A Routing Solution for a Global, Space-based Wireless Multimedia System

M. Guizani and I. G. Schneller
University of West Florida
11000 University Parkway
Pensacola, FL 32514

***Abstract* — This paper focuses on the Teledesic satellite constellation, which is an attempt to provide high speed, multi-media networking connectivity to any location on the globe. Since the routers (satellites) are in constant motion, this venture is unique and leads to a one-of-a-kind data routing scheme. This concept is still in the design phase, with implementation still a few years away. However, several large companies and capital investors have already committed funds to this venture. Supporters are claiming a full operational capability in roughly 3 years. Our focus in this paper is to develop a routing algorithm for the Teledesic satellite constellation and to model its operation in a simulator. The algorithm is conceptually developed and implemented using a software simulation. Our results, which indicate this algorithm is a viable solution to the routing problem, and our implementation details are presented below.**

INTRODUCTION

Teledesic is a limited license company comprised of Motorola, Boeing, Microsoft, and several other companies and venture capitalists. Their goal is to build a global, broadband Internet-in-the-Sky ™ network [1], which is targeted to begin in 2004. Using satellite technology, Teledesic is creating the world's first network to provide worldwide access to telecommunications services such as multimedia applications, computer networking, and broadband Internet access from any one point on the Earth to any other point, fully independent of any ground-based, cable infrastructure. The current constellation calls for 288 satellites orbiting in a Low Earth Orbit (LEO). The orbits are polar, meaning they travel over the poles of the Earth. The logical structure is similar to a mesh network topology.

Teledesic has two main components [2,3]: the space-based component as described above, and the ground-based component. The space-based component is the satellite constellation itself. The satellites act as routers in the network, moving the data from sender to receiver. The function of each router can be the gateway to an individual's network, or as intermediary between several routers. The ground-based component consists of transceivers either fixed upon or mobile near the surface, which serve as an interface point for messages entering and exiting the satellite constellation. Overall, this entire network is designed to support millions of simultaneous users. The Teledesic architecture must have a robust, redundant routing algorithm if it is to be successful. The algorithm must take into account several factors that traditional networks have not needed to address. The first is the extremely dynamic nature of the routers (satellites) themselves. Because of the orbit, a typical user will only be in the range of a satellite for about 20 minutes [5]. This means that the algorithm must be able to hand off to another router, possibly in the middle of a transaction. Since the orbits are polar, and the Earth is turning, this solution is not as simple as passing the transaction to the next satellite in the same plane. Cross-planar transfers may be needed. Another factor to consider is identifying which router will service which customer. For instance, with the current Internet structure, users know that the router servicing their facility will never move -- the gateway router will physically be the same. The Teledesic routing algorithm will need to identify servicing access points, in addition to the traditional role of routing the messages. While it is true that today's ground-based routers must be dynamic and robust as well, Teledesic is bringing this problem to a new dimension.

TELEDESIC ROUTING: POSSIBLE SOLUTIONS

To analyze Teledesic and pick a specific adaptive algorithm, consider the following example. If node *A* sends a message to node *B*, which happens to be exactly half-way around the world, the number of equal length paths, considering nodes and physical distance, is exactly equal to the number of planes the satellite constellation uses. To put the example differently, consider the number of equal length lines between the North and South poles -- infinite. If we limit the paths to lines of longitude only, there are 360 paths (one for each degree) from source to destination, with not one taking the same path, yet all are the same length. If one router is busy, the algorithm should take into account other equal length paths. Following this logic, it makes sense for each router to also consider its own internal characteristics to make a routing decision. For these reasons, the chosen algorithm must be a distributed, adaptive routing algorithm [4].

The Teledesic architecture requires the analysis of several characteristics that have never been a factor in conventional routing algorithms. The first is making the bridge from logical to physical addresses. In today's Internet, all devices are assigned an IP address, which is a logical number. With the proper routing configurations, two devices can be located halfway across the world from each other, yet have logical numbers that are practically neighbors. Teledesic routers, on the other hand, only have connectivity to a certain physical

location on the Earth at any one time; therefore, we propose the use of physical addresses. An address corresponding to latitude and longitude would meet this requirement. Each ground station must have one such physical address. If the fidelity of the address were carried out to the tenths of a second (degree:minute:second:tenths), the possibility of several ground stations in the same "address space" would be likely. A solution would be to assign an identifying network number to each station in that space. An advantage to this addressing scheme is that anyone could very easily determine their address without going through a controlling authority (InterNIC) to obtain their address. With GPS technologies, mobile ground stations could automatically and quickly update their address whenever needed [4].

