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 mul-

tiplicity 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

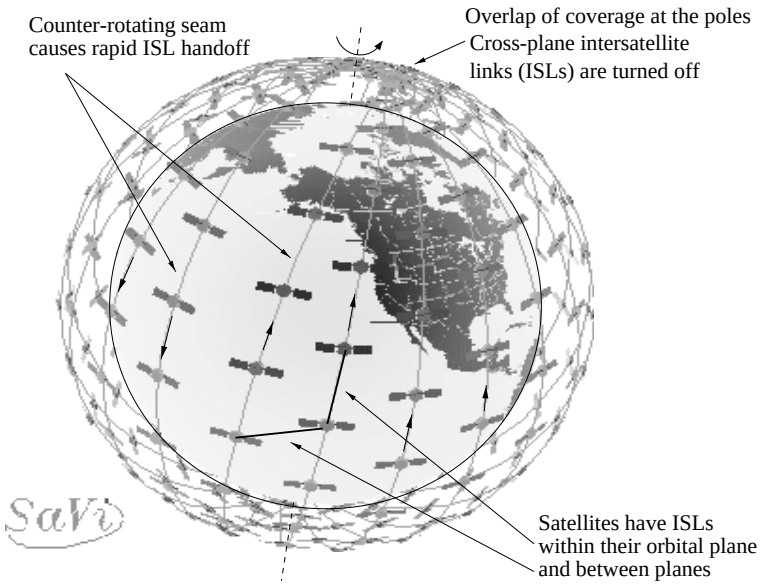


Figure 6.1: Example of a polar-orbiting satellite constellation (figure reproduced from Section 2.2).

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

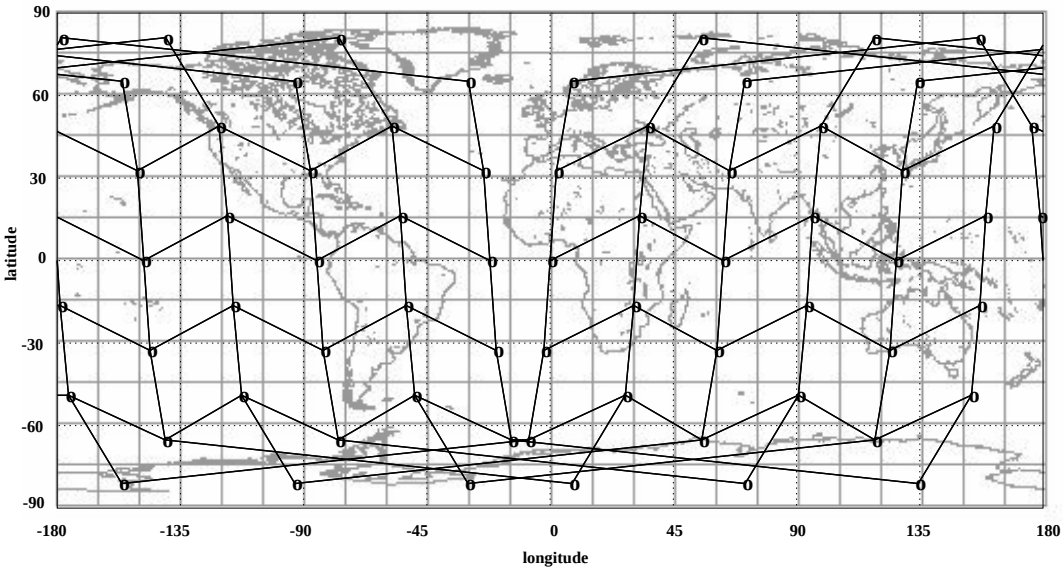


Figure 6.2: Snapshot of the Iridium constellation, illustrating active ISLs.

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

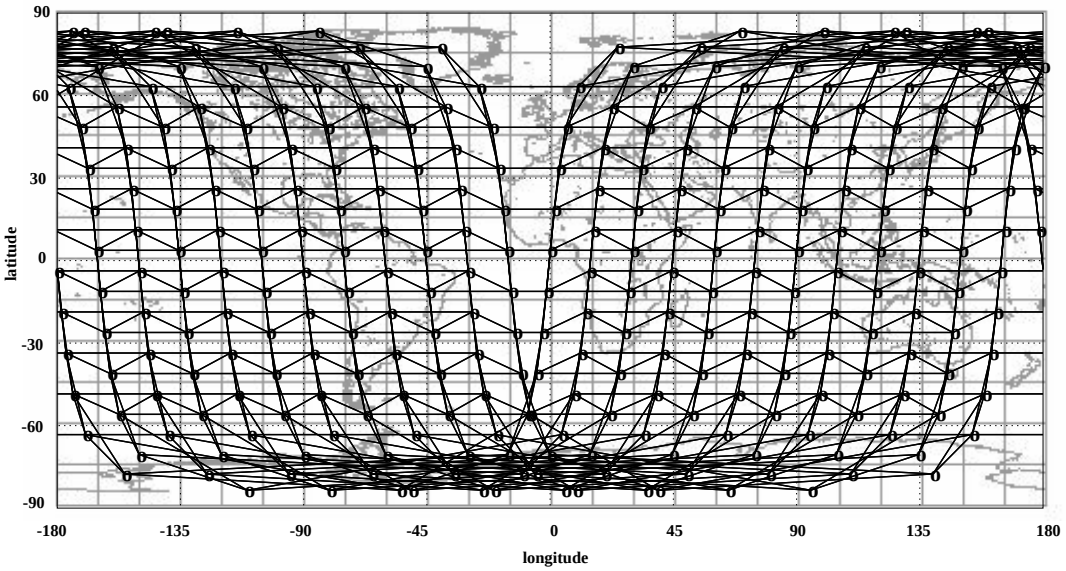


Figure 6.3: Snapshot of the Teledesic constellation, illustrating active ISLs.

