Chapter 2

Background and Related Work

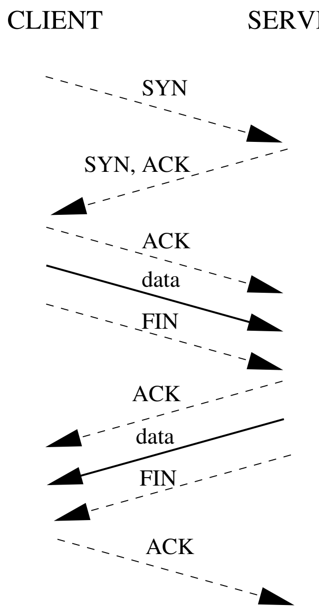
In this chapter we survey work related to our own, both to point out the many contributions of previous researchers and to place our contributions in the proper context. We organize this survey around the two main themes of our research on networking over broadband satellite systems: *reliable transport 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

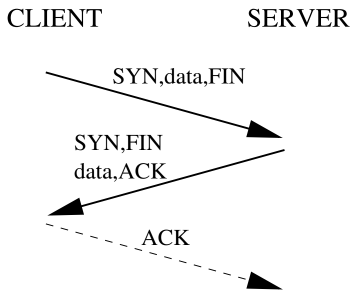
The Internet is a *best effort* network, which means that packets are neither guaranteed to arrive at the intended destination at all, nor guaranteed to arrive at the destination in the order that they were sent. This fundamental design feature of the Internet has allowed it to scale well, because reliability 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,

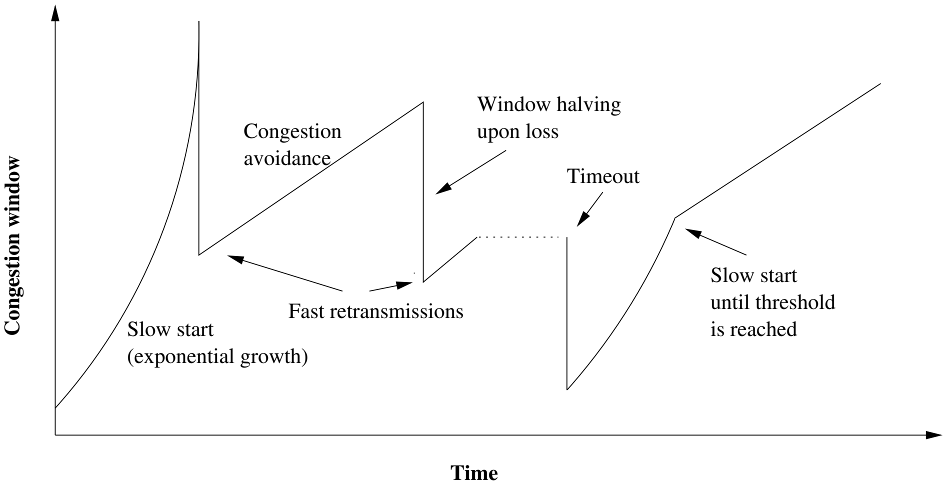


Figure 2.2: Basic operation of TCP slow start and congestion avoidance (from [9]).

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.

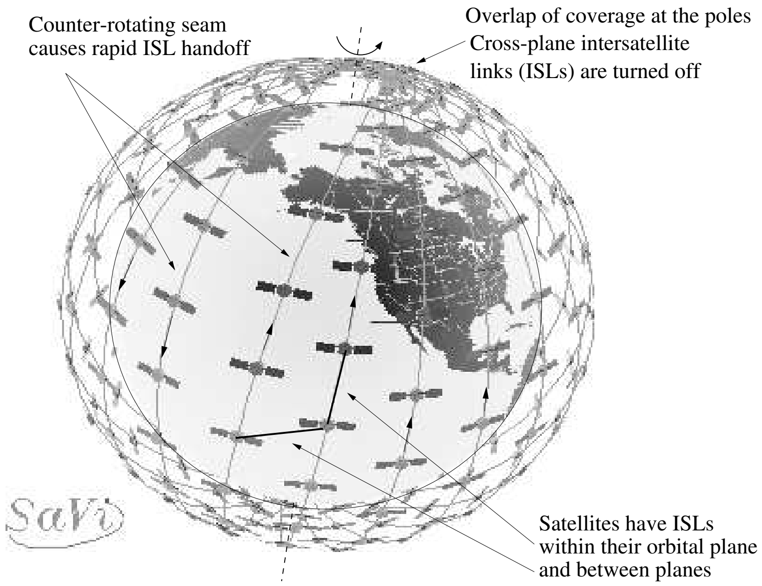


Figure 2.3: Example of a polar-orbiting satellite constellation. The figure (and subsequent SaVi-labelled figures) was generated using the SaVi software developed by the Geometry Center at the University of Minnesota.

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

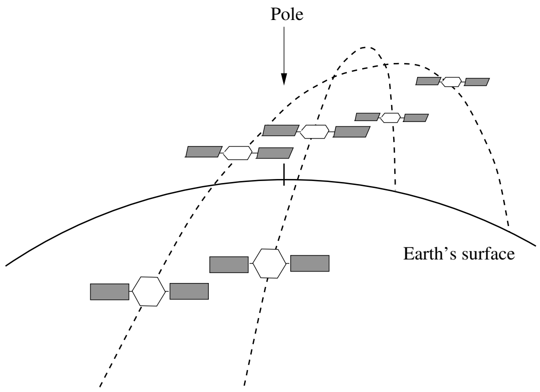
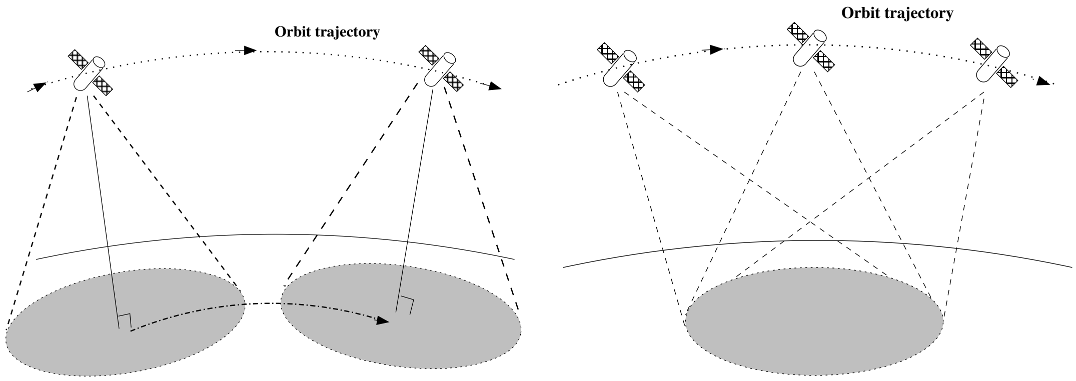


Figure 2.4: Illustration of how orbital planes intersect near the poles. A minimal amount of orbital eccentricity guarantees that one orbit passes over the other and no collisions occur.

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.