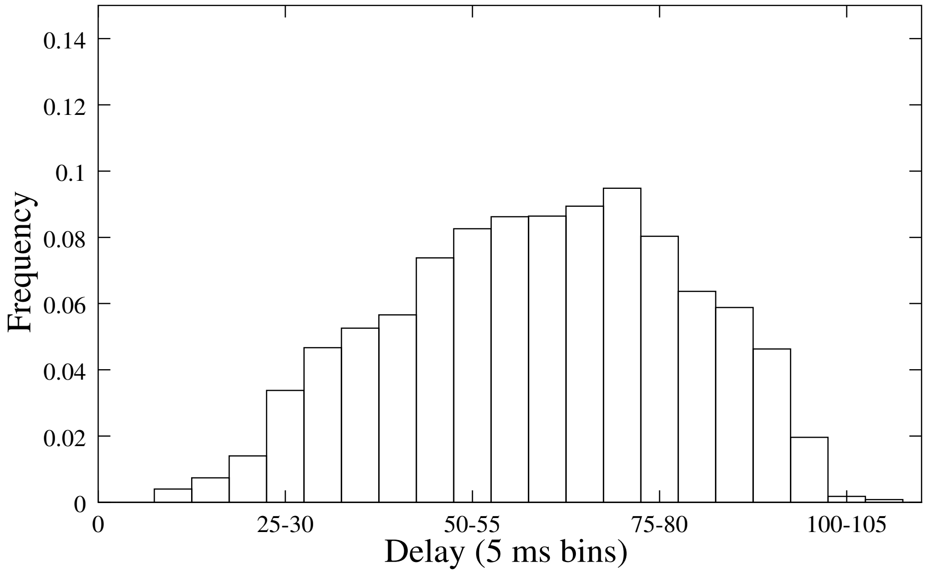


Figure 6.5: Histogram of values corresponding to Figure 6.4. Fewer than 1% of the delays exceeded 100 ms.

to the other, and measured the one-way delay. The LEO system used a centralized shortest-path routing algorithm based on minimization of the current propagation delay of each link– the routes were centrally computed and instantaneously loaded into each node in the simulator. Although this method of routing violates the speed of light limitation, it represents an upper bound on the achievable performance of a routing algorithm designed to obtain shortest paths. The distance plotted is the great circle distance between the two terminals. The figure illustrates that the end-to-end propagation delay in the Teledesic system is usually below 100 ms if shortest path routes can be found (fewer than 1% of our data points exceeded 100 ms, as illustrated by the histogram shown in Figure 6.5). Also, independent of the distance between the two terminals, a user may encounter an end-to-end delay that can differ by roughly 30 ms, depending on the particular configuration of the satellite constellation. This performance represents a lower bound on the achievable delay and delay variability that can be provided by a LEO satellite network.

The above routes were determined by considering the instantaneous propagation delays. We obtain slightly different, suboptimal results if we compute shortest paths based on minimizing the hop count, rather than propagation delays. In this case, as illustrated in Figure 6.6 for an identical set of terminal locations as plotted in Figure 6.4, the performance has the same lower bound but there is a bit more spread and some outliers. We will explore the difference between these two routing metrics a little later in this section.

Finally, Figure 6.7 illustrates a similar delay scatter plot for the Iridium constellation, were it to include cross-seam ISLs (we have included them here for comparison purposes). The delay performance of this constellation is similar to that of Teledesic (Figure 6.6) in terms of the lower bound, but the Iridium constellation exhibits higher variability due to the sparser satellite

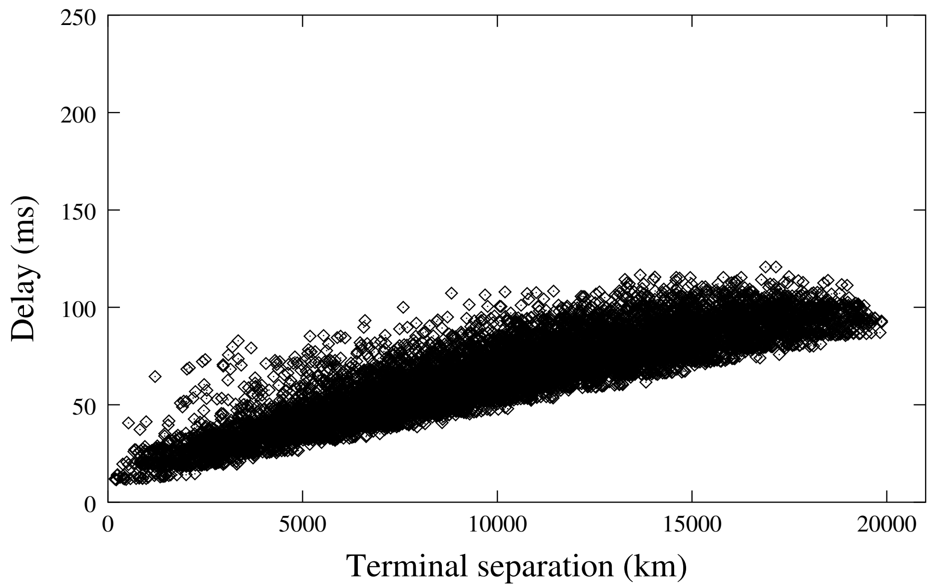


Figure 6.6: Scatter plot of the one-way delay experienced by 10,000 different pings between random locations on the Earth's surface, when global min-hop shortest path routing is used (Teledesic constellation configuration).

coverage.

Another way to observe the delay variability is to examine plots of a single session over a long period of time. Figure 6.8 plots end-to-end delay performance between a terminal located in New York and one in San Francisco over the course of one day. The data points are the delay experienced by a packet sent every 60 seconds. The end-to-end delay varies over a range of roughly 23-60 ms. Over an 11,000 second timespan beginning at time 57,600, the delay is noticeably increased. Even though the Teledesic constellation that we have considered uses cross-seam ISLs, there are certain instances in the mid-latitudes where they cannot be easily maintained; we will discuss this phenomenon in greater detail in Section 6.5. At a smaller timescale (Figure 6.9), it can be seen that the delay changes slowly as the satellites move with respect to one another, while handoffs somewhere along the route cause a step change in the delay of up to 8 ms. Such changes may cause packet reordering within the network. Although we did not experiment with the performance of TCP connections over such paths, first-order calculations suggest that the amount of packet reordering due to these delay changes should not trigger false fast TCP retransmissions for low to modest transmission rates.[1]

A similar plot between the same two terminals for the Iridium constellation is more interesting (Figure 6.10). Since Iridium does not employ cross-seam ISLs, whenever the seam lies between the two endpoints (which happens twice daily), the packets must be routed over the poles, causing a large increase in delay. Moreover, if a session is active across this seam at the critical handoff, the step increase or decrease in latency will be around 60 ms, which can cause a large

[1]For a 1 Mb/s session with 500 byte packets, a 8 ms delay decrease could cause at most 2 packets to be reordered.

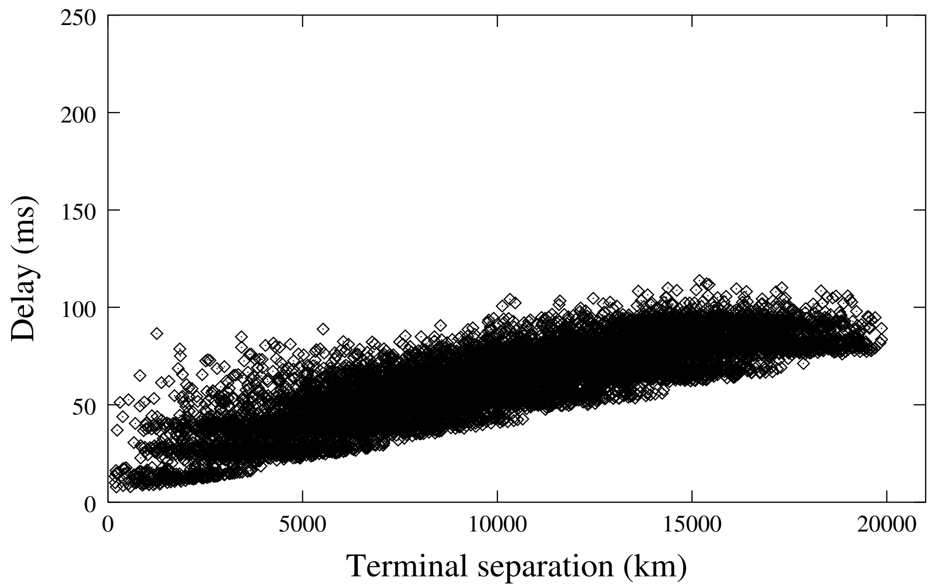


Figure 6.7: Scatter plot of the delay experienced by 10,000 different pings, when global min-delay shortest path routing is used (Iridium constellation).

amount of packet reordering. This kind of delay variability is inherent in a constellation that does not use cross-seam ISLs, and in the case of Iridium, the step change can be as large as 90 ms. Even without the increased delay at the counter-rotating seam (presuming that cross-seam ISLs are possible in such a constellation), Iridium exhibits much more delay variability than Teledesic, with the delay varying from 20 to 75 ms in much larger discrete steps. This is a direct consequence of having fewer satellites in the constellation, since every routing change that results in a different number of satellite hops also changes the path length by a significant amount. For example, in Figure 6.10, the cluster of points around 20 ms is due to the path only traversing two satellite hops, while the cluster of points around 80 ms results from certain instances in time when five satellite hops are required.

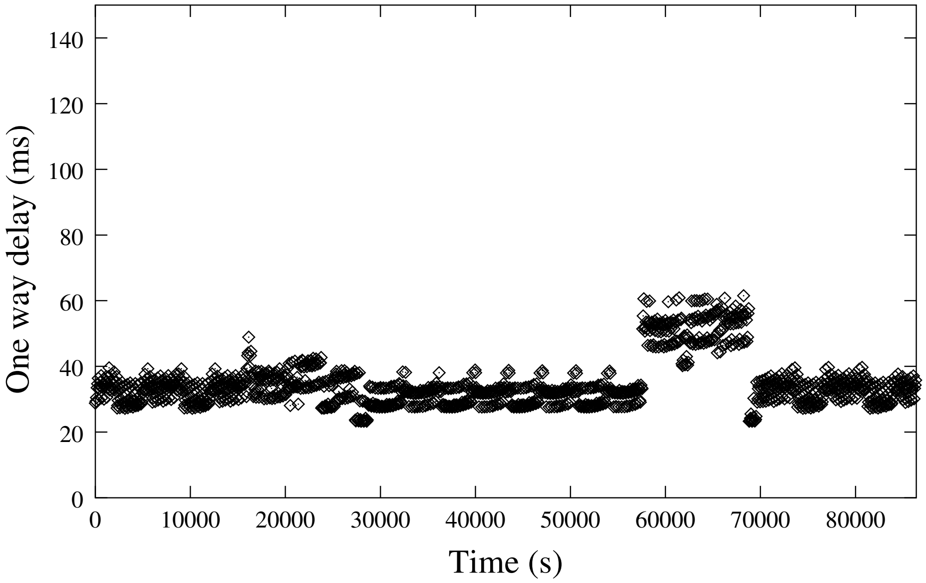


Figure 6.8: Delay variation between New York and San Francisco over the course of one day, for the Teledesic constellation.

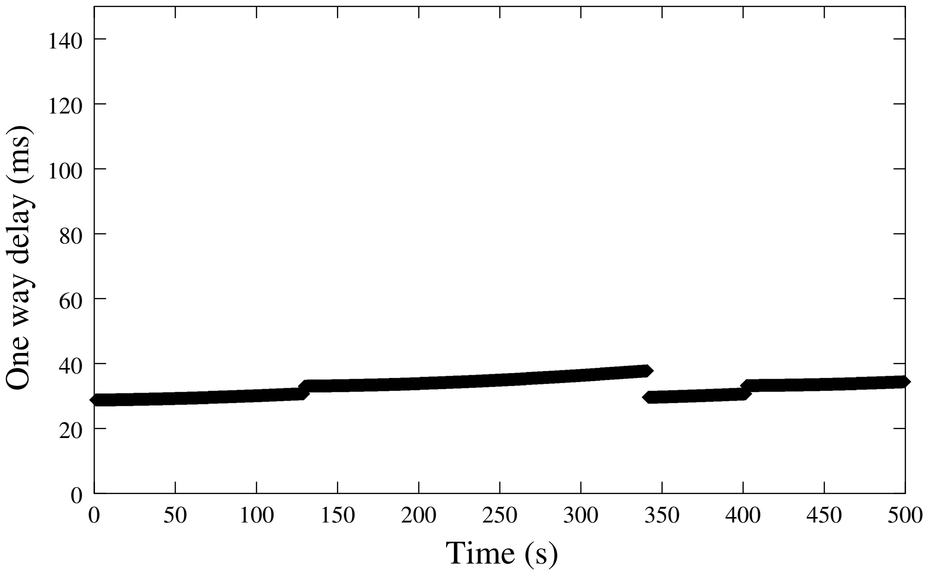


Figure 6.9: A view of the previous plot at a smaller timescale. End-to-end delays are characterized by slow variations over tens of seconds punctuated by step increases and decreases in the delay of generally no more than 8 ms.

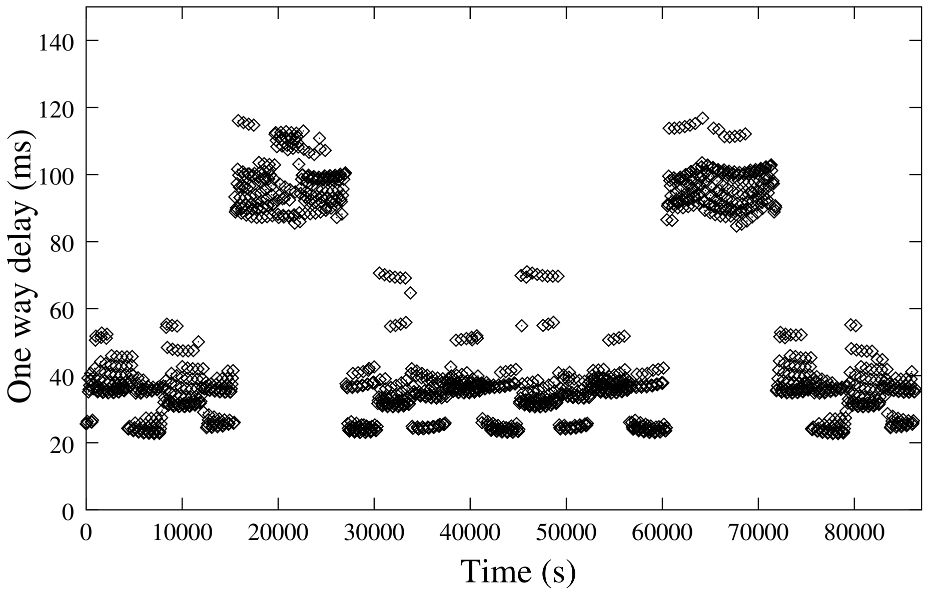


Figure 6.10: Delay variation between New York and San Francisco over the course of one day, for the Iridium constellation without cross-seam ISLs.