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.

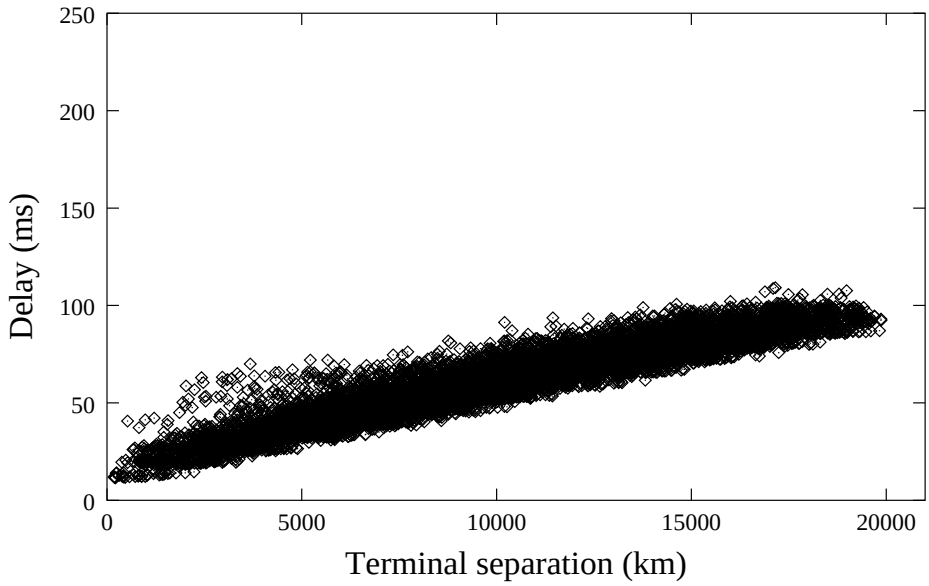


Figure 6.4: Scatter plot of the one-way delay experienced by 10,000 different pings between random locations on the Earth's surface, when global min-delay shortest path routing is used (Teledesic constellation configuration).

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

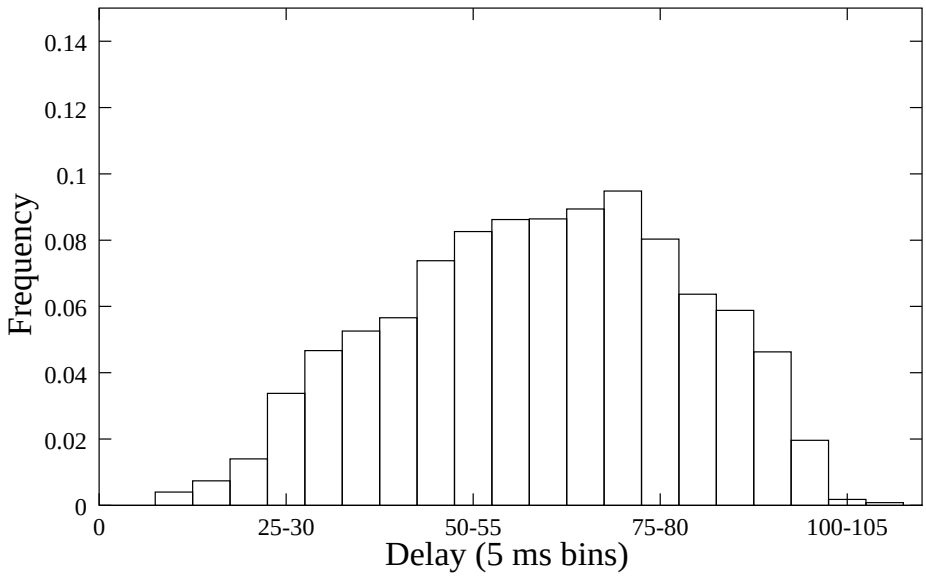


Figure 6.5: Histogram of values corresponding to Figure 6.4. Fewer than 1% of the delays exceeded 100 ms.

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

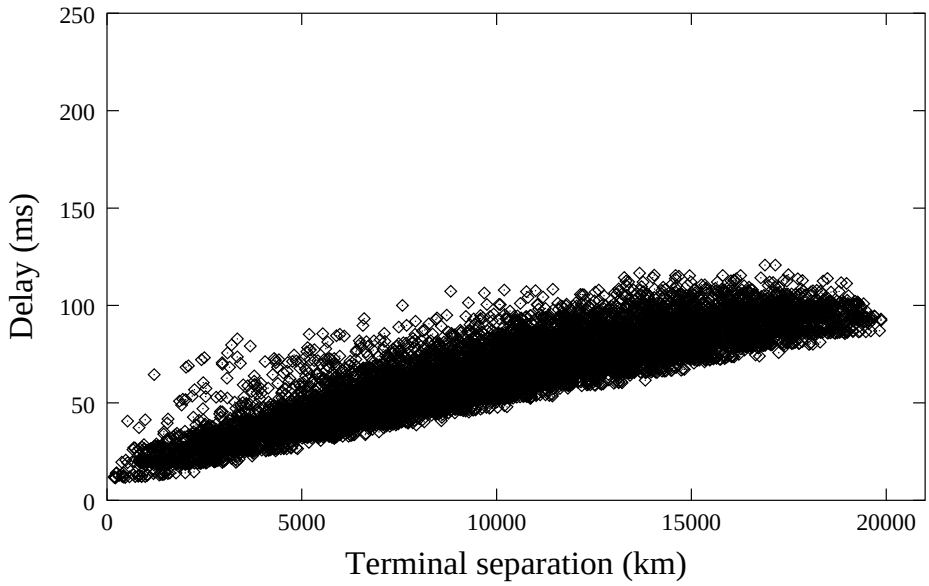


Figure 6.6: Scatter plot of the one-way delay experienced by 10,000 different pings between random locations on the Earth's surface, when global min-hop shortest path routing is used (Teledesic constellation configuration).