Another factor, moving routers, adds significant difficulty to the routing problem. Since the constellation is a low earth orbit (LEO), the satellites are constantly moving in reference to a single location on the Earth. This means that the satellite used to route traffic at the beginning of a transmission may soon move out of range of either ground station. The routing algorithm chosen must handle the possibility that gateway routers will change any number of times during a communication session. Fortunately, the physics to calculate the location of celestial objects based on time is quick, easy and widely available. The challenge is to handle the changes within a communication without having to reset the session every time a satellite moves out of ground station range [4].

PROBLEM SOLUTION

Consider the case where ground station (GS) A needs to communicate with GS B. A may determine that its default gateway is Satellite 1. Using a DNS concept, the user will obtain the lat/long pair and unique number of the destination. Also using the destination lat/long pair, our algorithm will determine which satellite is the destination gateway (satellite 2). This process solves the external routing problem. The second problem is the internal routing problem, which will determine a route from router (satellite) 1 to router 2. At this point the message can be successfully delivered to the destination [4].

Ground station addresses

All ground stations will have an address based on their physical location. The address will be in the format Degrees:Minutes:Seconds. By carrying the address out to the tenths of a second, we are allowing one unique address for every 100' square location on the Earth. In addition to the geographic location, a unique number will be assigned to each ground station. This number could be adequately represented with 10 bits, ensuring a maximum of 1024 unique nodes within an area as it is only necessary to identify unique users in each "cell" which is approximately 100' X 100' [4].

Satellite Node Addresses

Each satellite will need a unique identifier. A 2-Byte identifier will provide more than enough unique addresses for the constellation, which currently calls for 288 satellites [3]. This number will be used to identify the sender's and the recipient's gateway. This number will also be used for internal routing within the satellite constellation. [4]

Domain Name Servers

Obviously, a domain naming system will need to be implemented. It would be unrealistic for a user to need to remember lat/long coordinates to use an address. Traditional, land-line, ground based servers would service user's name requests. However, in this case, a lat/long coordinate and unique identification number is returned instead of an IP address. Also, as part of the registration process for mobile users, their DNS entry would be dynamically input and updated as necessary. [4]

Function to calculate satellite positioning given time

Given the current time, a ground station knowing its lat/long coordinates will be able to calculate which satellite will be the optimal gateway to use for transmission. Also, the ground station, knowing the destination lat/long pair, will be able to calculate the destination's gateway. [4]

Determining Local Gateway

Taking the router calculation concept a step further, a static routing table can be built at each ground station, if it is non-mobile. Each entry in the routing table will consist of three numbers: start time, end time and satellite number.

For example, if the current time is 1135, the default gateway would be 61. At this point, one may ask, why do we need to number the default gateway? Why not just send the message and let any satellite in range receive and route it? The message must be addressed to a particular satellite because at any one time, two or more satellites will be within transmission range. Numbering the gateway will remove any ambiguity to which the servicing router will be. If multiple routers are used, multiple copies of the same message will be sent, wasting bandwidth and processing time and convoluting the acknowledgement process. Since the orbital characteristics of the satellite constellation will remain constant, the default gateway routing table for the ground station will remain constant. When a ground station is set up, the local routing table can be built once, and then updated only when necessary. Of course a mobile station could perform these calculations dynamically at the beginning of each transmission with minimal additional overhead.