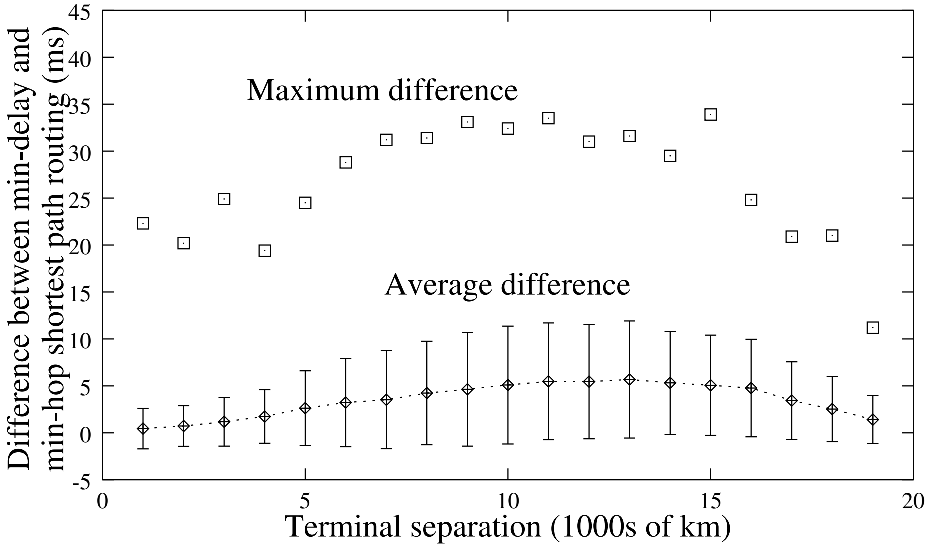


Figure 6.11: Average and maximum delay difference between min-hop and min-delay shortest path routing, as a function of the great-circle distance between terminals (Teledesic constellation).

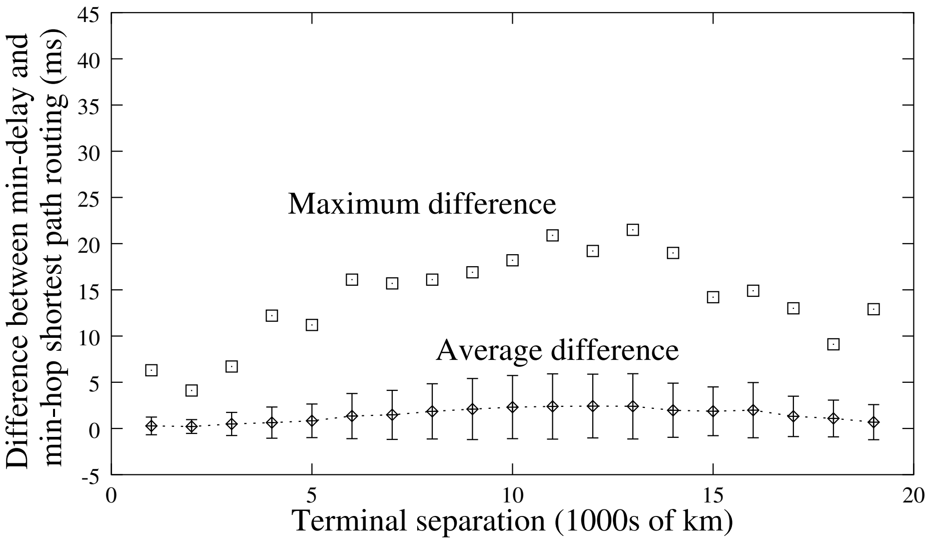


Figure 6.12: Average and maximum delay difference between min-hop and min-delay shortest path routing, as a function of the great-circle distance between terminals (Iridium constellation).

6.3.2 Routing Cost Metrics

In global shortest path computations, delay and hop count are two commonly minimized metrics. In the satellite mesh, a minimization of hop count, while potentially simpler, is suboptimal because the links have different propagation delays (shorter near the poles). To study the degradation incurred by using hop counts instead of delay as the cost metric, we ran simulations for two identical sets of source-destination pairs (again, 10,000 pings with the endpoints selected at random), with the simulator configured to compute global shortest paths based on link propagation delays (min-delay) on one hand, and hop counts (min-hop) on the other. We then calculated the difference in delay experienced for each ping. Figure 6.11 plots, as a function of the number of satellite hops, the average and maximum delay degradation from using min-hop instead of min-delay routing for a Teledesic-like constellation, while Figure 6.12 plots these values for an Iridium-like constellation, again assuming the presence of cross-seam ISLs. The error bars around the average values represent one standard deviation.

Although on average the penalty for using hop count as the routing metric is generally below 10 ms, the maximum difference can be quite high. These outliers were due to particular configurations in the constellation where there were a multiplicity of minimum hop paths through the mesh, some of which used more (short delay) links in the low latitudes, and some of which used more (longer delay) links in the high latitudes. In these outlier cases, the minimum hop path that was found first was one that included a lot of low latitude satellites. Figure 6.13 illustrates an example (using the Teledesic constellation) of how two routes with the same number of hops can have very different end-to-end delays. Another interesting feature of the data is that the maximum and average delay difference decreases for the very largest distances. Furthermore, the difference between min-hop and min-delay paths in the Iridium system are not as large, because there are fewer

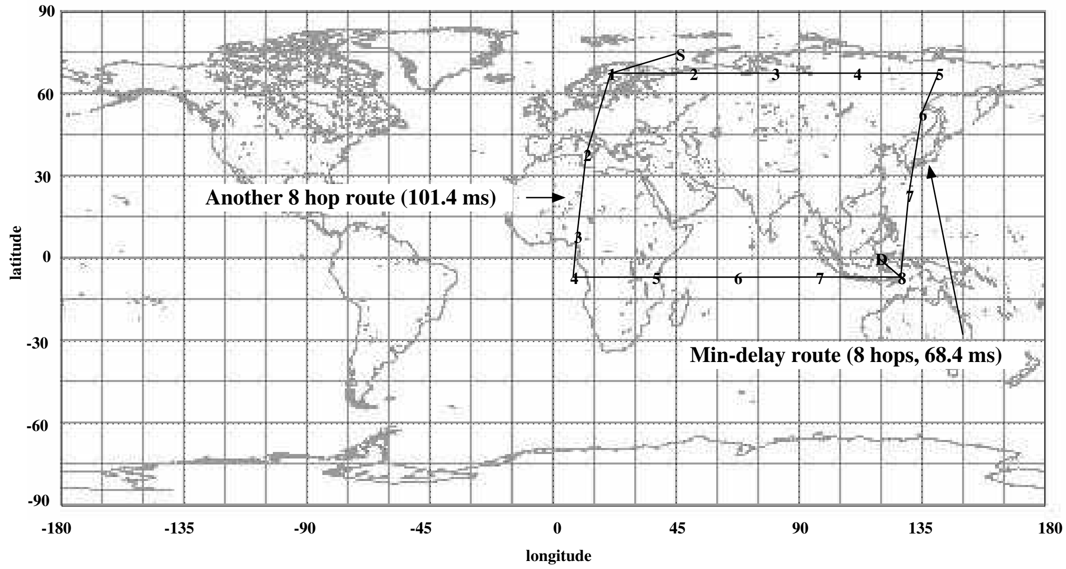


Figure 6.13: An example of how min-hop routing can occasionally select a route with a much larger delay than that of min-delay routing. Both routes contain eight satellite hops, but one route uses hops in the higher latitudes which incur much less propagation delay.

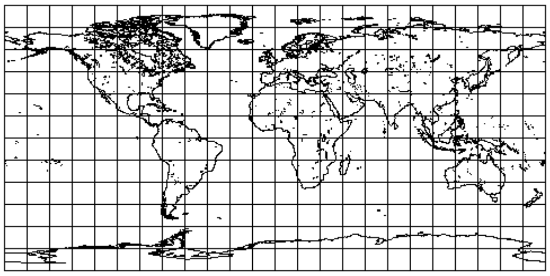
satellites and hence, fewer candidate paths to choose from.

6.4 Geographic Addressing and Cellular Geometry

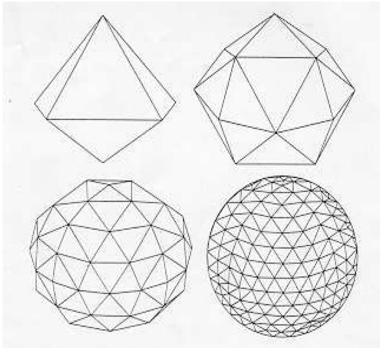
The results presented in the previous section illustrate the delay performance of the LEO constellations using an omniscient, centralized routing agent, running shortest-path algorithms, that immediately updates each node's forwarding table upon a topology change. As such, these results essentially bound the achievable delay performance in these constellations. However, such a centralized routing system carries with it a cost; namely, a high amount of traffic for routing updates on the ground to satellite links of the system. In the remainder of this chapter, we explore what kinds of routing algorithms are possible if we try to capitalize on the predictable and (nearly) regular topology of LEO networks. In this section, we first introduce *geographic addressing* as a potential technique to achieve better routing scalability. We conclude by evaluating alternatives for cellular geometries on the Earth's surface. In the remaining sections we will then explore two particular routing designs based on geographic addressing and the cellular geometry that we select.

6.4.1 Geographic Addressing and Mobility

Geographic-based addressing (i.e., including some representation of a terminal's geographic location as part of its address) is a natural addressing hierarchy for a LEO satellite system because terminals that are located close to one another are likely to have their packets routed in a



(a) Rectangular map projection
(from the Xerox PARC Map Viewer)



(b) Regular tessalation of octahedron
(from [35])

Figure 6.14: Alternatives for dividing the Earth's surface into cells.

similar way. We consider the possibility that each terminal is assigned a unique address that consists of a portion (for conceptual purposes, a "prefix") that represents the current geographic location of the terminal, and a portion that is globally unique to the terminal. The geographic portion of the address can be dynamically changed. The unique portion of the address is static and can be, for example, an IP address.

If a terminal changes its geographic location, the prefix can be dynamically updated. Determination of a change in geographic prefix can be relatively straightforward– if the change is semi-permanent, a postal code or GPS coordinate may be used to determine the new prefix. Alternatively, satellites could be configured to broadcast on a beacon a list of legal prefixes for a given spot beam, and a terminal could pick from among the set if its prefix was no longer valid. However, simply updating the address is not sufficient; some type of mobility mechanism must be implemented at the network layer for other network nodes to communicate with the mobile terminal, This could be embedded in the satellite nodes themselves similar to the mobile IP solution [95], or could be implemented in a mobility database updated by the mobile terminals themselves. The exact handling of terminal mobility is not central to our research; we merely point out that potential solutions exist and could be the subject of future research.

6.4.2 Cellular Geometry

If we choose to represent a terminal's geographic location by a finite set of address bits, we are implicitly requiring some kind of cellular structure on the Earth's surface. There are no strict requirements on the cell size; i.e., there does not have to be a precise match between the cell size used for addressing and the radiation footprints of the satellites. Large cells have the benefit of requiring fewer bits to represent the address and, to the extent that aggregation is successful, require fewer routing table entries. However, larger cells are less flexible in composing footprint-sized regions in an Earth-fixed cell system, because the granularity of where the footprint boundaries can occur becomes too coarse. Furthermore, cells at the perimeter of a footprint area may have some of the terminals served by neighboring satellites. This means that terminals in such cells cannot be aggregated and must be individually represented in multiple satellites' routing tables. We do not

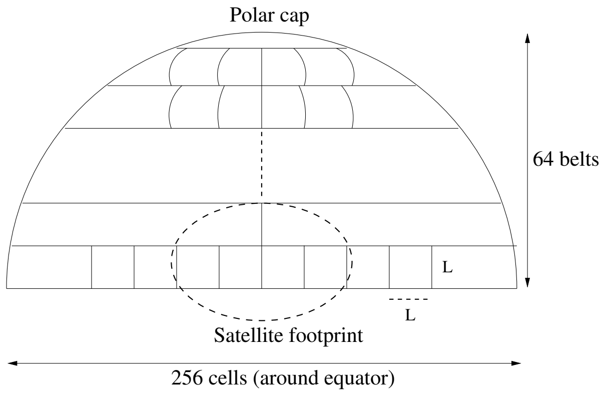


Figure 6.15: A cellular geometry consisting of roughly equal-sized trapezoidal cells [118].

explore these tradeoffs in detail because they are largely system-dependent. Instead, we focused on cell sizes that correspond roughly to the "supercell" sizes in the original Teledesic proposal (roughly 160 by 160 km).

We considered three alternatives for a cellular geometry. The first, a regular square grid superimposed on a rectangular map projection (Figure 6.14a), lends itself to an easy convention for addressing cells; in particular, cells can be numbered in such a manner that simple binary arithmetic operations can be used to compute rough distance estimates between the two cells. It has the drawback, however, of non-uniform cell sizes and severe distance distortion at high latitudes. The second technique, a tesselation of a regular polyhedron (Figure 6.14b) such as has been developed by GIS researchers [35], exhibits much less distortion because the mapping is largely invariant to the position on the globe. However, it is difficult to number the resulting cells in such a manner that distance computations between cells are straightforward and geographically contiguous cells can have their addresses aggregated (in particular, such a tesselation does not lend itself easily to mappping onto a lattice). A third technique is to use a cellular geometry as described by Restrepo and Maral [118]. This approach divides the Earth's surface into a number of roughly equal-sized trapezoidal regions, as shown in Figure 6.15, and has the benefit of being easily numbered in two dimensions without suffering from the distortions of a rectangular map projection. First, the surface is divided into a number of latitudinal bands of uniform height. Second, each latitudinal band is subdivided into a number of trapezoidal (almost square) cells. Fewer cells can be fit into latitudinal bands at higher latitudes, but as long as the constraint of an integer number of cells is satisfied, the cells at different latitude bands are roughly the same size. The cell structure is terminated at each pole with a polar cap. If we define 128 latitude bands (including two polar caps) and a maximum of 256 cells in each band, we obtain cells that are roughly the same size as the "supercells" in the original Teledesic proposal [18]. We selected the third technique because it is well suited to a cell numbering scheme that we consider below in Section 6.6. In the remaining sections, our descriptions of an underlying cellular geometry will refer to this third technique.

6.5 Design and Evaluation of a Distributed Routing Protocol

In the simulation results presented in Section 6.3, all satellite nodes had access to complete topology information so that they could generate explicit shortest path routes on demand. In practice, either this topology information must be present at all nodes, or it must be present at some centralized control station that periodically uploads complete forwarding tables to each satellite, or approximations can be made. In this section, we explore one such approximation that has been previously proposed in the literature: whether a geographic-based addressing scheme may be used in a distributed routing system by allowing local packet forwarding decisions to be based on reducing some distance measure to the destination.

6.5.1 Overview

Performing packet routing by using geographic information embedded in the addresses is based on the hypothesis that, in a LEO system with a regular mesh topology, a series of locally optimal forwarding decisions (namely, routing to the neighboring satellite that most reduces the distance to the destination) will yield a route that is close to optimal when compared with the globally optimal route. Each forwarding decision is based on reducing some measure of the distance to the destination: a satellite with a packet to route first determines its distance to the destination, and then determines the distance from each of its immediate neighboring satellites to the destination. It is assumed that location information for a satellite and its immediate neighbors is readily available, and that distances can either be computed on-demand via binary arithmetic or looked up in a table. A satellite then routes a packet to the neighboring satellite that most reduces the distance to the destination. Although this routing strategy has been previously proposed in the literature [124, 55], it has not been worked out fully.[2] In this section, we describe our efforts to base a distributed routing protocol on this strategy, and the challenges that we encountered in doing so.