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.

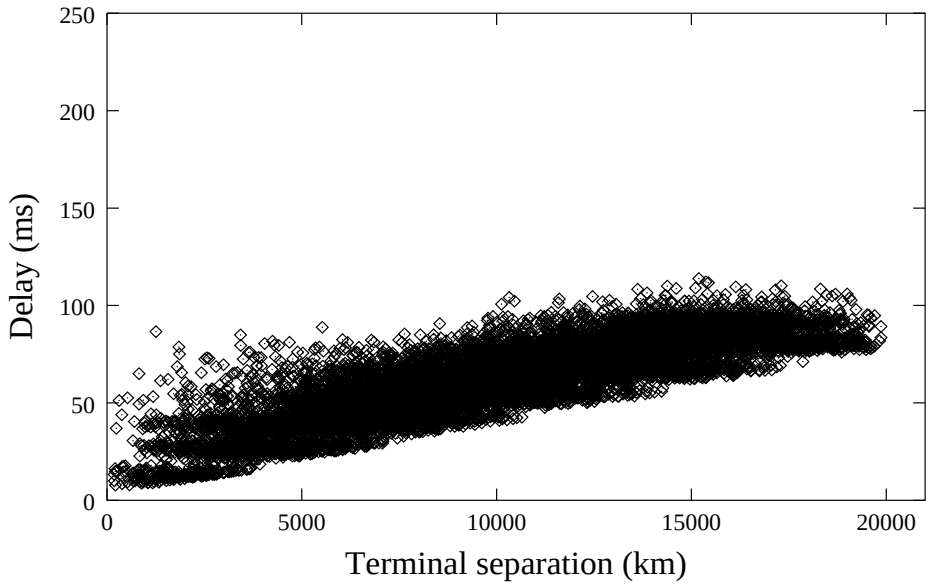


Figure 6.7: Scatter plot of the delay experienced by 10,000 different pings, when global min-delay shortest path routing is used (Iridium constellation).

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.

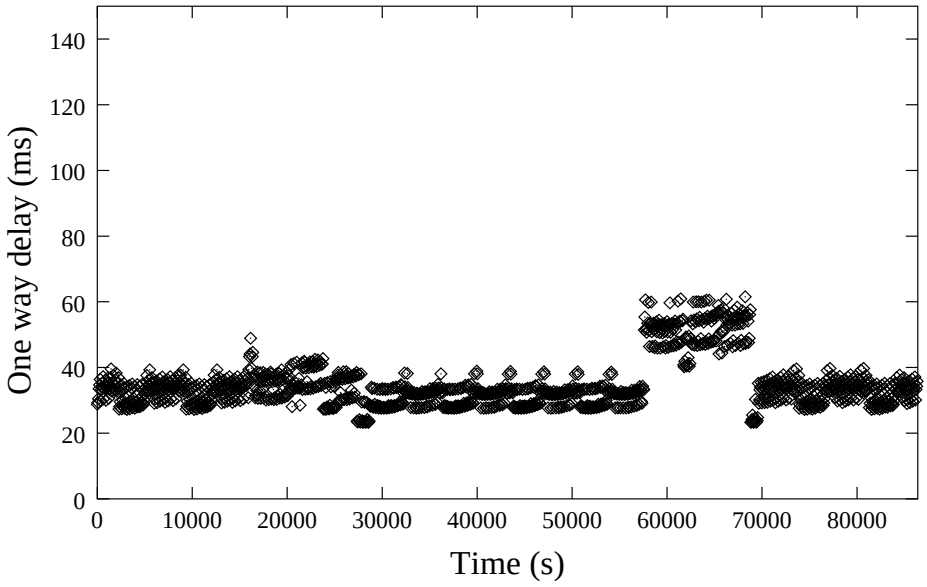


Figure 6.8: Delay variation between New York and San Francisco over the course of one day, for the Teledesic constellation.