Handoff Time

Using the proposed routing table solution, a ground station will also know the exact time the current gateway's footprint will leave the ground station's transmission range. This is critical for ensuring an efficient means exists for the destination to reply to the sender. Consider again, the situation where GS A is just sent a message to GS B. If the handoff time is within the timeout period of the current time, *B* will recalculate a new gateway for *A*. When *A* receives the reply, it will notice that it's handoff time has already expired. Therefore, before sending another message back to *B*, it will update its handoff field. Before *B* releases the message, it updates *B*'s own handoff time, using it's own static routing table. After one cycle of send and receive, both fields will be filled. [4]

Internal Routing

Internal routing is concerned with routing a message within the satellite constellation. One favorable characteristic of this satellite constellation is that in reference to itself, it is static. That is, the nodes within its network never move in relation to each other. This is the exact opposite of the case found in the external routing problem. Using this characteristic, the satellite constellation can be represented by a directed graph. The medium is a K-band transmission in outer space, and transmission time between nodes is roughly constant [5]. This leaves the graph as having a distance (or cost) of one for each link. Thus, a shortest path algorithm using node count as the metric will determine the internal route. The time consuming calculations of determining which gateways to use are left to the ground stations. Furthermore, some method of making a choice based on network health should also be implemented. A simple solution would be for a router to include in the header a number indicating it's congestion level. When the next router in the chain receives the message, it will note the congestion level and number of the sending router and update an internal table to reflect the situation. [4]

IMPLEMENTATION

This section outlines the operational details and results of implementing the pseudocode in the previous chapter into a working simulation. The main application is the simulator itself. The simulator creates an independent thread for each satellite in the system. This concept is repeated for each ground station in the simulation also. The application waits a tenth of a second between each instantiation to allow for deconfliction of system resources during startup. The rest of the simulation is simply a loop that runs for a user-defined set of time. Until time expires, a random message is generated and sent to a ground station, which calculates its gateway router using a random number. Since each thread is handling all the routing decisions and actions on their own, the only other function needed by the main application is to provide a global structure used to simulate locations of messages.

To simulate the sending of processes from satellite to satellite, or from ground station to ground station, two global linked lists are used, one for the satellite and one for the ground station. This list's relationship with the satellite threads is shown in Figure 1. Each object will be able to write to the queue of the corresponding recipient of a message. For instance, if satellite 4 receives a message to route to satellite 50, it may calculate the next satellite in the path is 5. Therefore, to simulate sending that message from 4 to 5, satellite 4 will place another node in satellite 5's queue

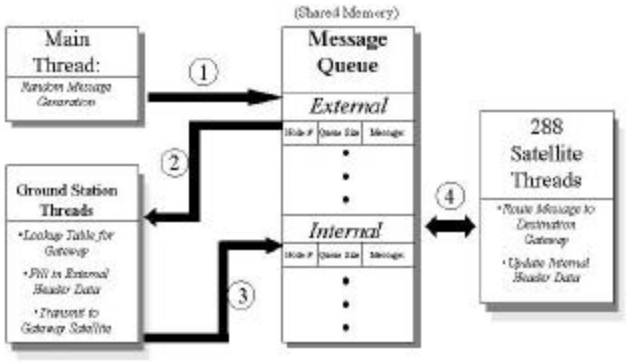


Figure 1. Simulator Architecture

and remove that same node from its own queue. A separate queue will be maintained for each ground station. When a satellite thread is created in the main program, the first thing the newly created thread does is complete a power on setup of the routing table. Initially, all paths have a cost of "one." This is due to the characteristic of the satellites not changing neighbors throughout their orbits. During the exchange of messages, the satellite is responsible for noting the congestion level in the field of the message it receives. If this level has changed, the satellite will change its internal routing table. This data structure was chosen to be implemented separately for each satellite. For this reason, each satellite may have a different routing table in a congested network. Given a unique satellite number, or node, the function GetNeighbors calculates a node's four neighbors. The calculation is trivial if the node in question is in the middle of the graph. The left, right, up and down neighbors are simply as shown in Figure 2.

However, if the selected node resides on a boundary node, then the calculation changes. The neighbors for a bordering side node are shown in Figure 3. Note that in this case, the neighbor isn't simply the first or last node in the same row of the graph. The neighbors are chosen in the manner shown because of the characteristics of "unwrapping" the satellite constellation from a three-dimensional circular object representing the Earth. The neighbors for a bordering node on the top or bottom row of the graph are shown in Figure 4.

Since Teledesic calls for 288 satellites, we implemented a 12 X 24 node graph in our simulator.