One concept that has appeared in the literature is that of defining satellite "virtual nodes" to simplify routing [84]. The key idea is to add a level of indirection to the system by assigning fixed portions of the Earth's surface a logical address. Then, by using the Earth-fixed cell technique described above in Section 2.2.1, a satellite embodies the virtual node above this fixed Earth footprint for the duration of time that it is serving that footprint. Carried to the extreme, a static logical network can be defined as exemplified by Figure 6.16, and no dynamic routing need be performed. However, this extreme case implies a one-to-one mapping between terminals in a given cell and the current satellite serving the cell, which will lead to a decrease in system availability for the following reasons. First, terminals at the very edges of these fixed footprints may often find that they could receive better coverage from the satellite serving the neighboring footprint than from the satellite to which they are forced to connect. Second, there will be occasions when the satellite serving a fixed footprint will be in the same line of sight as the sun and communications are impossible (this is known as a "sun outage"). Unless neighboring satellites can train a spot beam on this location for this period of time, the system will become unavailable. Third, since user density is highly non-uniform around the globe, it will be advantageous for neighboring satellites to train additional spot beams on regions of high density (although we do not consider such system

[2] We know of one current, commercial, packet radio network (Ricochet) that uses geographic information to route packets from poletop radios to a gateway station; however, the network topology over which this is used is static, and routing around congested nodes is not performed.

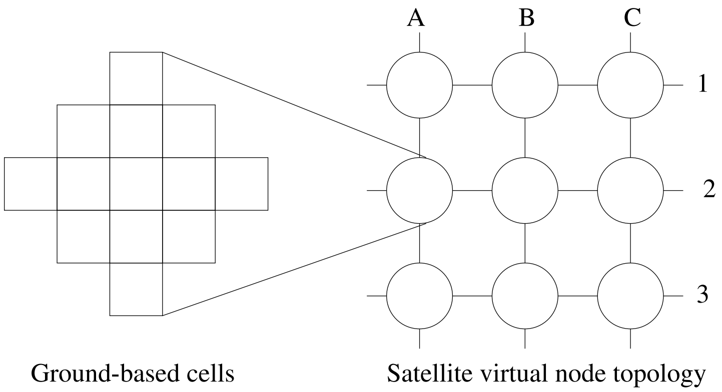


Figure 6.16: A logical network topology: fixed zones on the Earth's surface are assigned a logical address, and a satellite serving a particular zone embodies the logical node serving that region (from [84]).

optimizations herein). Once the one-to-one correspondence between terminal and satellite virtual node is broken, some type of dynamic routing becomes necessary. Nevertheless, the satellite virtual node concept is useful if one relaxes the constraint that the footprints be fixed on the Earth's surface. If a satellite footprint can be decomposed into multiple smaller cells, then "semi-fixed" footprints (fixed for some finite amount of time before a handoff is needed) can be composed of these smaller cells such that system availability is maximized (i.e., the boundaries of the Earth-fixed footprints can dynamically change as needed).

As we discuss in the next section (Section 6.6), a routing architecture based on centralized routing may be preferred to one based on distributed routing for several reasons. If, however, geographic-based routing were to be simple and robust enough to be easily deployed in a LEO system, then it would have certain advantages over centralized routing; namely, a reduction in the amount of message overhead between the ground and satellites, and smaller routing tables. We therefore were seeking to explore this routing concept further by designing a robust, distributed routing protocol simple enough to be an attractive option when compared with centralized routing.

6.5.2 Construction of a Distributed Routing Protocol

In this section, we describe our construction of a distributed routing protocol based on the above hypothesis and evaluate its performance. We first describe the basic technique and our performance metrics. Conceptually, the distributed geographic routing protocol is straightforward, but in applying the concept to real constellations we required certain enhancements for correct performance. We describe these enhancements, which include supplementing the protocol with locally-scoped shortest path information around destinations and special handling of packets in the high latitudes. Finally, we discuss the performance of this overall routing strategy.

We implemented the basic protocol in the *ns* simulator. Specifically, we assumed that each satellite knew the cell that contained its nadir point, and the corresponding nadir-pointing cells of all of its neighboring satellites to which it had active ISLs. When a satellite received a packet for a destination terminal that it did not serve, it computed the great-circle distance from the center of its cell to the center of the destination cell, and likewise computed the distance from all of its neighboring satellites to the destination. If one or more neighboring satellites had a smaller distance to the destination, the satellite forwarded the packet to the satellite that most reduced the distance to the destination; otherwise, the packet was dropped.

We evaluated the routing protocol performance using the following approach: we repeatedly picked two points on the globe at random, and tried to route two packets between them. The first packet was routed using a global shortest-path algorithm based on minimization of the propagation delay of the route. The second packet was routed via the distributed protocol based on geographic-based packet forwarding. We were interested in two performance metrics: the *robustness*, as measured by the ability to avoid routing "dead-ends" (and hence packet drops), and the *delay degradation* of the geographically-based route as compared with the optimal route. We therefore calculated the delay experienced by both packets if the routing was successful for both packets, and noted any routing failures for packets using the distributed routing failure (the packets routed by using globally-optimal shortest paths were never dropped). We chose to simulate a large set of random points rather than use an exhaustive combinatorial search because the latter would have require checking for successful routing from each cell to every other cell (an $O(n^2)$ operation, where n is on the order of 20,000) at each point in time (or a set of discrete points in time for which the topology is assumed static for a certain time interval). Unfortunately, this discretized state space is very large for commercially proposed topologies, and the exhaustive search is computationally infeasible. Nevertheless, as we show below, using a large number of random trials was sufficient for evaluation purposes because it exposed a number of weaknesses in the approach.

Regardless of the delay performance, a fundamental requirement of our protocol was robustness, or the avoidance of dropped packets due to routing dead-ends. As we describe in the following three subsections, we encountered a number of difficulties in achieving this robustness. First, in a polar-orbiting constellation, geographic routing frequently breaks down very near a destination. Second, in the polar regions, the regular mesh topology is disrupted, again leading to dead-ends. Finally, at the counter-rotating planes, the geometry of the orbits causes a large tear in the mesh topology. The next three subsections describe our efforts to engineer around these problems.

Locally Scoped Shortest Path

In a perfectly regular mesh topology in which destination terminals were always connected to the closest satellite, geographic-based packet forwarding would never result in a dead-end. However, since LEO satellites typically have overlapping footprints (since coverage redundancy is inherent in polar-orbiting constellations), the geographic forwarding may break down, as can be seen by the example shown in Figure 6.17. In the figure, a packet routed from S (connected to satellite 1) to D (served by satellite 6) proceeds via geographic routing to satellite 4. At this point, however, satellite 4 cannot route the packet to any of its neighboring satellites without increasing the distance to the destination. By forwarding to a satellite that increases the distance to the destination, we open the possibility for a routing loop to be formed, and although techniques can be used to prevent packets from being forwarded back to a previously visited node (such as encoding the history of the traversed route in the packet header), we still cannot guarantee that a packet so forwarded will eventually find the right egress node.

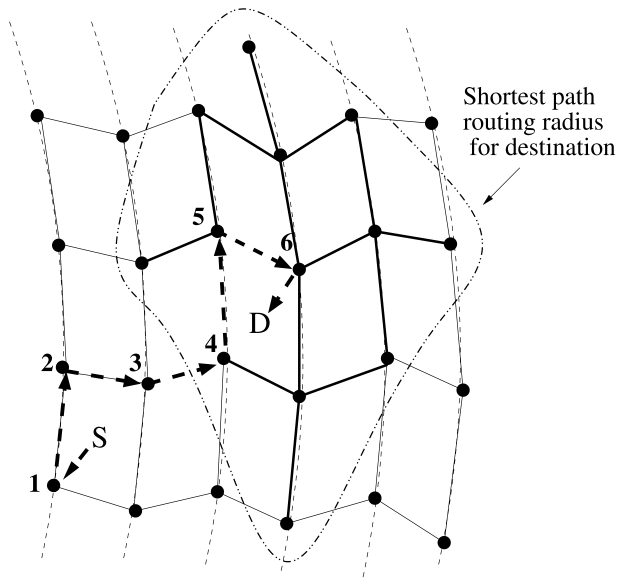


Figure 6.17: Hybrid routing strategy based on geographic packet forwarding for distant destinations and locally-scoped shortest path routing for local destinations. The figure denotes a subgraph of the satellite mesh and a hypothetical packet trace. A packet sourced at S is forwarded based on geographic information to the satellite numbered 4. Satellites use shortest-path routing information to complete the routing to destination D, which is served by satellite 6.

Our solution was to use a locally-scoped shortest path algorithm to complete the packet forwarding process close to the destination. We implemented a basic link-state routing algorithm such as is described in [109]. Instead of flooding each link state packet (LSP) to every node, however, we flooded an LSP only as far as the routing radius for a given satellite. The routing radius was determined such that it covered every possible satellite that could potentially serve the destination– typically two hops was sufficient for the Iridium constellation, and two or three for Teledesic. The flooding protocol makes use of packet numbers to suppress transmission of duplicates. Each satellite therefore had a map of a subgraph centered on itself. When computing routes, the satellite used only those LSPs for which it had records, and computed routes only as far as its own routing radius. In other words, even if a satellite had the LSPs available to compute routes to a destination further away than its routing radius, it did not do so (because each satellite is only able to guarantee having current LSPs from a number of hops away equal to the routing radius). The routing radius can be controlled by a TTL field in the routing protocol header. As an example, Figure 6.17 illustrates the case for which the routing radius is two hops, and the dashed boundary around satellite 6 denotes those links and nodes that are used in satellite 6's routing computations. The protocol therefore requires a hybrid approach that uses geographic-based packet forwarding to get a packet in the vicinity of a destination, and shortest path routing to finish the final few hops to the destination. Such a solution is also recognized by Mauger and Rosenberg [84], in that the authors propose to resolve the inherent last-hop ambiguity around a destination by flooding this connectivity information with neighboring satellites. However, they do not discuss how to make use of this information in a routing algorithm or how far to propagate this information around the destination. We would prefer to avoid a pure flooding approach because of the bandwidth that it would require.

Let us discuss the robustness and complexity of this approach. In general, routing loops can form whenever nodes make routing decisions based on inconsistent information. Transient loops are possible in any dynamic topology, but we can still strive for a protocol that converges to correct routes in finite time after any topology change. In our case, since all routing information is locally-scoped, each node has a slightly different view of the network topology, which can lead to the following problems. First, if different nodes have different routing radii, it may be possible for stale routing information to persist. For example, consider satellite A with a routing radius of two hops and satellite B with a radius of three hops, and assume that satellite A is initially within two hops of satellite B. If the topology changes and satellite A moves to three hops away from satellite B, satellite A's LSPs will no longer reach satellite B. However, satellite B can still route to (and through) satellite A based on satellite A's LSP because satellite A is within satellite B's routing radius. Second, we must prevent the occurrence of routing loops that could form if a packet enters a locally-scoped routing radius of a destination and is somehow subsequently forwarded to a satellite outside the routing radius. Third, it is well known that if different nodes use different routing metrics (such as dynamically adapting to congestion based on local information), loops are possible. This last problem is a general dynamic routing problem and can be avoided by making sure that all nodes use the same routing metric and have up-to-date link costs.

The key to avoiding such routing loops is for each node, when constructing a path, to consider the routing radii of all of the nodes along the path, and to ensure that stale routing information is successfully purged from each node. The first goal can be realized by requiring satellites to advertise their own routing radius in their LSPs. Furthermore, we modified the shortest path algorithm to construct complete paths to the destination and to check whether the satellite constructing such a path is within the routing radius of all nodes in the path. For example, consider satellite A using its gathered routing information to construct a (shortest) path to D through satellites B and C ($A \Rightarrow B \Rightarrow C \Rightarrow D$). Three constraints must be satisfied for satellite A to consider this a legal route:

1. Satellite D must have a routing radius of at least three hops.
2. Satellite C must have a routing radius of at least two hops.
3. Satellite B must have a routing radius of at least one hop (trivial).

If these constraints are satisfied, then satellite A can be sure (aside from the possibility of transient loops due to topology changes) that if A forwards a packet for D through B, that it will not receive the packet again. This is because, for a downstream node to forward the packet back to A, that node must have made a calculation that A lies on its shortest path to D, which is a contradiction because if so, A would have originally picked the remainder of this path to D to begin with. Note that this approach also precludes the troublesome possibility identified above that a packet may leave the shortest-path routing radius once it enters. Furthermore, we do not guarantee that the actual path followed will match exactly the path predicted by an upstream node, but if the actual path does in fact change downstream, it will do so only in a manner that does not increase the total path cost.

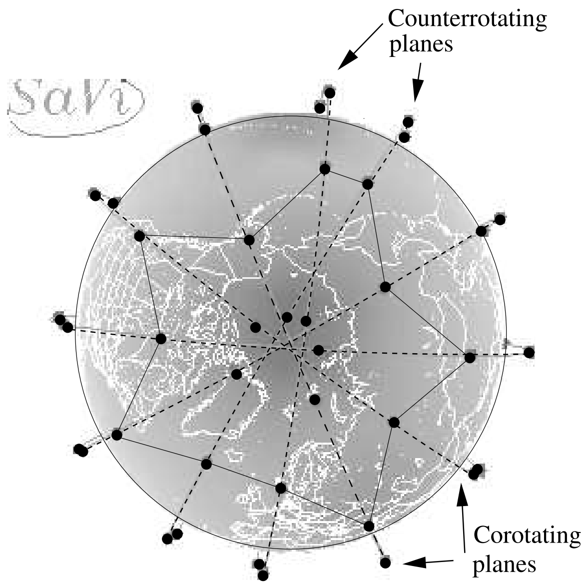


Figure 6.18: View of the Iridium topology above the North pole. Satellites closest to the pole have interplane ISLs turned off. The "polar region" is bounded by the set of satellites closest to the pole that have all of their interplane ISLs active.

With this approach, we still must make sure stale information is purged from the system. LSP updates will naturally purge stale information, except if a node dynamically decreases its routing radius. In this case, the node needs to make sure that its old LSPs are expunged from all nodes at the periphery of its routing radius.

As for complexity, although this approach requires implementation of a shortest-path protocol, the processing and memory overhead is significantly reduced by scoping the LSP propagation (and hence, the state information) to a small region around each satellite. The modifications to the shortest path algorithm discussed above do not significantly increase its complexity.

Geographic forwarding bears some resemblance to the Landmark routing hierarchy [133] in that packets at locations far away from a destination are routed in the general direction of the destination, but unlike the Landmark hierarchy, there are no nodes for which every node keeps precise routing information. In fact, this geographic-based routing strategy is not hierarchical in the traditional sense but is instead a hybrid approach between shortest path routing and geographic forwarding. Another hybrid routing protocol, the Zone Routing Protocol for ad-hoc networks [53], also makes use of routing zones around each node for local traffic, but routes for distant destinations are queried on demand, rather than obtained by using geographic information.

Routing in Polar Regions

As stated above, the routing radius is defined as including all those satellites that can be observed above the elevation mask of a terminal. In addition, the radius must be extended whenever there are breaks in the topology. In the high latitudes, the interplane ISLs must be deactivated,

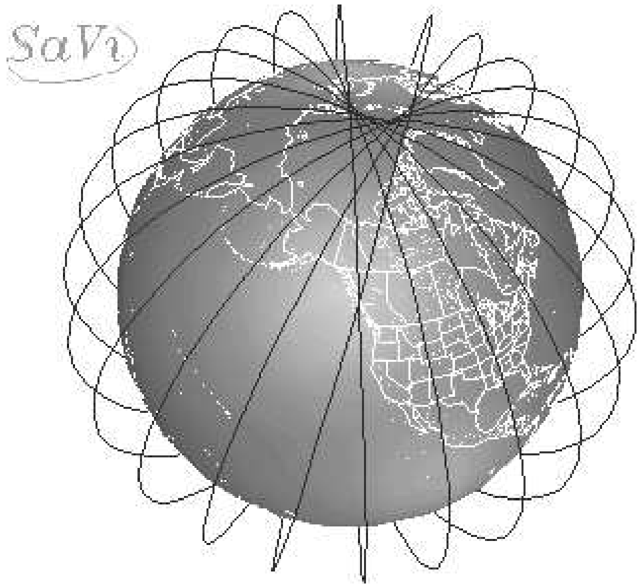


Figure 6.19: Illustration of the intersection of counter-rotating planes.

and for a packet to reach a satellite that has its interplane ISLs deactivated, the packet must first be routed to a satellite in the same plane but at a lower latitude. As a result, geographic-based packet forwarding can break down several hops away from the eventual destination. This implies that we should increase the routing radius such that all satellites in the polar region can obtain LSPs for all other satellites in the polar region. However, such a radius is sufficiently large (five or six hops in our simulations) that it would spill over significantly into the lower latitudes, increasing the amount of routing state required on each satellite (the amount of routing state required grows roughly quadratically with each hop). To compensate for this, we developed a special routing zone for the polar regions that specifically limited the scope of polar-area routing information to the polar region.