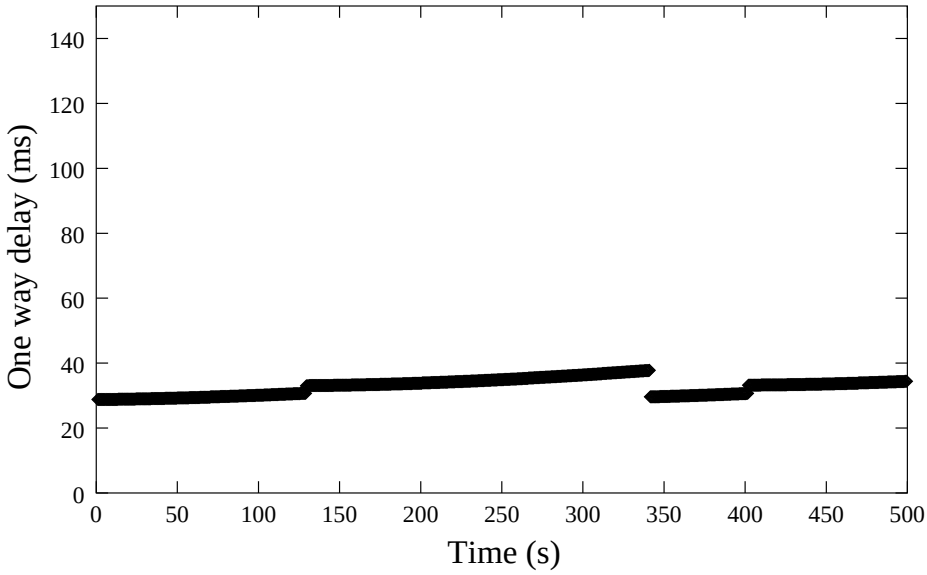


Figure 6.9: A view of the previous plot at a smaller timescale. End-to-end delays are characterized by slow variations over tens of seconds punctuated by step increases and decreases in the delay of generally no more than 8 ms.

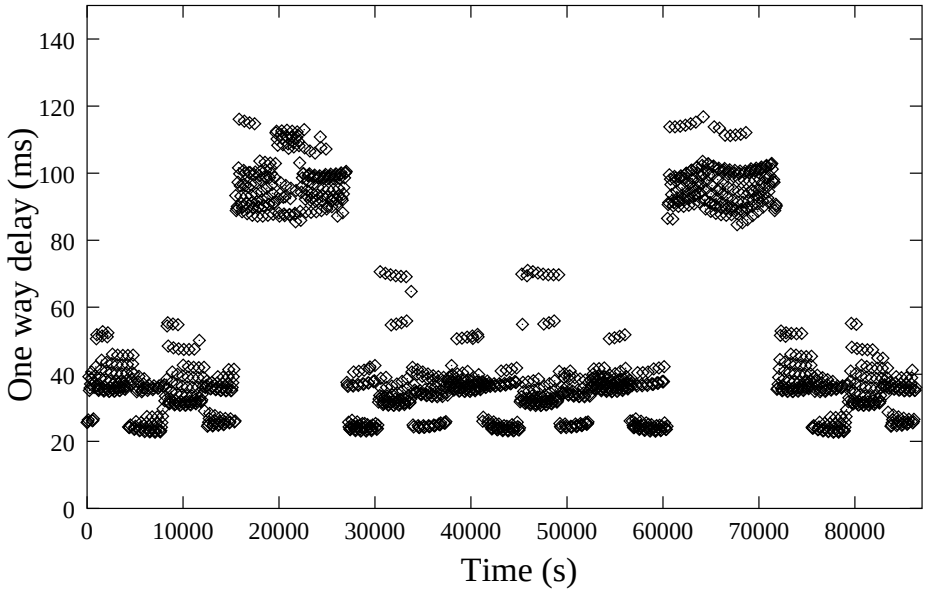


Figure 6.10: Delay variation between New York and San Francisco over the course of one day, for the Iridium constellation without cross-seam ISLs.

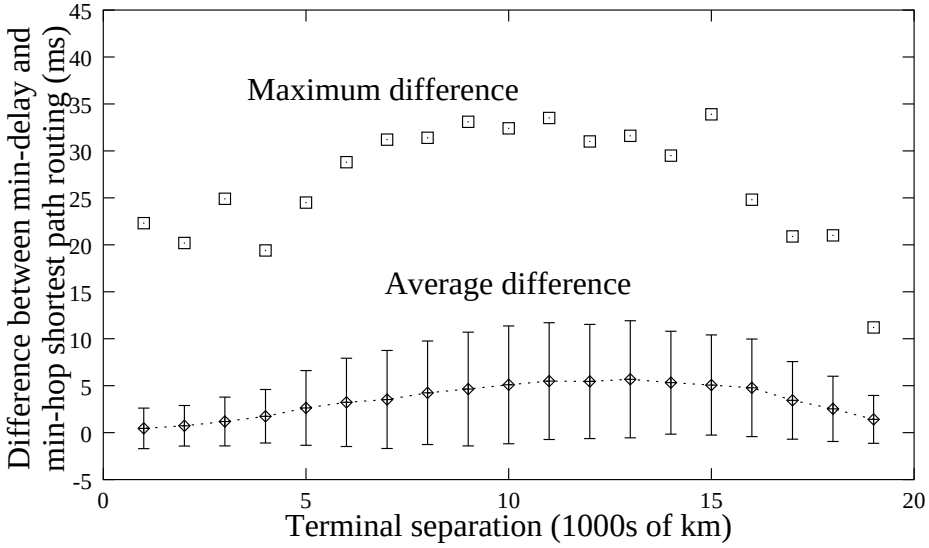


Figure 6.11: Average and maximum delay difference between min-hop and min-delay shortest path routing, as a function of the great-circle distance between terminals (Teledesic constellation).