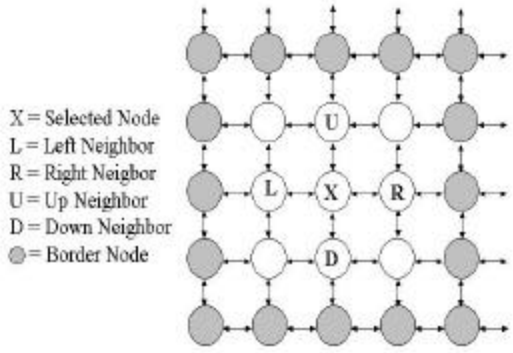


Figure 2. Trivial Neighbor Solution

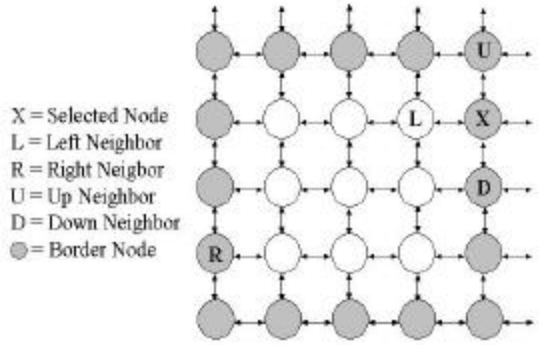


Figure 3. Side Border Neighbor Solution

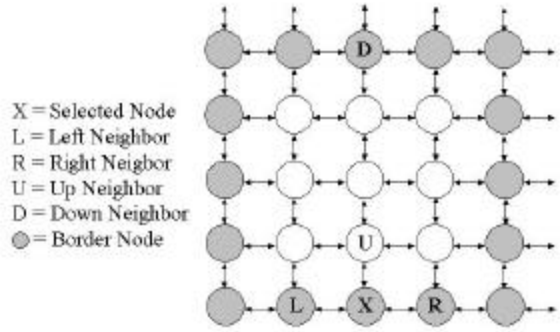


Figure 4. Top/Bottom Border Neighbor Solution

Each satellite node, after startup, runs in a loop constantly searching its row in the linked list for messages. If its "numMessages" field is greater than zero, then the node removes the first message from the queue and recalculates a destination router. If the message structure contained a changed congestion level, then the node will update its own routing table at that point.

IMPLEMENTATION RESULTS

Test cases of the simulation implementation have been successful in proving that the algorithm proposed in this study will route a message from ground station to ground station via a network of mobile routing satellites. This section will describe the data collected in addition to an analysis of the results. A total of 126 test cases were created and run, resulting in over 40,600 messages simulated being delivered. In all cases, 10 ground stations were simulated, along with 288 satellite nodes. The following data was collected during each test case:

- *Message Density:* The speed at which the simulation generated messages for delivery. The number roughly translates to the number of milliseconds to wait in between sending the next message. Therefore, a higher number results in a slower delivery rate.
- *Ground Sleep:* The amount of time in milliseconds that a ground station and satellite router will pause if it determines that it's queue is empty before searching it again.
- *Congestion Routing (on or off):* Indicates whether congestion routing was turned on or not during that particular test run.
- *Total Messages Delivered:* The total number of messages successfully delivered during the test case.
- *Total Messages Created:* The total number of messages generated by the simulation of that particular test case.
- *Average Delivery Time:* The average time in seconds that each message needed from creation at the source ground node until delivery at the destination ground node.
- *Average Hops:* The average number of routing decisions needed for each delivered message during the simulation.
- *Number of Handoffs:* The number of times a destination gateway satellite moved out of range during the execution of the simulation.

From the collected data, several data items could then be calculated for each test case:

- *Percent Delivered:* [Number of Messages Delivered] divided over [Number of Messages Created]
- *Time per Hop:* [Average Delivery Time] divided over [Average Hops per Message]
- *Handoffs per Message:* [Number of Handoffs] divided over [Number of Messages Delivered]

We present three testing phases in the remainder of this section.

Phase I: Determine a Baseline. The first phase of testing was designed to find an ideal set of parameters for [Ground Sleep] and [Message Density]. The goal was to find a pair that would generate a large number of messages, without tying up the computer's resources to the point of causing many missed messages. With a broad range of values for ground sleep, the experiments showed that by setting message density to 1000 (1 new message created approximately every 1 second) provided the most stable testing environment, which averaged just over a 91% delivery rate. For the purposes of future test cases, a value of 200 was chosen for a [Ground Sleep]. By setting [Ground Sleep] to 200 and [Message Density] to

1000, tests resulted in an overall average successful delivery rate of 97%.