The key is to properly define and dynamically identify the polar region. Figure 6.18 illustrates a view of the polar region from directly above the rotation axis of the Earth, in which satellites near the poles do not have their interplane ISLs turned on, while satellites at lower latitudes do have interplane ISLs. The Iridium topology, with an orbital inclination of 86.4 degrees, is plotted. We define the polar region as including all satellites that have one or more interplane ISLs turned off (the POLAR satellites), as well as all satellites that border the POLAR satellites (the POLAR_BORDER satellites). If we define a third state (LOW_LATITUDE) that includes all other satellites, it is easy for each satellite to determine which state it is in by simply examining the state information of its neighboring intraplane satellites. Satellites can propagate state information to their neighbors using the same protocol as for propagating LSPs (since state changes are generally coincident with link state changes anyways). The key, then, is to extend the scope of LSP propagation of a satellite to the entire polar region in addition to the normal routing radius. Any packets directed toward a destination in the polar region will eventually find a satellite in this polar region, and then shortest

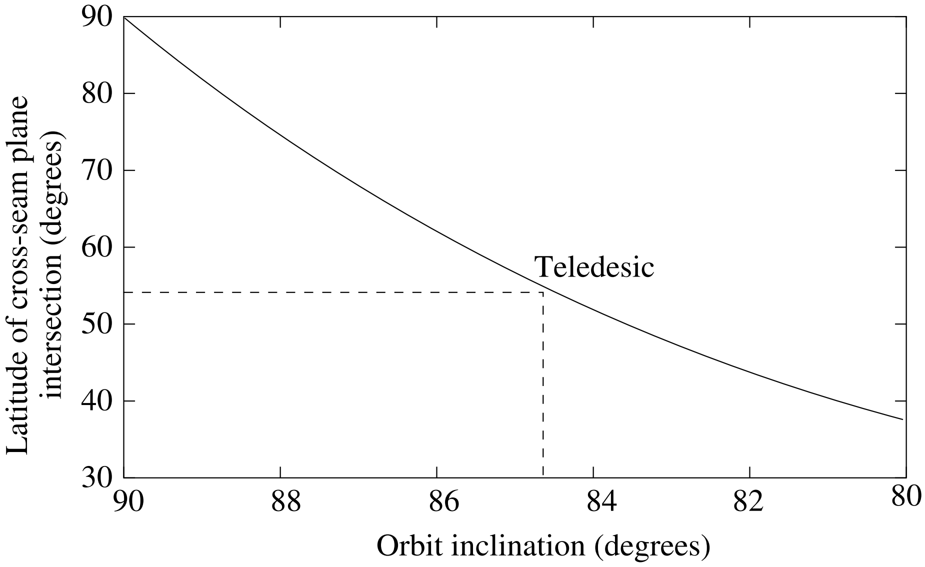


Figure 6.20: An illustration of how deviations from pure polar orbits cause the latitude at which the counter-rotating planes intersect to degrade. This plot assumes a 15^o degree plane separation such as used in the Teledesic design.

path routing can take over. Basically, LSPs that must be flooded to the entire polar region can be indicated by a bit in the header. Satellites expunge this extra state information when they leave the polar region, and announce their departure to the remainder of the polar region so their LSPs can be expunged from the rest of the polar satellites.

We also used this state information to "tunnel" packets to outside of the routing radius. If a packet is sourced by a terminal connected to a POLAR satellite, and the packet destination is outside of the polar region, then the packet will ultimately be routed to one of the two POLAR_BORDER satellites in the same orbital plane. Therefore, the satellite should use the location information of the two POLAR_BORDER satellites in computing the forwarding direction, instead of the location of the immediately neighboring satellites. This location information can be easily provided to the POLAR satellites for such computations.

Although constructing a special polar routing radius increases the amount of state kept by satellites at higher latitudes, and accounts for a increased message overhead in that region, this increase is offset by the fact that the normal traffic density in the polar region is likely to be extremely light. In the Teledesic constellation, the polar regions contained approximately 100 satellites (50 in each region), while the Iridium polar regions contained roughly 36 of the 66 satellites.

Problems at the Seams

Although handling the polar regions and the regions around the destinations required additional protocol, we were able to eliminate routing dead-ends in our experiments. However, a third problem presented more of a challenge. As mentioned above, the counter-rotating planes in

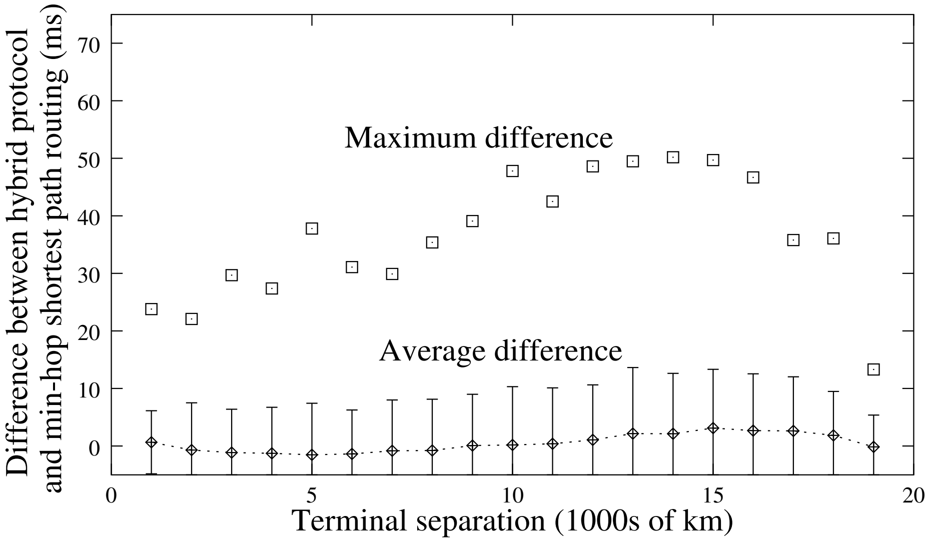


Figure 6.21: Average and maximum delay difference between using geographic forwarding and minimum-hop shortest path routing as a function of terminal separation (Teledesic constellation). Error bars denote one sample standard deviation from the sample mean.

a polar constellation form a "seam." It is possible to establish ISLs across this seam, although the link acquisition and synchronization associated with these ISLs are much more difficult than with interplane ISLs. However, the mesh is distorted in this region. First, as discussed above, there is only one ISL per satellite across the seam, since the second ISL will be used to acquire the next satellite before handover occurs. Therefore there is a paucity of links available in this region. A more significant problem, however, is that the (non-polar) inclination angle of the orbital planes causes the two counter-rotating planes to intersect at a much lower latitude than the other planes. This effect is clearly visible in Figure 6.19 for Teledesic (which plans an inclination angle of 84.7 degrees), where the two planes intersect at a latitude of approximately 54 degrees. If we let i denote the inclination angle of the orbital planes, and s denote the spacing between planes, then the latitude at which the cross-seam planes intersect is given by $\arctan(\sin(s/2) * \tan(i))$. This relationship is plotted in Figure 6.20 for an interplane separation of 15 degrees, as is planned for Teledesic. As a result, the cross-seam ISLs must be switched off at a relatively low latitude (actually, probably no higher than 45 degrees), which causes a tear in the ISL connection mesh. Regardless of whether geographic forwarding is used or not, this appears to be a drawback to using an orbital inclination angle that deviates significantly from 90 degrees. However, launching satellites into a purely polar orbital plane is considered to be prohibitively expensive, and these inclination angles may be the best that are economically feasible.

Although we tried various techniques (all based on distributed protocols) to tunnel around this tear in the topology, we were not successful in finding one that was reasonably simple to implement. Even when we constructed tunnels around these tears in the topology, we could always find cases for which the hybrid routing protocol faced a dead-end. These dead-ends are likely to persist,

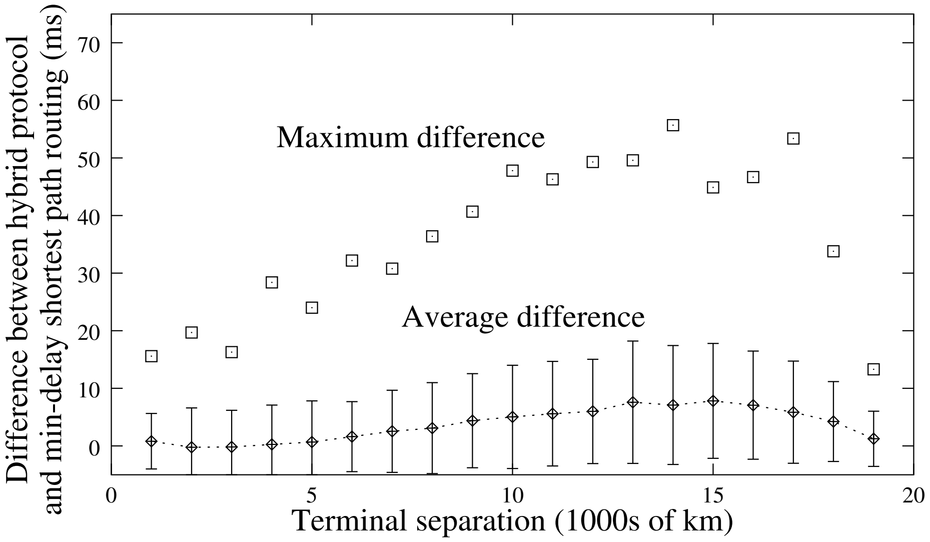


Figure 6.22: Average and maximum delay difference between using geographic forwarding and minimum-delay shortest path routing as a function of terminal separation (Teledesic constellation). Error bars denote one sample standard deviation from the sample mean.

at least intermittently, for as long as the seam separates the two endpoints (which could be hours). We note also that similar dead-ends are likely to occur when there are other tears in the topology due to satellite failures, which we did not investigate. In summary, we were not successful in guaranteeing the robustness of a geographic-based routing in the presence of a counter-rotating seam for the Teledesic and Iridium constellation topologies. The solution to this routing problem seems to require assistance from a centralized routing system, perhaps in the form of judicious installation of (several hop) packet tunnels across the seam.

6.5.3 Performance

Despite the routing breakdowns due to the counter-rotating planes, we did find that, on average, the delay performance of our hybrid protocol was comparable to that of min-hop shortest path. Figure 6.21 plots the average and maximum delay differences between geographic-based forwarding and min-hop shortest path routing for the Teledesic constellation. Figure 6.22 plots the average and maximum delay differences between geographic-based forwarding and min-delay shortest path routing for the Teledesic constellation. The data is drawn from an experiment of 10,000 random terminal locations. Three cases were run with the same set of terminals: the hybrid routing protocol described above (which used locally-scoped min-hop routing) global min-hop shortest path routing, and global min-delay shortest path routing. We then took the results from the hybrid protocol and computed the delay difference, point-by-point, between that protocol and each of the two shortest path protocols. We have collated the data points into 1000 km bins before performing the averages (e.g., point number 1 on the x axis lists the results for distances between 1000 and 2000 km). Each satellite used a routing radius of 2 hops while below 45 degrees latitude, and 3 hops while above (to reduce the occurrence of routing dead-ends). The main points to consider are those above 5000 km, for those are the ones for which a packet must traverse one or more geographic forwarding hops before hitting the shortest path routing radius. In addition to the averages, we tracked the maximum delay difference (penalty) from using the geographic-based protocol, as compared to the delays observed by min-delay routing.

We note from the figures that, on average, the geographic routing is comparable (no more than about 3 ms worse) to min-hop shortest path, but is roughly 5-10 ms worse than min-delay shortest path routing. Such an increase in average delay would probably not be considered significant to LEO network users. However, the maximum delay differences can be very large (up to 55 ms), and are from a small set of outliers. These points occur near the poles when the geographic routing initially brings the packet close to the destination in terms of distance, but far away from it in terms of topology, and it consequently must be routed back towards the particular orbital plane containing the satellite serving the destination.

6.5.4 Summary

In this section, we have studied whether using geographic-based addresses can enable a simple distributed routing protocol based on reducing the geographic distance to a packet's destination. Although the delay performance of the hybrid routing protocol that we designed was adequate, the robustness in terms of avoidance of routing failures was not. We encountered a number of difficulties in making the this routing approach robust: i) the redundancy in coverage around a terminal's destination requires some form of locally-scoped routing information, ii) the regular mesh structure is disrupted in the polar regions, requiring a special protocol to efficiently handle the routing in that area, and iii) the counter-rotating planes in polar-orbiting constellations intersect at a low latitude, preventing the establishment of cross-seam ISLs in a large region and thereby causing a tear in the topology. Because we were not successful in establishing robust routing when there were no node or link failures, we did not investigate the effects of such failures; however, we note that such a distributed routing protocol would also need to be robust in the face of such equipment failures. We have concluded that, for polar-orbiting constellations, basing a distributed routing protocol on geographic forwarding is prone to either failure modes or high complexity.

6.6 Centralized Routing Performance

Recall that our main design goals for a LEO packet routing architecture, aside from the basic goals of correctness and route completion, are a minimization of spacecraft hardware requirements (memory and processing), a minimization of routing traffic, and robustness in the routing algorithms. A distributed routing protocol offers the opportunity to minimize message exchange between the ground-based network operations center (NOC) and the satellites, but this minimization typically comes at a cost of increasing both the amount of message traffic that must be exchanged between satellite nodes and the processing required to consume this routing information. In the previous section, we showed that one such distributed protocol, based on geographic packet forwarding, presented some subtle difficulties when applied to commercially-proposed polar-orbiting constellations. In this subsection, we consider the alternative of a centralized routing architecture.

A centralized routing system would consist of a ground-based route computation center that frequently uploads forwarding tables to satellites. This approach may be preferable to distributed routing for three main reasons. First, all of the topology information will already be located at a centralized location that performs additional functions such as medium access admission control and network management. Moreover, this information will generally be available in advance of the time needed because many topology changes are predictable. Second, there will be a need to communicate with each satellite on a regular basis to perform other control functions, such as dynamically adjusting the scanning beam patterns of each satellite's antennas. Third, centralized routing more readily permits sophisticated routing algorithms in the network. For instance, the traffic load may evolve in such a manner that load balancing within the satellite mesh becomes necessary; a centralized routing system would be more easily upgradable.

Although centralized routing reduces the spacecraft processing requirements by requiring that it only lookup next-hop interfaces and not compute and distribute routing information, it is still important to reduce i) the amount of routing information that must be sent to the satellites, and ii) the size of satellite routing tables (which, in the worst case, could require on the order of a million entries if no hierarchy or table aggregation is used). In this section, we describe techniques, again centered on the concept of geographic-based addresses, that may be useful in meeting both goals. First, we describe in more detail the cellular structure that we use and present a numbering scheme that is optimal from the standpoint of aggregating contiguous cells. Next, given such a cellular structure, we focus on whether we can take advantage of temporal and geographic consistencies in the routing table to reduce the amount of routing information that must be dynamically uploaded. Finally, we explore the problem of actually performing the aggregation of geographically contiguous cells into a small number of routing table entries.