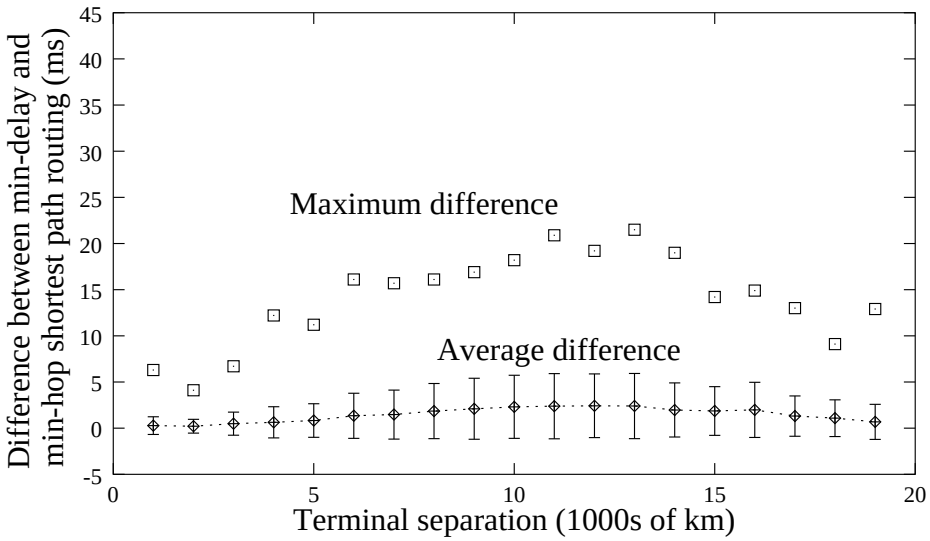


Figure 6.12: Average and maximum delay difference between min-hop and min-delay shortest path routing, as a function of the great-circle distance between terminals (Iridium constellation).

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

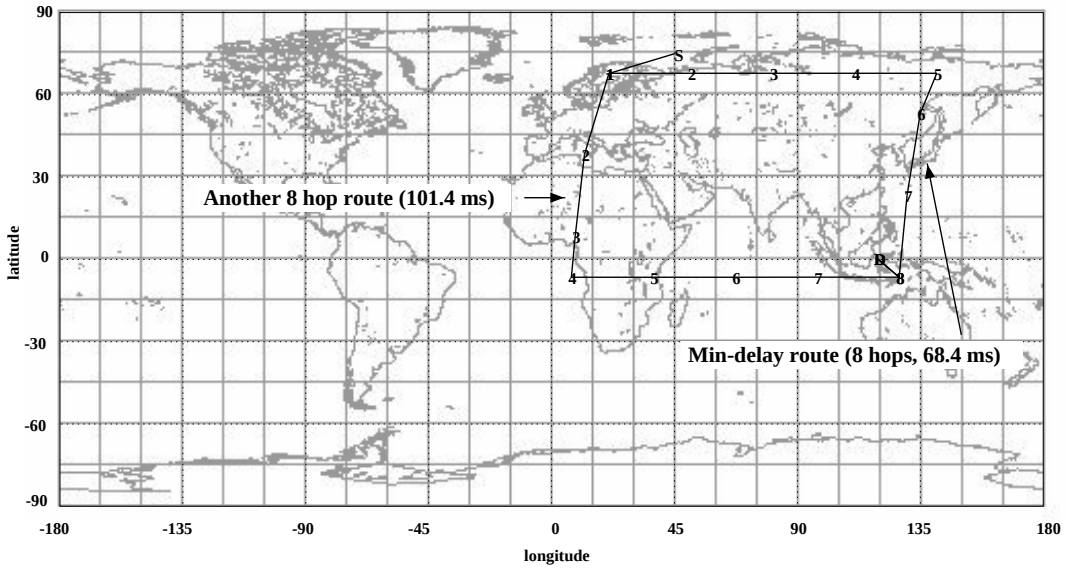


Figure 6.13: An example of how min-hop routing can occasionally select a route with a much larger delay than that of min-delay routing. Both routes contain eight satellite hops, but one route uses hops in the higher latitudes which incur much less propagation delay.

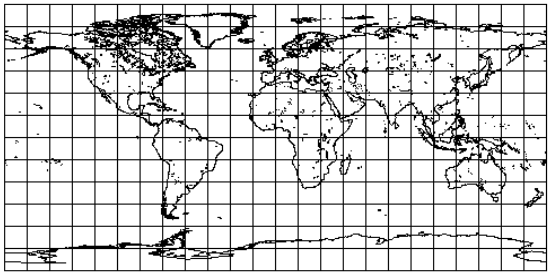
satellites and hence, fewer candidate paths to choose from.

6.4 Geographic Addressing and Cellular Geometry

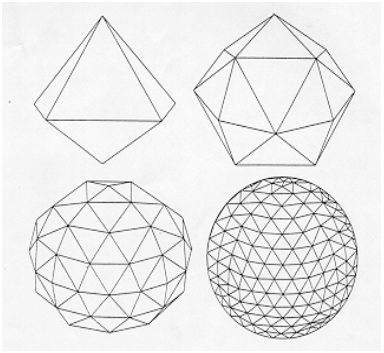
The results presented in the previous section illustrate the delay performance of the LEO constellations using an omniscient, centralized routing agent, running shortest-path algorithms, that immediately updates each node's forwarding table upon a topology change. As such, these results essentially bound 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

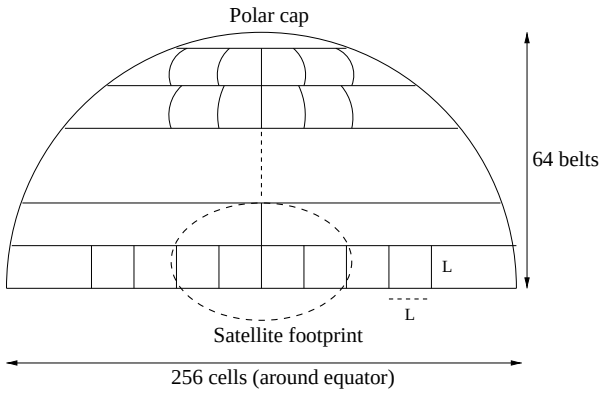


Figure 6.15: A cellular geometry consisting of roughly equal-sized trapezoidal cells [118].

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.