Phase II: Evaluation of Success of the Algorithm. The second phase of the testing process was to generate and analyze a group of test cases and evaluate whether or not the algorithm in this thesis performed in a manner expected based on various simulation parameters. Figure 5 shows a summary of the results of 58 separate test cases, each simulating the delivery of a random number of messages

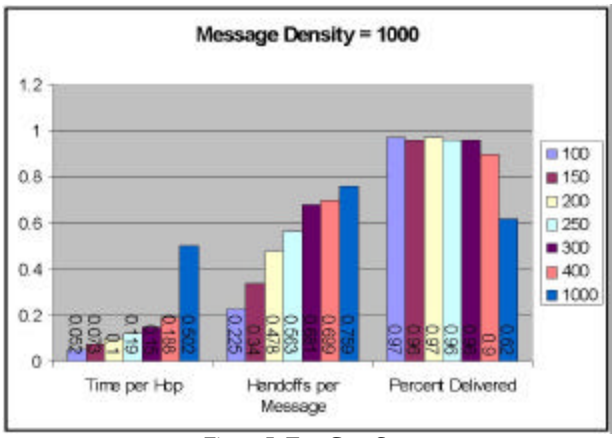


Figure 5. Test Case Summary

(minimum: 100, maximum: 1500). In each test, the simulation was set with the [Message Density] parameter to 1000, simulating approximately 1 message every second. Each message was randomly set to consist of between 1 and 10 packets. The legend on the right shows the [Ground Sleep] values. Three main characteristics of each test case are shown for each varying value of [Ground Sleep].

As the [Ground Sleep] time is lengthened, we would expect the characteristics listed below:

- *Time per Hop:* increases by a constant rate (with the exception of the last case). We attribute this anomaly due to errors caused by resource constraints in the testing machine. By setting the [Ground Sleep] to a high value, router queues fill up quickly, exhausting the memory allocated to each thread, causing it to fail and lose its currently stored messages. To avoid this, we suggest that a machine with a large amount of memory (at least 512 MB) is chosen. If this solution is not feasible, then a choice for [Ground Sleep] must be chosen so as to not allow the queues to grow too large. For the computer in our simulation, setting [Ground Sleep] equal to 1000 was too large, but values between 100 and 400 allowed the simulation to work correctly.
- *Handoffs per Message:* Increases by a constant rate, which is consistent with a longer [ground sleep] time.
- *Percent Delivered:* Fairly constant, with the exception of having a ground sleep of 1000, which simulates the routing thread blocking for 1 second if the queue is empty.

This data supports a stable, consistent and correct simulation of the proposed satellite routing model. A second supporting test is shown in Figure 6, which also demonstrated a consistent operating simulation.