6.6.1 Cellular Structure and Addressing

We described above in Section 6.4 a cellular geometry introduced by Restrepo and Maral based on roughly equal-sized trapezoidal cells (Figure 6.15), and we have patterned our cellular geometry after theirs. One difference in our geometry is that we require that the number of cells in each latitudinal band (aside from the polar cap) be an integer multiple of four (a convenience we take advantage of as described in the next paragraph). Further, we require that cells be no larger than those in the band abutting the equator. If there are n cells in this first latitudinal band, then the base of each cell in this band is $B = 40,074/n$ km, where $40,074$ km is the circumference of the Earth at the equator. To first order (assuming a spherical Earth), the height of each cell in each band is also B. In each latitudinal band, then, let C denote the circumference of the base of the latitudinal band (e.g., for the first band, C is the circumference at the equator). There are then $4 * k$ cells in the latitudinal band, where k is the smallest integer satisfying: $C/(4 * k) <= B$. The last latitudinal band is a single cell ("polar cap"). By picking $n = 256$, we obtain cells roughly the size of the original Teledesic supercell design [18], and a total of 21,352 cells of approximately equal area (with bases ranging from 156.5 to 146.3 km at all latitudes except those very near the poles, and uniform height of 156.5 km). In general, n could be any integer, but there is a coding efficiency gained if n is a power of two.

Our next step is to map this (spherical) cellular structure to a rectilinear grid, to facilitate address aggregation. We will then use the grid for addressing in latitudinal and longitudinal directions. We can map a rectangular grid of size n by $n/2$ onto the cellular geometry that we

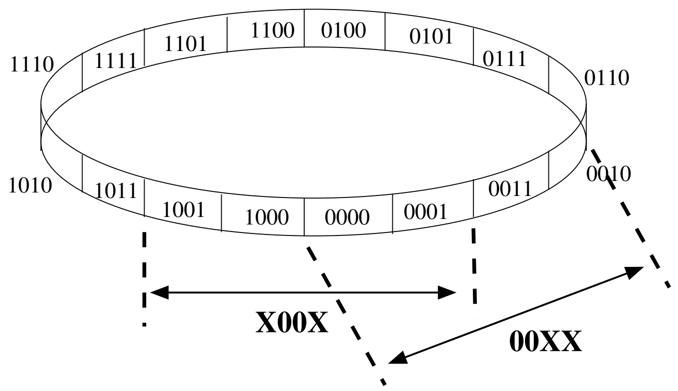


Figure 6.23: Example of the cellular numbering strategy in one dimension. Two potential size-4 aggregations are illustrated.

just described. For latitudinal bands near the equator, there is a one-to-one mapping between grid points and cells. However, if the number of cells in a latitudinal band is less than n, then some cells in that band will have more than one grid point mapped onto them. In this case, our mapping strives to distribute these redundant grid points uniformly around the latitudinal band. For example, if $(4 * k) = 252$, we have four redundant grid points. Every 63rd cell, then, would have two grid points mapped onto it instead of one. The polar cap would have 256 gridpoints mapped onto it. In total, 32,768 grid points would map to 21,352 cells. The result of this mapping is that, if we number the gridpoints in two dimensions (corresponding roughly to a latitude and longitude), cells along the globe that fall on the same longitudinal line will have roughly the same longitudinal component in their addresses.

We must next number these grid points. The grid point (and hence, cell) numbers will then form the geographic prefix portion of the terminal address. In the case of cells with more than one prefix, all terminals can be assigned to one of the prefixes in the set. Since our aim is to aggregate geographically contiguous cells, it will help if adjacent cells on the grid have similar addresses. With this concept in mind, we decided to use the principles of Gray encoding used in digital modulation [115]. A Gray code is a function $G(i)$ of integers i ranging from $0 \leq i \leq 2^N - 1$ that is one-to-one and for which the binary representation of $G(i)$ and $G(i+1)$ differ by exactly one bit [114]. The two dimensions can be numbered independently (8 bits per dimension).

Figure 6.23 is an example of this numbering scheme applied to 16 cells in one dimension. The figure illustrates two examples of blocks of four cells being aggregated into a 4 bit routing table entry and 4 bit (non-contiguous) bit mask. Such an aggregation would be useful if, for example, packets from a given satellite were forwarded over the same interface to each cell in the block; the routing table for that satellite would only require a single (address, mask) entry. In general, non-contiguous bit masks are deprecated in the assignment of IP subnet masks because they preclude the use of certain lookup algorithms [91]. However, we consider using them in our case if we can obtain a large reduction in the number of table entries.

We now prove that this method of numbering of cells is optimal, from the standpoint that it offers the most opportunities for address aggregations of various block sizes across various cell boundaries. We consider the numbering of cells in one dimension, as the two dimensions are orthogonal. Consider $2^k = n$ cells defined by k bits, where k is a natural number. Define a *mask level* l, where $l \leq k$ is the number of masked bits. With l bits masked, there are therefore 2^{k-l} unique address representations, each of which are of size 2^l cells. Note that there are also potentially 2^l possible ways to partition the space of n cells into 2^{k-l} contiguous blocks of size 2^l cells (i.e., there are 2^l locations to draw the block boundaries). Our goal is to maximize the number of possible partitions that consist of contiguous cells, so as to maximize the subnetting flexibility.

Lemma 6.1 *At mask level l, there are at most two ways to partition the cells into blocks of* 2^l *contiguous cells such that the blocks can be aggregated. In particular, at mask levels* 0 *and k, there is only one possible partition.*

Proof: For the sake of discussion, assume that adjacent cells are numbered sequentially from zero to $n-1$, starting from some arbitrary cell; this numbering is not necessarily related to the addressing bit assignments. For $l = 0$ there are no bits masked and therefore no aggregations are possible. For $l = k$, all bits are masked, and there is only one partition, which contains every cell. For $0 < l < k$, we proceed as follows.

Assume that at mask level l, cells are addressed in such a manner that there exists at least one partitioning of the n cells into contiguous blocks of size 2^l cells. Consider an arbitrary partition that starts, without loss of generality, at cell 0. The partition boundaries delimit sets of cells identified by the unique combination of $k - l$ non-mask bits. Note that among the masked bits, each bit must have an equal number of ones and zeros across each contiguous block, because the masked bits must represent 2^l cells uniquely. Also, note that among the unmasked bits, each bit must hold the same value across a contiguous block. Next, consider the same addressing, but with a second partitioning into contiguous blocks of size 2^l cells, across different cell boundaries. To obtain this partition, we must unmask one or more mask bits, and mask the same number of previously non-masked bits. Again, for this to be a valid partition, we require that for each masked bit, there must be an equal number of zeros and ones across the contiguous block. However, since at least one of these newly masked bits was previously unmasked, and hence had the same value in blocks from the old partition, the new partition boundaries must be offset by the original partition boundaries by exactly 2^{l-1} cells in order for the zeros and ones density to work out. Furthermore, by the same argument there can be no further partitions. ■

Theorem 6.1 *The method of Gray encoding of cells described above is optimal for aggregation of contiguous cells.*

Proof: At each mask level $0 < l < k$, there are exactly two ways to partition the blocks of 2^l cells: one on cell boundaries of $\{0, 2^l, 2*2^l, ..., (2^{k-l}-1)*2^l\}$, and the other on cell boundaries of $\{2^{l-1}, 2^l + 2^{l-1}, 2*2^l + 2^{l-1}, ..., (2^{k-l}-1)*2^l + 2^{l-1}\}$. By Lemma 1 above, this achieves the maximum possible partitioning. ■

6.6.2 Reducing Routing Table Size Updates

Given the above cell numbering scheme, we next seek to reduce the amount of bandwidth consumed by a centralized routing system that periodically uploads forwarding tables to satellites. A key assumption for this part of our work is that the satellite network topology is held static for

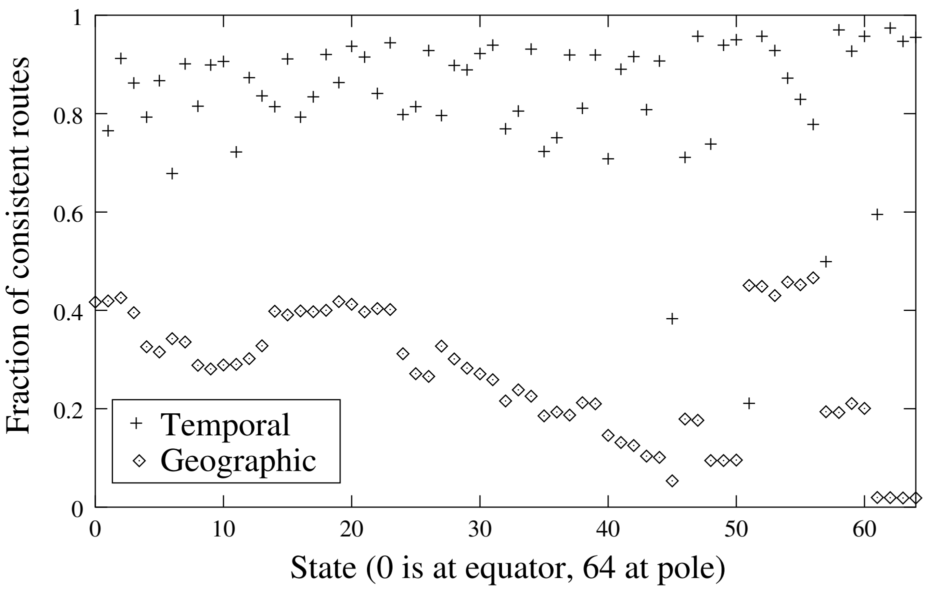


Figure 6.24: Comparison of temporal and geographic consistency of forwarding tables across topology states (Teledesic constellation).

a certain time interval, [3] and that routes can be precomputed on the basis of anticipated topology changes. An Earth-fixed cell system (described above in Section 2.2.1), in which ISL topology changes are also constrained to occur only at certain times, is one example of such a system. The system can then be thought of as moving through a (possibly very large) set of discrete states, each of which has a static topology (not considering unexpected topology changes due to link or equipment failures). If the system topology is not approximately static for a reasonable interval (e.g., tens of seconds), then there is little hope of constructing a low overhead, low latency centralized routing system.

A satellite forwarding table should contain enough information to forward packets to any terminal in the system, because approaches that require querying for routes on demand will be too slow for a broadband satellite system even at LEO altitudes. Terminals in cells distant from a given satellite can be aggregated into a single cell entry based on their prefix, while terminals in nearby cells may require individual listings in the forwarding tables if different satellites are serving terminals in a cell. In this subsection, we focus on techniques used to reduce the amount of message overhead required to populate correct forwarding tables on-board the satellites.

Rather than upload completely new forwarding tables each time the state changes, we have investigated two techniques aimed at minimizing the amount of information that must be uploaded. We seek to capitalize on the following two properties of the forwarding tables:

Temporal consistency: If only a few entries in the forwarding table change between states, then a centralized routing system can use delta encoding (sending only the changed entries).

[3] By static, we mean that the ISL topology is unchanged, and that intersatellite handoffs of terminals are minimized or avoided. Terminals may be connected or disconnected to the system at arbitrary times.

Geographic consisitency: We showed in the previous section that basing a distributed routing protocol completely on geographic-based packet forwarding decisions is fragile. However, if this core packet forwarding technique could be supplemented by additional forwarding instructions from a centralized routing system, then geographic forwarding could be used by default and only those forwarding entries needed to override the geographic forwarding (i.e., entries for which the central controller determines that the satellite would otherwise make a bad decision) need be uploaded.

We investigated the potential for both of these techniques by studying the forwarding tables created by min-delay shortest path routing for a representative Teledesic satellite as it moves through its orbit. Using the cellular structure described above, we assumed that the topology may be held static for the interval defined by the time required for the satellite's nadir point to traverse a cell (26.5 seconds). In a real system, the topology may be held static for longer than this interval (depending on the steering capabilities of the spacecraft's antenna), but there appears to be no advantage to making the interval shorter. We computed complete forwarding tables at every state (64 intervals from the equator to the north pole) by placing a terminal in each of the 21,352 cells in our cellular geometry. For each cell, we compared the forwarding table entry with the forwarding decision that geographic forwarding would have made (geographic consistency) and with the entry from the last state (temporal consistency). The results are shown in Figure 6.24, where the fraction of matches (among the 21,352 cells) are recorded for each state. The figure illustrates that the temporal consistency is generally quite high, generally ranging from 0.7 to 0.95; i.e., forwarding entries don't change much from state to state. There are a couple of exceptions, however. At states 45 and 51, the consistency from states 44 and 50, respectively, is much lower. This is because the first two, and then the last two of the satellite's interplane ISLs were shut down between these state transitions, causing many of the forwarding entries to change. States 57 and 61 have related changes (a neighboring satellites interplane ISLs were being deactivated). By taking advantage of this temporal consistency, a centralized routing system would only have to dedicate, on average, on the order of a few Mb/s of bandwidth to update the forwarding tables for the entire constellation.[4]

The geographic consistency is much lower than the temporal consistency, however. The main reason is that Teledesic satellites generally have two ISLs oriented towards each of the four cardinal directions. Often, the interface picked by shortest path routing is in the same direction as that picked by geographic routing, but for reasons further downstream, the best geographic next-hop is not part of the shortest path. In the Iridium constellation, where there is only one ISL in each cardinal direction, the consistency can be much higher (typically around 70%). It may be possible to relax the requirements on the geographic next-hop being an exact match with the next-hop picked by shortest path routing (e.g., allow use of the geographic next-hop if the resulting route will be within a certain delay tolerance of the optimal route). However, this is not as easy as it first seems, because care must be taken in determining that inconsistent routing decisions are not taken that result in a loop. Also, there may be recursion problems in determining whether a forwarding decision will result in a route within the delay tolerance. In summary, more investigation would be needed to establish the correctness of a routing policy that did not require an exact match between the next-hop interfaces picked by geographic forwarding and shortest-path routing. However, we conclude that there is not much advantage to be gained by pursuing this approach because the temporal consistency of the forwarding tables is already very high.

[4] Not considering the updates due to user terminals local to each satellite, which cannot be aggregated in any case.

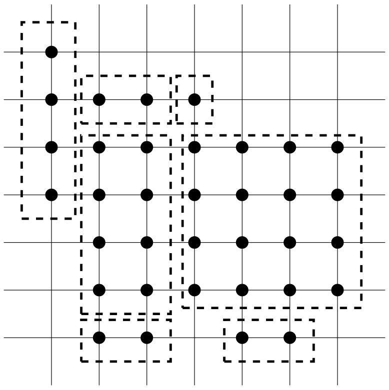


Figure 6.25: A hypothetical aggregation of 35 cells into 7 forwarding table entries.

6.6.3 Address Aggregation

In the previous subsection, we demonstrated that by taking advantage of temporal consistency in the routing tables, we can reduce the amount of message overhead between the ground and the satellites to a tolerable level. The final piece to optimize is the size of the forwarding tables. Large forwarding tables are costly in two ways: they require more memory, and they take longer to search. In the cellular geometry considered above, there are over twenty thousand cells, but only eight next-hop interfaces on a (Teledesic) satellite. We noticed, by looking at satellite routing tables generated using shortest path algorithms, that many of the cells served by the same next-hop interface were geographically contiguous. Therefore, by making use of the cell numbering scheme described above, which is optimized for aggregating geographically contiguous cells, we can reduce the number of forwarding table entries required.