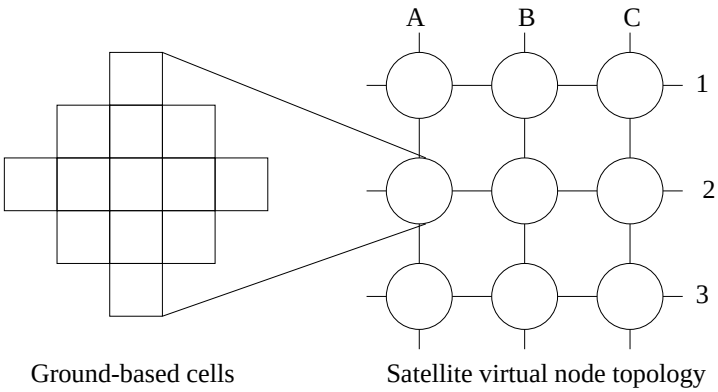


Figure 6.16: A logical network topology: fixed zones on the Earth's surface are assigned a logical address, and a satellite serving a particular zone embodies the logical node serving that region (from [84]).

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.

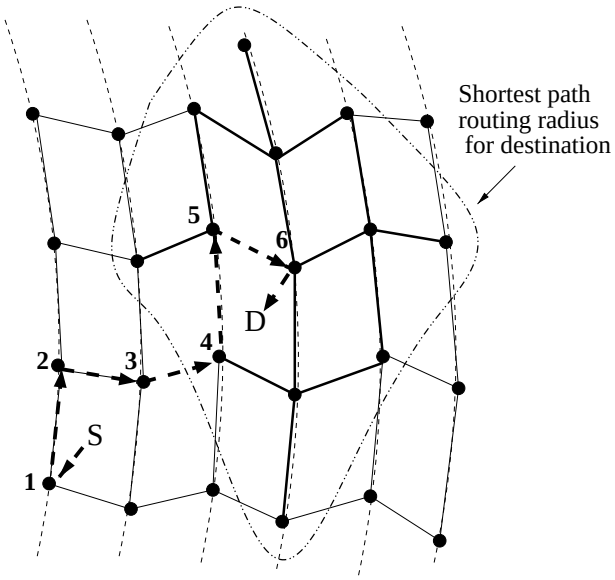


Figure 6.17: Hybrid routing strategy based on geographic packet forwarding for distant destinations and locally-scoped shortest path routing for local destinations. The figure denotes a subgraph of the satellite mesh and a hypothetical packet trace. A packet sourced at S is forwarded based on geographic 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.

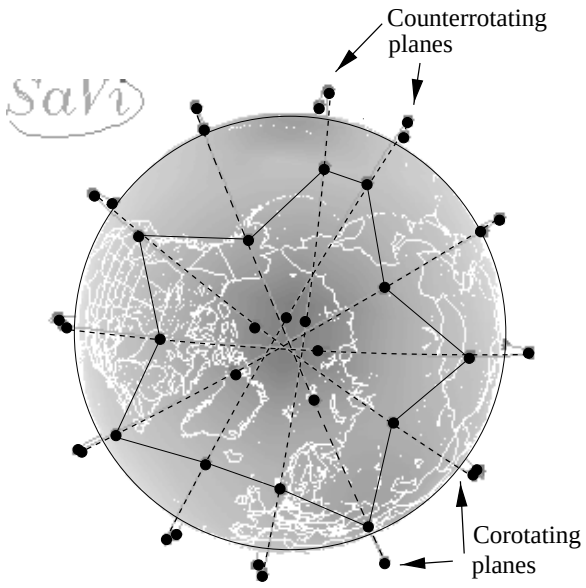


Figure 6.18: View of the Iridium topology above the North pole. Satellites closest to the pole have interplane ISLs turned off. The "polar region" is bounded by the set of satellites closest to the pole that have all of their interplane ISLs active.

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,

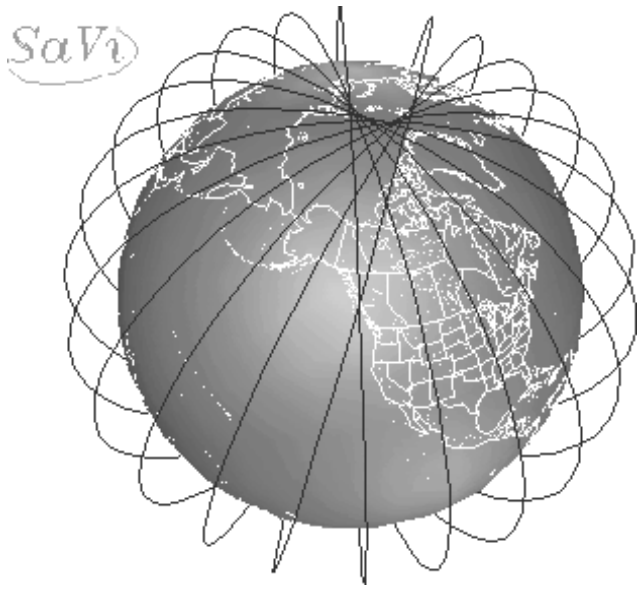


Figure 6.19: Illustration of the intersection of counter-rotating planes.