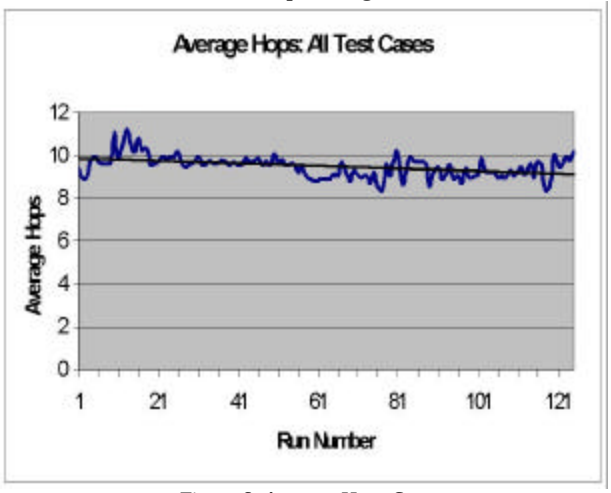


Figure 6. Average Hops Summary

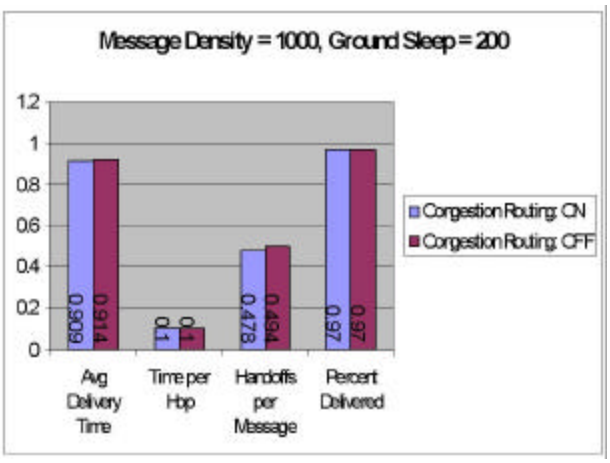
Since the satellite constellation is a 12 by 24 node directed graph, the maximum number of hops needed to transit the internal network can be calculated by adding together half of the maximum x value and the maximum y values: .5(12) + .5(24) = 6+12 = 18. With a large number of test cases, one could expect to see an average of 9 hops per test case. Also, the average number of hops should be relatively constant. In fact, an analysis of the data shows this is true, with an approximate value of just under 10 for average number of hops. This observation is shown in Figure 6, with the single straight horizontal line representing the average trend.

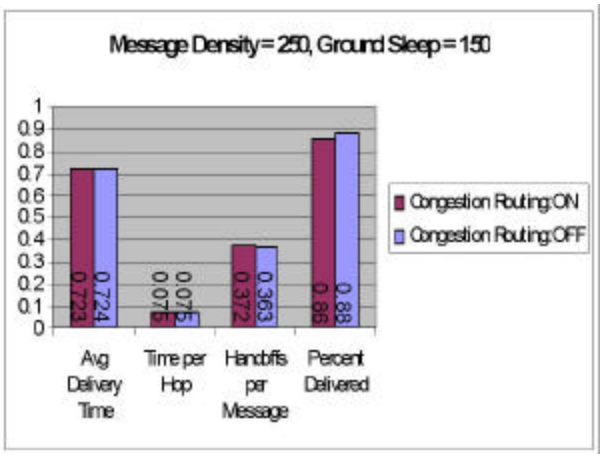
Phase III: Congestion Routing. The third and last phase of the data analysis evaluated the congestion routing portion of the algorithm. Several tests were run with congestion routing on and off, with all other parameters of the simulation being the same. Two sets of test cases were run:

- Set 1: [Message Density]=1000 and [Ground Sleep]=200
- Set 2: [Message Density]=250 and [Ground Sleep]=150

In both cases, all of the results are almost identical for all characteristics collected or calculated for the simulation. This is probably due to the complexities that arose in trying to start the simulation with heavily preloaded router queues. The system used to run the simulation would fault whenever a substantial load was applied prior to starting the simulation. However, it is important to note, that with or without congestion routing turned on, the simulation successfully routed messages to their intended recipient. This is an important conclusion since one should reasonably be concerned if using congestion routing on a low-load network

would inappropriately degrade performance due to the extra





calculations involved. In this implementation, the answer is that turning congestion routing on has no negative impact on performance even in a low-load system. Figure 7 shows the results of executing both sets of congestion routing test cases.

CONCLUSION

The complexities that arise from the Teledesic concept of networking have spawned a unique and complex problem space to routing high-speed multimedia network operations. To solve this problem, a routing algorithm that uses physical addresses instead of logical addresses can be used, which is a divergence from normal routing operations today. Our tested algorithm breaks away from the traditional logical addressing concept and solves the physical routing problem. Our ideas were tested successfully via a software-based simulator, which indicated high-speed multimedia application data can be efficiently and reliably be routed using our algorithm.

REFERENCES

[1] Gilder, George, "Ethersphere," *Simon & Schuster*, 1996.

[2] Khare, Rohit, *Reflections on the Teledesic Security Architecture*, CalTech University, 28 March 97.

[3] Motta, Mary, *Teledesic to Roll Out Plan for Battered ICO*, www.space.com, 7 Oct 2000.

[4] Schneller, Guizani and Murray, *A Routing Solution for a Global, Space-based Multimedia System*, IEEE GLOBECOM 2001, 3 Nov 2001.

[5] *Teledesic Technology Overview*, www.teledesic.com, 2000, Teledesic, LLC.