Figure 6.25 illustrates an example, in which 35 contiguous cells (the solid dots on the graph) can be reduced to 7 entries. Using this representation, it can be seen that address aggregation is a variant of the classical minimum set covering problem. The *minimum set cover* problem is defined as follows: [48]:

INSTANCE: Collection C of subsets of a finite set S, positive integer $K <= |C|$.

QUESTION: Does C contain a cover (a subset $C' \subseteq C$) for S of size K or less such that every element of S belongs to at least one member of C'?

The minimum set cover problem is known to be NP-complete in the strong sense, unless all $c \in C$ satisfy $|c| <= 2$, in which case matching techniques can be used to solve the problem in polynomial time [48].

In our case, the set S is defined as the collection of cells on a rectilinear grid numbered according to the Gray code described above, and the collection C of subsets is the collection of blocks of cells ("rectangles") in S that may be aggregated into a single address/mask combination. Our problem is a set covering problem with the following additional constraints:

- Rectangles are arbitrary shapes with sizes corresponding to a non-negative integer power of two, because all bit masks cover a number of cells equal to a power of two, and

- Rectangles must fall on certain boundaries. In particular, rectangles sides of length n can fall on boundaries of every $n/2$ cells, as described above in Section 6.6.1.

Note that by framing this problem more generally as a set covering problem rather than a set packing problem (a covering by mutually disjoint rectangles), we permit a cell to be covered by more than one rectangle. The implication of this is that more than one matching entry for that cell may exist in the forwarding table.

In general, even in one dimension, packing or covering problems involving objects of different sizes are NP-complete (the "Knapsack" problem is one such example) [46]. A number of problems closely related to the address aggregation problem identified above have been shown to be NP-complete. In the context of image processing, Fowler, Paterson, and Tanimoto have shown that the planar geometric covering problem using $2x2$ squares is NP-complete [46]. We have proven above that not all possible address aggregations are geographically contiguous. If we define the optimal address aggregation as including also non-contiguous cells, then the problem is equivalent to a classical problem of Boolean logic minimization known as the *minimum sum* problem [87]. Briefly, if we consider the bits of an address to be inputs in a Boolean truth table, and we set the output of the table to be 1 if the address is in the set to be aggregated, then the solution of the minimum term Boolean sum function will yield the most optimal address aggregation. This problem can be reduced to the NP-complete problem known as 3-SAT [120].

Interestingly, if we constrain the problem to one dimension and restrict the possible address aggregations to those involving contiguous cells only, the problem can be optimally solved in polynomial time by the following greedy algorithm. The availability of a polynomial-time algorithm is specifically tied to the constraint that rectangles may only fall on a restricted set of boundaries. In the algorithm, the value $n = 2^k$ is equal to the total number of cells in the system (represented by k bits), and A is the set of cells to be aggregated, with $|A| \leq n$.

algorithm *greedy_aggregate*
begin
    $i \leftarrow n$;
    while $A \neq \emptyset$ **do**
        choose $S_i \in A$ such that S_i contains only non-overlapping, legal
          blocks of size i and $|S_i|$ is maximized;
        $A \leftarrow A \backslash \{S_i\}$;
        $i \leftarrow i/2$;
    od
end

Finding the maximum packing of blocks of size i can be made with two passes through the space of n cells. Any set of cells A will have up to $n/2$ distinct contiguous blocks of cells. Each block of contiguous cells can be aggregated into blocks of i cells, if at all, in only two ways– the boundaries on which legal blocks of i cells can fall are separated by $i/2$ cells, according to the second constraint above. Therefore, two passes through each block of contiguous cells can be used to determine which one of the two boundaries (in each contiguous block) yields the most blocks of size i among the contiguous cluster of cells. Since there are $log_2(n)$ steps to this algorithm, it runs in polynomial time with $O(n * log_2(n))$.

Theorem 6.2 *The above greedy algorithm obtains the aggregation with the fewest number of contiguous blocks in a one-dimensional space.*

Proof: Starting from the largest possible block size, the algorithm searches for and removes the maximum number of (non-overlapping) blocks of each size before reducing the block size searched. It should be clear that, given a contiguous subset of cells numbering exactly i for which an aggregation into a block of size i is permitted, there is no advantage for passing up the opportunity to aggregate this subset of cells into one block of size i. A little less obvious is the fact that there is no penalty incurred upon subsequent iterations of the algorithm for removing a block c of size i from the set. In general, this would not be the case, because removing a block of cells would further constrain the possible blocks that could be formed during later iterations. However, given the restrictions on placement of blocks on the grid, any blocks smaller than i that would have contained cells in c will have exactly half of their cells in c and half outside of c, and since any legal block greater than size one can be divided in half to form two legal blocks, we do not constrain the choices available at later stages by removing any block.

Therefore, we only need to check whether removing a block of size i is optimal when it has fewer than i contiguous adjacent neighboring cells on either or both sides (if it has more than i cells on either or both sides, it would have constituted a part of a larger block of size $2 * i$ or greater that would have been removed by a previous step of the algorithm). For simplicity, we consider the case in which the block of size i has adjacent cells to aggregage on only one side; the case in which there are cells on both sides is handled similarly. Suppose that the algorithm were not optimal; i.e., suppose that there exists a collection of contiguous cells of size greater than i for which removing a given legal block *B* of size i, and collecting the remaining adjacent contiguous cells during later stages of the algorithm into a minimal set of blocks of size less than i (resulting in a total set of blocks that we denote as S_{orig}), results in more blocks than if block *B* were not removed and the cells within the block *B* were left to be removed at a later stage, resulting in a set of blocks we denote as S_{alt}. With this set notation, we can rephrase our supposition as being that $|S_{alt}| < |S_{orig}|$ by not containing block *B* in set S_{alt}. This can only be the case if some cells in *B* were needed in the optimal aggregation to form another block of size less than i that straddled the boundary of *B*; we call this a *straddling block* B_s. If the block *B* were broken up in this fashion, the remaining cells in what would have formed block *B* must be represented by no fewer than two blocks of size less than i. Now consider the remaining cells outside of both *B* and B_s. These cells can be aggregated into a set of blocks of cardinality no less than $|S_{orig}| - 2$. This is because the block B_s can be decomposed into two smaller blocks, one exactly contained within *B* and one entirely outside of *B*, so if the cardinality of the set of blocks formed by these residual cells were less than $|S_{orig}| - 2$, we would have originally had a maximal aggregation set of cardinality less than $|S_{orig}|$. Therefore, considering that B_s is one block and the remaining cells of *B* require at least two blocks, $|S_{alt}|$ is greater than or equal to $(|S_{orig}| - 2) + 1 + 2$, which is strictly greater than $|S_{orig}|$. Therefore, the supposition is invalidated. ■

In two dimensions, the above greedy algorithm is not polynomial, nor does it guarantee an optimal solution. Moreover, the use of brute force combinatorial minimization on the entire problem is not computationally feasible because of the size of the input. Nevertheless, because the above greedy algorithm is optimal for aggregation in in one dimension and is intuitively a reasonable approach, we explored the use of this algorithm to reduce the problem into a series ($log_2(n)$) of smaller problems (at each step, the problem is to find the maximum number of non-overlapping

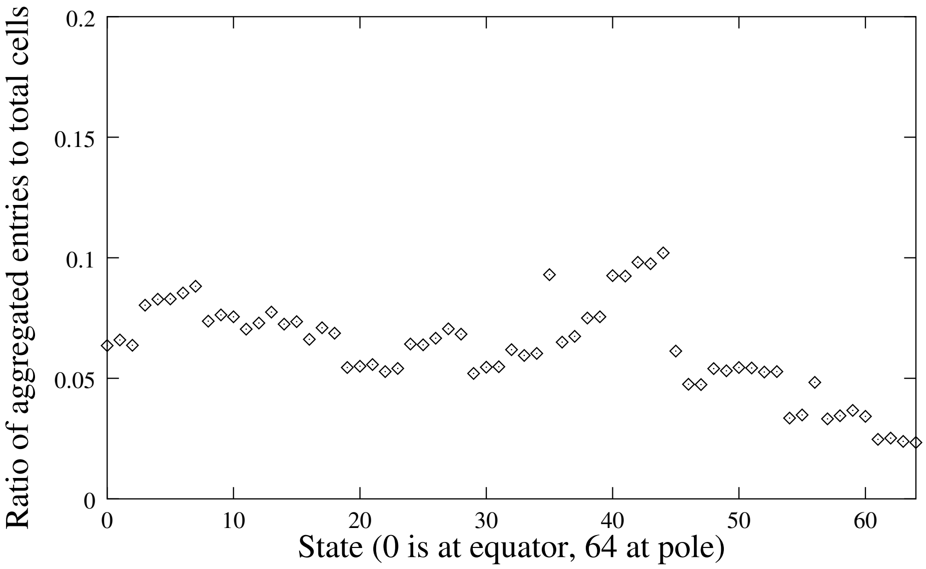


Figure 6.26: Benefit of aggregating contiguous forwarding table entries. The number of aggregated entries is roughly a tenth (or fewer) of the total number of cells that must be represented (Teledesic constellation).

blocks of size n that can be removed), and performing brute force combinatorial *maximization* on the resulting problems to take out as many large blocks as possible. While the resulting smaller problems are also factorially large, generally the input size of contiguous blocks is on the order of ten cells or less and can be computed in a very small amount of time on a contemporary PC;[5] for those blocks of cells for which the input is larger, other approximation methods such as simulated annealing may be used [114].

Figure 6.26 displays numerical results corresponding to the application of the two dimensional aggregation algorithm on the routing information previously analyzed in Figure 6.24. This example indicates that, through the aggregation described in the preceding paragraph, one is usually able to reduce the number of forwarding entries by over an order of magnitude (from over twenty thousand to a couple of thousand). Therefore, this approach slightly outperforms using temporal consistency between sequential topology configurations (which was able to reduce the routing information by a factor of three to twenty). Note that as the satellite moves closer to the poles, it has fewer next-hop satellites to consider, since some of the ISLs will be shut off. This results in less fragmentation of the set of cells to be aggregated, thereby improving aggregation.

We close this section by noting that the exploitation of temporal consistency in the routing tables and aggressive cell aggregation techniques described above are not mutually exclusive. For example, suppose that a new large entry, composed of some preexisting smaller entries in the current forwarding table, can be constructed and uploaded for the next state of a satellite's forwarding table. It may be advantageous to only upload the smaller block that "completes the puzzle" rather than

[5] In our computations, we used a 400 MHz Pentium II machine.

the new larger aggregated entry, thereby reducing the amount of bandwidth used at the expense of carrying a few more entries in the forwarding table. This may suggest that the use of very large aggregated entries that contain cells on the border of a routing region may be disadvantageous in that the large cell is likely to persist in the forwarding table for only a short time and will incur more signaling traffic in the future (i.e., some type of "persistence" metric could be added to the algorithm that constructs aggregated cell entries, giving more weight to entries that are likely to temporally persist). We did not explore further optimizations of the algorithm along these lines, but mention it as a candidate for future research.

6.7 Summary

In this chapter, we have studied the packet routing problem for LEO networks. LEO systems are sophisticated networks with a large number of degrees of freedom in the design, and therefore, in principle, there could be a wide variety of solutions to the packet routing problem. However, we based our work on the assumption that satellite communications payloads would continue to be mass and power constrained, and that bandwidth on the intersatellite communications links (ISLs) is much less scarce than that of the ground-to-satellite links (GSLs). The following are our key results:

- We described the construction of a LEO network simulator, based on the *ns* simulator, suitable for routing studies. This simulator revealed some interesting fundamental delay performance properties of LEO networks, especially pertaining to the effects of whether or not cross-seam ISLs are present in polar-orbiting constellations. Our extensions for simulating LEO networks have been incorporated into the main *ns* distribution and are now freely available.

- We explored the hypothesis that, by making locally optimal packet forwarding decisions that minimize the geographic distance to the destination, one can obtain routes that are close to optimal in terms of delay performance. We constructed a distributed routing protocol based on this hypothesis, and found that while the LEO network mesh was sufficiently dense and regular to admit good routes based on this approach (routes that were, on average, no more than 5 to 10 ms worse than globally optimal routes), there are a number of problems with commercially proposed LEO network topologies that make construction of a robust protocol difficult. In particular, the distortions in the topology in the polar regions and at the counter-rotating orbital planes require significant additions to a distributed routing protocol based on geographic addresses.

- We examined the use of geographic addressing and cell geometries for use with a centralized routing system. In this case, the objective is to reduce the amount of traffic between the centralized controller and the satellites. A key to this type of system is the concept that the state of the network evolves through a set of discrete states with fixed topologies, and that the frequency of change is not so large that it swamps the uplinks and downlinks with control traffic. We developed an optimal cell numbering scheme for rectilinear grids on the Earth's surface and proved its optimality. We then compared two approaches for reducing the amount of traffic needed to support forwarding table changes that must be periodically uploaded to satellites. For the first approach, exploiting temporal consistency in routing tables from state to state, our numerical results for a simulated Teledesic-like system indicated that, generally, 70 to 95 percent of the routing table entries persist between states. We then examined the benefit of aggregating contiguous forwarding table entries into a smaller number of entries. We developed a greedy algorithm optimal for cell aggregation in one dimension, and demonstrated that it could be used as an effective approximation algorithm for cell aggregation in two dimensions, since the problem of address aggregation in two dimensions is NP-complete. Our numerical results indicate that, with this algorithm, the size of satellite-borne forwarding tables devoted to non-local destinations can be reduced from over twenty thousand to a few thousand entries.

Chapter 7

Conclusions and Future Work

In this chapter we conclude this dissertation by summarizing our contributions and discussing directions for future work.

7.1 Summary

In this dissertation, we have focused on two problems relevant to Internet data networking over next-generation broadband satellite systems that provide "last-mile" network access: i) improving the performance of reliable transport protocols over high-latency paths, and ii) routing strategies for constellations of low-earth-orbiting satellites. In this section, we briefly review our main conclusions.

7.1.1 Transport Protocols for Broadband GEO Systems

Recall that we focused on the performance problems encountered when using TCP over GEO satellite connections. Our data supports the following conclusions:

- **TCP fairness** As described earlier, the problem of TCP fairness in a hetergeneous environment is a long-standing research problem. While we were able to reproduce the simulation results reported by Floyd [39] that showed how TCP connections, if they were to all use a "Constant-Rate" linear increase rate during congestion avoidance, could achieve better global network fairness, our simulations suggest that it would be difficult to deploy this algorithm incrementally in the network, because "Constant-Rate" connections are less aggressive in obtaining a share of congested network bandwidth. We presented some simulation results indicating that long-delay TCP connections may be able to unilaterally improve the fairness of the network by becoming slightly more aggressive during congestion avoidance. The interesting aspect of this result is not that more aggressive connections could do better but that, in our simulation topologies, such connections *always* improved the network fairness metric without compromising utilization of the bottleneck link. Based solely on our simulation results, we cannot recommend this strategy or a particular policy for deployment until it has been examined further (particularly through network experiments), but the potential seems promising. We note that the best solution to this problem appears to be the inclusion of mechanisms within routers that guarantee fair sharing of bandwidth on a per-flow basis, but it is unclear at this time whether there are sufficient incentives to deploy such mechanisms in the Internet.