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

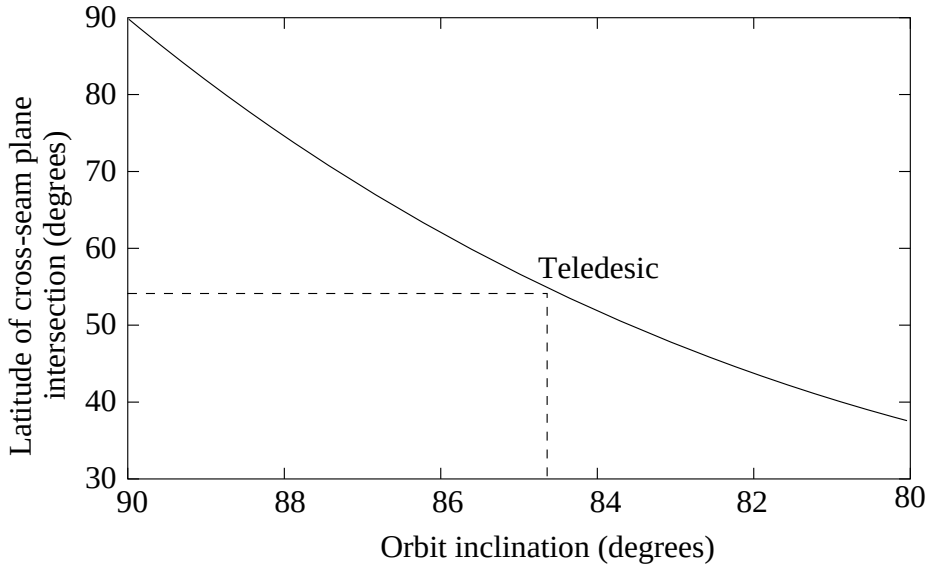


Figure 6.20: An illustration of how deviations from pure polar orbits cause the latitude at which the counter-rotating planes intersect to degrade. This plot assumes a 15^o degree plane separation such as used in the Teledesic design.

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

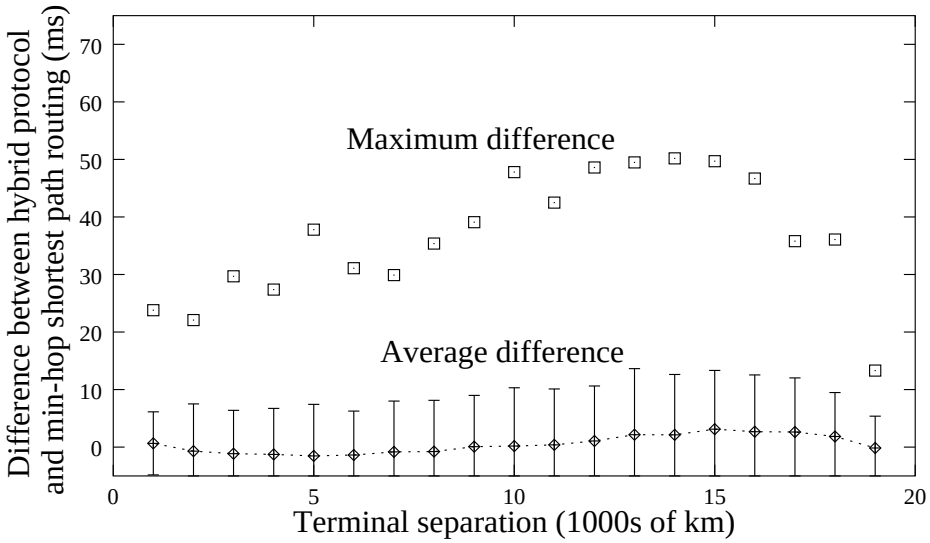


Figure 6.21: Average and maximum delay difference between using geographic forwarding and minimum-hop shortest path routing as a function of terminal separation (Teledesic constellation). Error bars denote one sample standard deviation from the sample mean.

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,

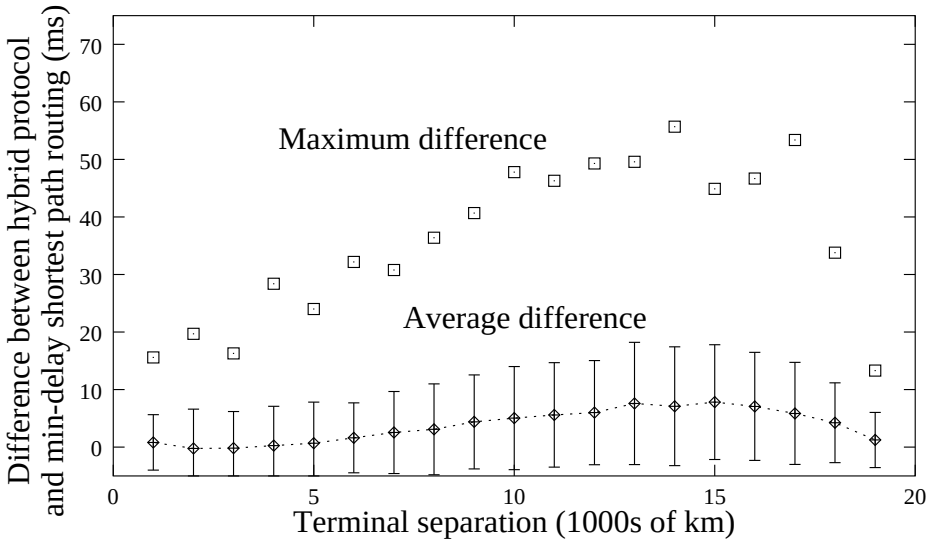


Figure 6.22: Average and maximum delay difference between using geographic forwarding and minimum-delay shortest path routing as a function of terminal separation (Teledesic constellation). Error bars denote one sample standard deviation from the sample mean.