- **Satellite optimized TCP implementations and split connections** We conducted a study of satellite TCP performance, focusing on end-to-end performance in which part of the satellite TCP connection traverses the Internet, and investigating both file transfer performance and short Web-like connection performance. We demonstrated how SACK loss recovery and NewReno congestion avoidance principles should be used in conjunction to achieve good TCP file transfer performance over GEO satellite links, and we explained why other types of TCP implementations often perform much worse. Nevertheless, we showed how even moderate levels of congestion in the wide-area Internet can derail the file transfer performance of even satellite-optimized TCP connections. We also quantified the performance gains that TCP enhancements such as TCP for Transactions and an increased initial window can have on Web-like traffic, showing that user-perceived latency can be reduced by a factor of two to three through such protocols. Because congestion can have such a negative impact on end-to-end satellite TCP file transfers, and because enhancements like TCP for Transactions may not be deployed in the Internet due to security concerns, we concluded that the safest bet for ensuring high performance TCP connections over GEO satellite links appears to be a split-connection approach. We demonstrated how the end-to-end performance degradations could be nearly eliminated through a split connection approach, provided that IP security protocols do not constrain deployment of such gateways.

- **Satellite-optimized transport protocol** We described the overall design and performance of a satellite-optimized transport protocol (STP) that is specifically designed for a broadband satellite network characterized by high degrees of bandwidth asymmetry. We then experimented with simulation models and a BSD/OS kernel implementation of the protocol. We compared this protocol to TCP and found that it provides very good performance in a high loss environment (even with bit error ratios as low as 10^{-5}), less sensitivity to large variations in the round trip delay experienced by packets, delay performance approaching that of TCP for Transactions for short transfers, and a reduction of up to an order of magnitude in the amount of bandwidth used on the reverse channel to return acknowledgments.

7.1.2 Unicast Packet Routing for LEO Constellations

We also studied the unicast packet routing problem for LEO constellations and concluded the following:

- **LEO extensions for the *ns* simulator** We described the design and construction of a packet-level simulator for LEO networks. This network simulator is valuable because it is integrated with the *ns* simulator already in wide use by the research community. This simulator revealed some interesting fundamental delay performance properties of LEO networks that have not yet been seen in the literature. In particular, we illustrated typical delay performance that might be seen by users of the Iridium and (proposed) Teledesic systems, and showed the impact of using hop counts, rather than delay metrics, as the cost metric used in shortest path routing algorithms.

- **Distributed LEO routing using geographic-based addresses** We explored the hypothesis, advanced by several researchers, that by making locally optimal packet forwarding decisions that minimize the geographic distance to the destination, one can obtain routes that are close to optimal in terms of delay performance. We constructed a distributed routing protocol based on this hypothesis, and found that while the LEO network mesh was sufficiently dense and regular to admit good routes based on this approach (routes that were, on average, no more than 5 to 10 ms– less than 10%– worse than globally optimal routes), there are a number of problems with commercially-proposed LEO network topologies that make construction of a robust protocol difficult. In particular, the distortions in the topology in the polar regions and at the counter-rotating orbital planes require significant additions to a distributed routing protocol based on geographic addresses.

- **Centralized LEO routing using geographic-based addresses** We examined the use of geographic addressing and cell geometries for use with a centralized routing system. We developed an optimal cell numbering scheme for rectilinear grids on the Earth's surface and proved its optimality. We then compared two approaches for reducing the amount of traffic needed to support forwarding table changes that must be periodically uploaded to satellites. For the first approach, exploiting temporal consistency in routing tables from state to state, our numerical results for a simulated Teledesic-like system indicated that, generally, 70 to 95 percent of the routing table entries persist between states. We then examined the benefit of aggregating contiguous forwarding table entries into a smaller number of entries. We developed an greedy algorithm for cell aggregation in one dimension that is optimal in aggregating contiguous cells, and demonstrated that it could be used as an effective approximation algorithm for cell aggregation in two dimensions, since the problem of address aggregation in two dimensions is NP-complete. Our numerical results suggest that, with this algorithm, the size of satellite-borne forwarding tables devoted to non-local destinations can be reduced by an order of magnitude, from tens of thousands of entries to a few thousand entries.

7.2 Software Availability

We have made available online, in source code form, most of the protocol and software implementations described herein:

- Our satellite-optimized TCP implementation (described in Appendix A) for BSD/OS 3.0 is available at `ftp://daedalus.cs.berkeley.edu/pub/tcpsack/`

- Simulator extensions for generating background HTTP traffic in the *ns* simulator are available at `ftp://daedalus.cs.berkeley.edu/pub/ns/httptrafficgen.tar`

- Source code for BSD/OS 3.0 and FreeBSD kernel implementations of the Satellite Transport Protocol (STP) are available at `ftp://daedalus.cs.berkeley.edu/pub/stp/`

- Our LEO satellite extensions to the *ns* simulator have been incorporated into the main distribution and documented. *ns* is available at `http://www-mash.cs.berkeley.edu/ns/`

7.3 Future Directions

The results of this dissertation point to several interesting directions for future work:

- **TCP fairness** Our results above indicated that it may be worthwhile to selectively increase the aggressiveness of certain TCP connections to improve network fairness. However, our results are preliminary and need to be experimentally validated before deployment can be recommended. For example, a wider range of topologies should be considered, and experiments as well as more simulations are needed to decide on the best policy. The impact of this algorithm on connections used for short data transactions (such as many small Web transfers) should be studied more. Finally, mechanisms for more accurately determining a connection's RTT, as well as policies that might be invoked as a function of the RTT observed, require more study. It should be emphasized that mechanisms for per-flow fair sharing of congested links would obviate the need for improvements to TCP's end-to-end algorithm, so further work on the design and deployment of these mechanisms would also be very useful. Also, Mo has demonstrated the potential existence of a fair, distributed flow control algorithm but has not been able to construct such an algorithm [89]; if such an algorithm were discovered, it may be a substantial improvement over the current one.

- **Satellite-optimized TCP** We have identified the performance of the slow start and congestion avoidance algorithms, as well as implementation details such as correct sizing of socket buffers and use of the correct TCP options, as the biggest hurdles to overcome to improve TCP performance over satellite links. Further work on automating the correct configuration of TCP, such as described in [122], could help the deployment of more satellite-friendly implementations.

- **Split connections** The main obstacle to the deployment of split-connection protocol gateways is their interaction with a security infrastructure. In particular, any IP security protocols that encrypt the payload of an IP packet render a split connection gateway useless. Work on how to integrate performance enhancing proxies into the trust infrastructure of a secure network would be valuable. Another issue that could be pursued further is how data flows between split connections should interact. Specifically, consider the case of using a split connection to aid in Web browsing. In this case, packets may trickle in at a slow rate from the terrestrial (server) side of a connection. If the split connection is implemented as a pure byte pipe, then these packets would be sent over the satellite as they arrive at the gateway. However, these types of short packet exchanges are the least efficient because they require frequent use of the backchannel for acknowledgments. It would be much more efficient to bundle and send the whole Web page at once rather than to stream it gradually across a satellite connection. In other words, knowing application-level data boundaries may aid in efficient communication (this is the concept of Application Level Framing). Design of split connnection gateways around this approach may outperform simple connection-splicing approaches as described herein.

- **LEO networking** The field of LEO networking is ripe for further work, as there are many more interesting issues involving unicast LEO routing than we were able to cover. For example, integration of a LEO network with terrestrial wireline or wireless networks is an interesting problem. Given a LEO network with multiple gateways, how can the network decide upon which gateway to exit the network? What kind of load balancing, such as routing around hot spots, is needed? Would alternate constellation designs admit simpler routing protocols? Can loop-free distributed routing protocols with support for load balancing be developed? The problems are not limited to unicast routing. The performance of TCP connections in a LEO environment that induces RTT variations could be studied. We did not even touch on multicast routing; what kind of multicast routing protocols are best for a LEO constellation? What are optimal queue sizes for on-board switches? We hope that our *ns* simulator extensions enable research in these areas and in other areas not touched by our research, such as multiple access protocols, handoff strategies, and the performance of other classes of applications such as reliable multicast. The publishing of additional information or research on issues such as the link availability and error performance of LEO links and the terminal and satellite hardware capabilities related to link handoff would aid such future studies greatly.

Appendix A

Congestion Avoidance and Selective Retransmission Policies for TCP

Our TCP SACK-NewReno implementation obeys standard congestion avoidance policies and rules for selective acknowledgments (SACKs) as specified in [129] and [83], with the following extensions.[1] The following extensions apply whether or not SACK is enabled for a given connection:[2]

1. Initialize a new state variable, *snd_recover*, to the value of *snd_una* upon connection start.

2. Upon receiving three duplicate acknowledgments, if the sequence number acknowledged is greater than or equal to *snd_recover*, then set *snd_recover* equal to *snd_max*, and perform fast retransmit according to [129].

3. If, while in fast recovery phase, a segment acknowledging new data is received and the sequence number acknowledged is greater than or equal to *snd_recover*, then exit fast recovery by setting *snd_cwnd* to either *snd_ssthresh* or the amount of outstanding data in the network plus one segment, whichever is smaller.

4. While in fast recovery phase, if a segment acknowledging new data is received, and the sequence number acknowledged is less than *snd_recover*, if SACK is not enabled for the connection then retransmit the next unacknowledged segment. Additionally, whether or not SACK is enabled, partially deflate the (inflated) *snd_cwnd* by the amount of new data acknowledged, add back one segment to *snd_cwnd*, and call *tcp_output()*.

In addition, if SACK is enabled for a given connection, the following rules apply to retransmissions and new data transmissions during the recovery phase:

5. A given segment is considered “eligible” for retransmission if it has not already been retransmitted and if either three duplicate acknowledgments have arrived for the segment just prior to it or the SACK information implies that the receiver is holding a segment that was sent at least three segments beyond the given segment.

[1] This description assumes a TCP implementation similar in structure to Berkeley-derived TCP implementations.

[2] These first four guidelines are for the TCP NewReno portion of the implementation and have been accepted as (experimental) RFC 2582 within the IETF [42].

6. While in fast recovery, upon reception of each ACK that does not end the fast recovery phase, the TCP sender first checks whether there are any eligible retransmissions to be sent. If so, one such retransmission is sent. If not, the TCP sender inflates *snd_cwnd* by one segment and attemps to send one or more new segments if permitted by the window.

7. When *snd_max* is greater than *snd_nxt* (e.g., following a TCP timeout), any SACK information received subsequent to the timeout is used to avoid retransmitting data for which the receiver is sending a SACK.

Bibliography

[1] W. Adams and L. Rider. Circular Polar Constellations Providing Continuous Single and Multiple Coverage Above a Specified Latitude. *Journal of Astronautical Sciences*, 35(2):155–192, April 1987.

[2] M. Allman. TCP Byte Counting Refinements. *ACM Computer Communications Review*, 29(3):14–22, July 1999.

[3] M. Allman et al. Ongoing TCP Research Related to Satellites. *Internet Draft: draft-ietf-tcpsat-res-issues-06.tx t*, March 1999.

[4] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. *Internet RFC 2414*, 1998.

[5] M. Allman, D. Glover, and L. Sanchez. Enhancing TCP Over Satellite Channels using Standard Mechanisms. *Internet RFC 2488*, 1981.

[6] M. Allman, H. Kruse, and S. Ostermann. An Application-Level Solution to TCP's Satellite Inefficiencies. *Proceedings of 1st Workshop on Satellite-Based Information Systems (WOSBIS '96)*, 1996.

[7] E. Amir. An Agent-based Approach to Multimedia Transmission over Heterogeneous Environments. *Ph.D. Thesis, University of California, Berkeley*, 1998.

[8] S. Bajaj et al. Improving Simuation for Network Research. *Technical Report, University of Southern California*, 1999.

[9] H. Balakrishnan. Challenges to Reliable Data Transport Protocols over Heterogeneous Wireless Networks. *Ph.D. Thesis, University of California, Berkeley*, 1998.

[10] H. Balakrishnan, V. Padmanabhan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. *Proceedings of First ACM/IEEE MobiCom Conference*, November 1995.

[11] H. Balakrishnan, V. Padmanabhan, and R. Katz. The Effects of Asymmetry on TCP Performance. *Proceedings of Third ACM/IEEE MobiCom Conference*, pages 77–89, September 1997.

[12] T. Berners-Lee et al. The World Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.

[13] D. Beste. Design of Satellite Constellations for Optimal Continuous Coverage. *IEEE Transactions on Aerospace and Electronic Systems*, 14(3):466–473, 1978.

[14] R. Binder et al. Crosslink Architectures for a Multiple Satellite System. *Proceedings of the IEEE*, 75(1):74–82, 1987.

[15] R. Braden. T/TCP– TCP Extensions for Transactions, Functional Specification. *Internet RFC 1644*, 1994.

[16] L. Brakmo, S. O'Malley, and L. Peterson. TCP Vegas: New Techniques for Congestion Avoidance. *Proceedings of ACM SIGCOMM '94*, pages 24–35, October 1994.

[17] L. Brakmo and L. Peterson. Performance Problems in BSD4.4. TCP. *ACM Computer Communications Review*, 25(5):69–86, October 1995.

[18] H-W. Braun. Architecture and Performance of Large Internets, Based on Terrestrial and Satellite Based Infrastructure. *Presentation at 1997 ACM Sigmetrics Conference*, 1997.

[19] K. Brayer. Packet Switching for Mobile Earth Stations Via Low-Orbit Satellite Network. *Proceedings of the IEEE*, pages 1627–36, 1984.

[20] E. Brewer et al. A Network Architecture for Heterogeneous Mobile Computing. *IEEE Personal Communications*, 5(5):8–24, October 1998.

[21] K. Brown and S. Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM Computer Communications Review*, 27(5):19–43, October 1997.

[22] P. Brunt. IRIDIUM– Overview and Status. *Space Communications*, 14(2):61–68, 1996.

[23] D. Chakraborty. Survivable Communication Concept via Multiple Low Earth Orbiting Satellites. *IEEE Transactions on Aerospace and Electronic Systems*, 25(6):879–89, 1989.

[24] H. Chang, B. Kim, C. Lee, Y. Choi, S. Min, H. Yang, and C. Kim. Topological Design and Routing for LEO Satellite Networks. *Proceedings of IEEE Globecom '95*, pages 529–35, 1995.

[25] C. Charalambous et al. Experimental and Simulation Performance Results of TCP/IP over High-Speed ATM over ACTS. *Proceedings of Int. Conf. on Communications (ICC)*, 1:72–78, 1998.

[26] C. Charalambous, V. Frost, and J. Evans. Performance of TCP Extensions on Noisy High BDP Networks. *IEEE Communications Letters (to be published)*, 1999.

[27] D. Chitre. A Selective-Repeat ARQ Scheme and its Throughput Analysis. *Proceedings of Int. Conf. on Communications (ICC)*, 3:6G4.1–6G4.6, 1982.

[28] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[29] L. Clare, C. Wang, and M. Atkinson. Multiple Satellite Networks: Performance Evaluation via Simulation. *Proceedings of MILCOM '87*, pages 404–10, 1987.

[30] D. Clark, M. Lambert, and L. Zhang. NETBLT: A Bulk Data Transfer Protocol. *Internet RFC 998*, 1987.

[31] W. Doeringer et al. A Survey of Light-Weight Transport Protocols for High-Speed Networks. *IEEE Transactions on Communications*, 38(11):2025–38, November 1990.

[32] R. Donnan. Method and System for Retransmitting Incorrectly Received Numbered Frames in a Data Transmission System. *U.S. Patent No. 4439859*, 1984.

[33] B. Doshi et al. Error and Flow Control Performance of a High Speed Protocol. *IEEE Transactions on Communications*, 41(5):707–19, May 1993.

[34] R. Durst, G. Miller, and E. Travis. TCP Extensions for Space Communications. *Wireless Networks*, 3(5):389–403, 1997.

[35] G. Dutton. Polyhedral Hierarchical Tessellations: The Shape of GIS to Come. *Geo Info Systems*, 1(2):35–42, February 1991.

[36] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *ACM Computer Communications Review*, 26(3):5–21, July 1996.

[37] R. Fielding, J. Gettys, J. Mogul, H. Prystyk, and T. Berners-Lee. Hypertext Transfer Protocol– HTTP/1.1. *Internet RFC 2068*, 1997.

[38] E. Fitzpatrick. SPACEWAY System Summary. *Space Communications*, 13(1):7–23, 1995.

[39] S. Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks, Part 1: One-way Traffic. *ACM Computer Communications Review*, 21(5):30–47, October 1991.

[40] S. Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks, Part 2: Two-way Traffic. *Unpublished draft*, 1991.

[41] S. Floyd. A Proposed Modification to TCP's Window Increase Algorithm. *Unpublished draft, cited for acknowledgement purposes only*, August 1994.

[42] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. *Internet RFC 2582*, 1999.

[43] S. Floyd and V. Jacobson. On Traffic Phase Effects in Packet Switched Gateways. *Internetworking: Research and Experience*, 3(3):115–156, September 1992.

[44] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 3(3):115–156, September 1993.

[45] C. Fossa. A Performance Analysis of the IRIDIUM Low Earth Orbit Satellite System with a Degraded Satellite Constellation. *Mobile Computing and Communications Review*, 2(4):54–61, 1997.

[46] R. Fowler, M. Paterson, and S. Tanimoto. Optimal Packing and Covering in the Plane are NP-Complete. *Information Processing Letters*, 12(3):133–37, 1981.

[47] J. J. Garcia-Lunes-Aceves. Loop-Free Routing Using Diffusing Computations. *IEEE/ACM Transactions on Networking*, 1(1):130–41, February 1993.

[48] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[49] B. Gavish and J. Kalvenes. The Impact of Satellite Altitude on the Performance of LEOS-based Communication Systems. *Wireless Networks*, 4(2):199–212, 1998.

[50] G. Gordon and W. Morgan. *Principles of Communications Satellites*. John Wiley, 1986.

[51] S. Gribble and E. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems*, December 1997.

[52] J. Grubb. IRIDIUM Overview. *IEEE Communications Magazine*, 29(11), November 1991.

[53] Z. Haas and M. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. *Proceedings of ACM Sigcomm '98*, pages 167–77, 1998.

[54] E. Hashem. Analysis of Random Drop for Gateway Congestion Control. *Report LCS TR-465, Laboratory for Computer Science, MIT, Cambridge, MA*, 1989.

[55] Y. Hashimoto and B. Sarikaya. Design of IP-based Routing in a LEO Satellite Network. *Proceedings of Third International Workshop on Satellite-Based Information Services (WOSBIS '98)*, pages 81–88, 1998.

[56] J. Heidemann. Performance Interactions Between P-HTTP and TCP Implementations. *ACM Computer Communications Review*, 27(2):65–73, April 1997.

[57] J. Heidemann, K. Obraczka, and J. Touch. Modeling the Performance of HTTP Over Several Transport Protocols. *ACM/IEEE Transactions on Networking*, 5(5):616–630, October 1997.

[58] T. Henderson. Design Principles and Performance Analysis of SSCOP: A New ATM Adaptation Layer Protocol. *ACM Computer Communications Review*, 25(2):47–59, April 1995.

[59] T. Henderson and R. Katz. Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks. *Proceedings of 2nd Workshop on Satellite-Based Information Systems (WOSBIS '97)*, 1997.

[60] J. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. *Proceedings of ACM SIGCOMM '96 Conference*, pages 270–280, 1996.

[61] D. Hoffman. personal communication, 1998.

[62] Y. Hubbel. A Comparison of the IRIDIUM and AMPS Systems. *IEEE Network*, 11(2):52–59, March 1997.

[63] Internet Software Consortium. Domain Survey. *http://www.isc.org*, 1999.

[64] ITU-T Recommendation Q.2110. B-ISDN Signaling ATM Adaptation Layer– Service Specific Connection Oriented Protocol (SSCOP), 1994.

[65] I. Jacobs, R. Binder, and E. Hoversten. General Purpose Packet Satellite Networks. *Proceedings of the IEEE*, 66(11):1448–67, 1978.

[66] V. Jacobson. Congestion Avoidance and Control. *Proceedings of ACM SIGCOMM '88 Conference*, pages 314–329, 1988.

[67] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. *Internet RFC 1323*, 1992.

[68] A. Jamalipour. *Low Earth Orbital Satellites for Personal Communication Networks*. Artech House, 1998.

[69] D. Johnson and D. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications*, 3(1):34–42, February 1996.

[70] R. Katz. Satellites and the Next Generation Internet. *Keynote Address, Second International Workshop on Satellite-Based Information Services (WOSBIS '97)*, 1997.

[71] H. Keller and H. Salzwedel. Link Strategy for the Mobile Satellite System Iridium. *Proceedings of the 1996 IEEE Vehicular Technology Conference*, 2:1220–1224, 1996.

[72] S. Keshav. REAL: A Network Simulator. *Technical Report 88/472, University of California, Berkeley*, 1988.

[73] T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[74] T. Lakshman, U. Madhow, and B. Suter. Window-based Error Recovery and Flow Control with a Slow Acknowledgment Channel: A Study of TCP/IP Performance. *Proceedings of INFOCOM '97*, pages 1199–1209, 1997.

[75] R. Leopold and A. Miller. The IRIDIUM Communications System. *IEEE Potentials*, 12(2):6–9, April 1993.

[76] D. Lin and R. Morris. Dynamics of Random Early Detection. *Proceedings of ACM Sigcomm '97*, pages 127–37, 1997.

[77] M. Liron. U.S. Patent No. 5740164: Traffic Routing for Satelite Communication System, 1998.

[78] G. Lundy and H. Tipici. Specification and Analysis of the SNR High-Speed Transport Protocol. *IEEE Transactions on Networking*, 2(5):483–96, October 1994.

[79] B. Mah. An Empirical Model of HTTP Network Traffic. *Proceedings of INFOCOM '97*, 1997.

[80] A. Mankin. Random Drop Congestion Control. *Proceedings of ACM SIGCOMM '90 Conference*, pages 1–7, 1990.

[81] G. Maral and M. Bousquet. *Satellite Communications Systems*. John Wiley, 1993.

[82] G. Maral, J. de Ridder, B. Evans, and M. Richharia. Low Earth Orbit Satellite Systems for Communications. *International Journal of Satellite Communications*, pages 209–25, 1991.

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

[84] R. Mauger. QoS Guarantees for Multimedia Services on a TDMA-Based Satellite Network. *IEEE Communications Magazine*, 35(7):56–65, July 1997.

[85] S. McCanne. Private communications, 1997.

[86] S. McCanne and V. Jacobson. The BSD Packet Filter: A New Architecture for User-Level Packet Capture. *Proceedings of the 1993 Winter USENIX Conference*, pages 259–69, 1993.

[87] E. McCluskey. Minimization of Boolean Functions. *Bell Systems Technical Journal*, 35:1417–44, November 1956.

[88] K. Mills, D. Chitre, H. Chong, and A. Agarwal. A Joint COMSAT/NBS Experiment on Transport Protocol. *Proceedings of 7th International Conference on Digital Satellite Communications (ICDSC)*, May 1986.

[89] J. Mo and J. Walrand. Fair End-to-End Window-based Congestion Control. *Proceedings of SPIE '98 International Symposium on Voice, Video, and Data Communications*, 1998.

[90] J. Mogul and S. Deering. Path MTU discovery. *Internet RFC 1191*, 1990.

[91] J. Mogul and J. Postel. Internet Standard Subnetting Procedure. *Internet RFC 950*, 1985.

[92] M-J. Montpetit. personal communication, 1998.

[93] S. Murthy and J. J. Garcia-Lunes-Aceves. An Efficient Routing Protocol for Wireless Networks. *Journal of Special Topics in Mobile Networks and Applications (MONET)*, 1(2):183–97, October 1996.

[94] J. Musey. Presentation. *29th Annual Banc of America Securities Investment Conference, San Francisco*, September 1999.

[95] A. Myles, D. Johnson, and C. Perkins. A Mobile Host Protocol Supporting Route Optimization and Authentication. *IEEE Journal on Selected Areas in Communications*, 13(5):839–49, June 1995.

[96] S. Nanda, R. Ejzak, and B. Doshi. A Retransmission Scheme for Circuit-Mode Data on Wireless Links. *IEEE Journal on Selected Areas in Communications*, 12(8):1338–52, October 1994.

[97] A. Netravali, W. Roome, and K. Sabnani. Design and Implementation of a High-Speed Transport Protocol. *IEEE Transactions on Communications*, 38(11):2010–24, November 1990.

[98] M. Noakes et al. An Adaptive Link Assignment Algorithm for Dynamically Changing Topologies. *IEEE Transactions on Communications*, pages 694–706, 1993.

[99] V. Padmanabhan. Addressing the Challenges of Web Data Transport. *Ph.D. Thesis, University of California, Berkeley*, 1998.

[100] V. Padmanabhan et al. Networking Using Direct Broadcast Satellite. *Proceedings of the First International Workshop on Satellite-Based Information Services (WOSBIS '96)*, 1996.

[101] V. Padmanabhan and R. Katz. TCP Fast Start: A Technique for Speeding Up Web Transfers. *Proceedings of IEEE Globecom '98 Internet Mini-Conference*, 1998.

[102] V. Padmanabhan and J. Mogul. Improving HTTP Latency. *Proceedings of the Second International World Wide Web Workshop*, October 1994.

[103] E. Papapetrou, I. Gragopoulos, and F. Pavlidou. Performance Evaluation of LEO Satellite Constellations with Inter-Satellite Links under Self-Similar and Poisson Traffic. *International Journal of Satellite Communications*, 17(1):51–64, 1999.

[104] C. Partridge and T. Shepard. TCP Performance over Satellite Links. *IEEE Network*, 11(5):44–49, September 1997.

[105] B. Pattan. *Satellite-Based Cellular Communications*. McGraw Hill, 1998.

[106] D. Patterson and M. Liron. U.S. Patent No. 5796715: Non-Blocking Dynamic Fast Packet Switch for Satellite Communication System, 1998.

[107] D. Patterson and M. Sturza. U.S. Patent No. 5408237: Earth-Fixed Cell Beam Management for Satellite Communication System, 1995.

[108] V. Paxson. Automated Packet Trace Analysis of TCP Implementations. *Proceedings of ACM SIGCOMM '97 Conference*, pages 167–180, 1997.

[109] R. Perlman. *Interconnections: Bridges and Routers*. Addison Wesley, 1992.

[110] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 1996.

[111] J. Postel. Transmission Control Protocol. *Internet RFC 793*, 1981.

[112] S. Pratt et al. An Operational and Performance Overview of the IRIDIUM Low Earth Orbit Satellite System. *IEEE Communications Surveys*, 1(3):2–10, 1999.

[113] T. Pratt and C. Bostian. *Satellite Communications*. John Wiley, 1986.

[114] W. Press et al. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[115] J. Proakis. *Digital Communications*. McGraw Hill, 1995.

[116] M. Rahnema. U.S. Patent No. 5430729: Method and Apparatus for Adaptive Directed Route Randomization and Distribution in a Richly Connected Communication Network, 1995.

[117] M. Rahnema. U.S. Patent No. 5596722: Packet Routing System and Method for Achieving Uniform Link Usage and Minimizing Link Load, 1997.

[118] J. Restrepo and G. Maral. Cellular Geometry for World-Wide Coverage by non-GEO Satellites using 'Earth-Fixed Cell' Technique. *Space Communications*, 14:179–189, 1996.

[119] N. Samaraweera. Return Link Optimization for Internet Service Provision using DBS-S Networks. *ACM Computer Communications Review*, 29(3):4–13, July 1999.

[120] A. Sangiovanni-Vincentelli. Private communications, 1999.

[121] S. Segaller. *Nerds 2.0.1– A Brief History of the Internet*. TV Books, L.L.C., 1998.

[122] J. Semke, J. Mahdavi, and M. Mathis. Automatic TCP Buffer Tuning. *Proceedings of ACM SIGCOMM '98*, pages 315–23, October 1998.

[123] K. Seo et al. Distributed Testing and Measurement across the Atlantic Packet Satellite Network (SATNET). *Proceedings of ACM Sigcomm '88*, pages 235–46, 1988.

[124] N. Shacham. Protocols for Multi-Satellite Networks. *Proceedings of IEEE MILCOM '88*, pages 501–505, 1988.

[125] S. Shenker. Fundamental Design Issues for the Future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, September 1995.

[126] W. Simpson. The Point-to-Point Protocol (PPP). *Internet RFC 1548*, 1993.

[127] W. Stevens. *TCP/IP Illustrated, Volume 1*. Addison Wesley, 1994.

[128] W. Stevens. *TCP/IP Illustrated, Volume 3*. Addison Wesley, 1996.

[129] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. *Internet RFC 2001*, 1997.

[130] M. Sturza. The Teledesic Satellite System. *Proceedings of 1994 IEEE National Telesystems Conference*, pages 123–126, 1994.

[131] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury. Design Considerations for Supporting TCP with Per-flow Queueing. *Proceedings of INFOCOM '98*, pages 299–306, 1998.

[132] J. Touch. TCP Control Block Interdependence. *Internet RFC 2140*, 1997.

[133] P. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks. *Proceedings of ACM Sigcomm '88*, pages 35–42, 1988.

[134] E. Tuck et al. The Calling Network. *International Journal of Satellite Communications*, 12(1):45–61, 1994.

[135] H. Uzunalioglu. A Connection Handover Protocol for LEO Satellite Networks. *Proceedings of ACM MobiCom '97*, pages 204–14, 1997.

[136] H. Uzunalioglu. Probabilistic Routing Protocol for Low Earth Orbit Satellite Networks. *Proceedings of IEEE ICC '98*, pages 89–93, 1998.

[137] T. Watts. Telephony, Internet, and Broadband over Satellite. *Invited talk, Global Opportunities in Internet via Satellite Conference, Washington D.C.*, May 1999.

[138] M. Werner. A Dynamic Routing Concept for ATM-Based Satellite Personal Communication Networks. *IEEE Journal on Selected Areas in Communications*, 15(8):1636–48, 1997.

[139] M. Werner, C. Delucchi, H-J. Vogel, G. Maral, and J. de Ridder. ATM-Based Routing in LEO/MEO Satellite Networks with Intersatellite Links. *IEEE Journal on Selected Areas in Communications*, 15(1):69–82, 1997.

[140] M. Werner, A. Jahn, E. Lutz, and A Bottcher. Analysis of System Parameters for LEO/ICO-Satellite Communication Networks. *IEEE Journal on Selected Areas in Communications*, 13(2):371–81, 1995.

[141] D. Wetherall and C. Lindblad. Extending Tcl for Dynamic Object-Oriented Programming. *Proceedings of the Tcl/Tk Workshop*, July 1995.

[142] L. Wood. *Network Performance of Non-Geostationary Constellations Equipped with Intersatellite Links*. Masters thesis, University of Surrey, 1995.

[143] L. Wood. Big LEO Tables. *http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/tables/tables.html*, 1999.

[144] G. Wright and W. Stevens. *TCP/IP Illustrated, Volume 2*. Addison Wesley, 1995.

[145] XTP Forum. The XTP 4.0 Specification, 1995.

[146] Y. Zhang, D. DeLucia, B. Ryu, and S. Dao. Satellite Communications in the Global Internet: Issues, Pitfalls, and Potential. *Proceedings of INET '97*, June 1997.

[147] Y. Zhang, E. Yan, and S. Dao. A Measurement of TCP over Long-Delay Network. *Proceedings of 6th Int'l Conference on Telecommunication Systems, Modelling, and Analysis*, March 1998.