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

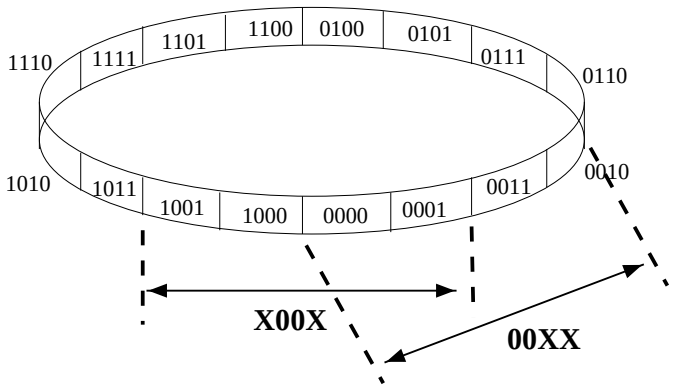


Figure 6.23: Example of the cellular numbering strategy in one dimension. Two potential size-4 aggregations are illustrated.

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

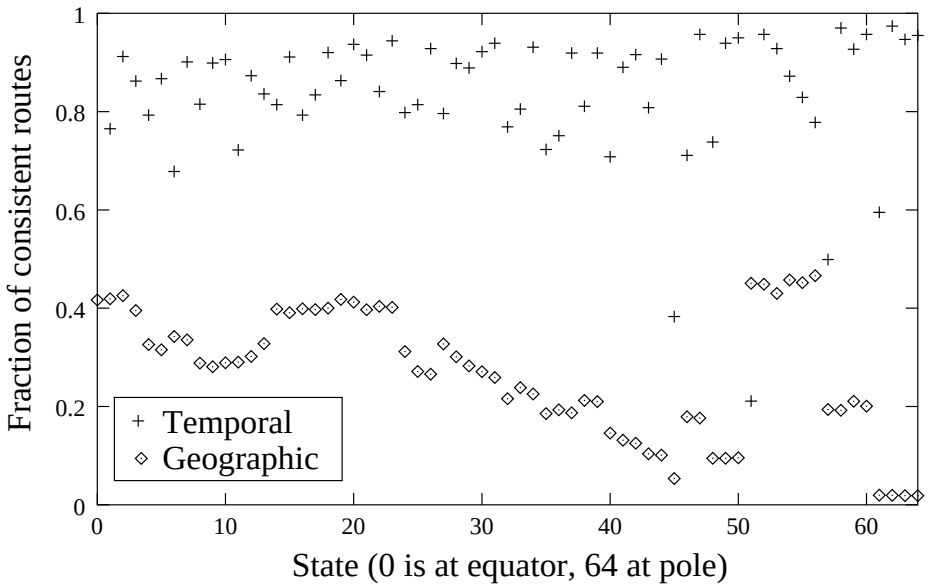


Figure 6.24: Comparison of temporal and geographic consistency of forwarding tables across topology states (Teledesic constellation).

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.

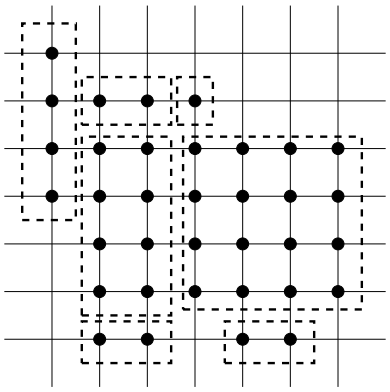


Figure 6.25: A hypothetical aggregation of 35 cells into 7 forwarding table entries.

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 ← n;
    while A ≠ ∅ do
        choose S_i ∈ A such that S_i contains only non-overlapping, legal
          blocks of size i and |S_i| is maximized;
        A ← A\{S_i};
        i ← 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

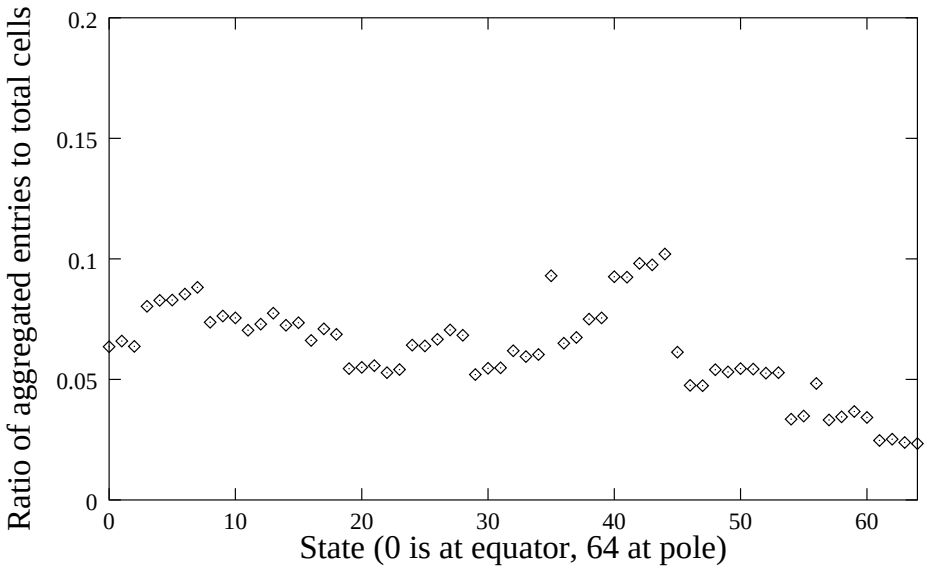


Figure 6.26: Benefit of aggregating contiguous forwarding table entries. The number of aggregated entries is roughly a tenth (or fewer) of the total number of cells that must be represented (Teledesic constellation).

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.