Data Centric, Position-Based Routing In Space Networks

Omprakash Gnawali[‡], Mike Polyakov[†], Prasanta Bose[†], and Ramesh Govindan[‡]
[‡] Department of Computer Science, University of Southern California
941 W. 37th Place, Los Angeles, CA 90089
gnawali@usc.edu and ramesh@usc.edu
[†] Lockheed Martin Corporation
1111 Lockheed Martin Way, Sunnyvale, CA 94089
michael.v.polyakov@lmco.com and prasanta.bose@lmco.com

Abstract—Envisioned space exploration systems and planned space science missions involve increasingly large number of satellites and surface rovers/sensors communicating for coordinated science operations or for on-demand commanding and/or transfer of data. Current approaches that use static routing cannot scale to large numbers of satellites and spacecrafts of future missions. This requires a dynamic approach that can discover networks and links as they become available and intelligently use them for routing. Furthermore, most of the science missions will be geared towards collecting data using various sensors. Adoption of a data-centric communication mechanism can enable in-network aggregation and processing which help make data forwarding more efficient. In this paper, we briefly describe ASCoT, a routing system for science missions of tomorrow, which a) leverages the predictability of satellite trajectories to effect position-based routing in the space backbone, and b) departs from traditional address-centric communication and uses a data-centric architecture to enable energy efficient and low latency operation in proximity networks. Our simulation study using STK/OPNET shows that ASCoT architecture is viable.

TABLE OF CONTENTS

1. INTRODUCTION

In the space exploration and science missions of the future, a large number of spacecrafts, rovers and sensor nodes on planetary surfaces will need to interact seamlessly for on-demand commanding, coordination, and transfer of data. Current approaches based on planned communication routes among space assets are inadequate to meet the on-demand and resource optimizing communication constraints, especially with such large number of nodes.

Static routing has been the method of choice for most routing in space networks. Network engineers manually program the routing tables into the nodes. When a node receives a packet, it looks up its routing table and decides the next hop for that message. This technique works well if the nodes and links as well as traffic characteristics are fixed and well-known ahead of time. Static routing also works well for inter-node communication in a formation flying system. But if new satellites become available, routing tables on nodes that need to communicate with that satellite need to be changed. Newly deployed nodes might also provide an alternate path for communication and better communication services such as higher bandwidth. It is possible to manually update routing tables in order to accommodate new nodes and capabilities, of course. However, when the number of such nodes increases, it becomes increasingly vital to have a system that can adapt itself to network dynamics without requiring a manual intervention. ASCoT can adapt itself to use new resources available in a space network and keep on functioning gracefully in a dynamic space environment.

In this paper, we briefly describe ASCoT, a dynamic routing system for science missions of tomorrow, which a) leverages the predictability of satellite trajectories to effect position-based routing in the space backbone, and b) departs from traditional address-centric communication and uses a data-centric architecture to enable energy efficient and low latency operation in proximity networks.

On satellites providing backbone (interplanetary) communication links, ASCoT leverages predictable positioning and dynamics of nodes in space to make routing decisions. Positional Link State Routing (PLS), an extension to the link state routing used in the Internet, enables ASCoT to compute paths using links that will be available even in the future. PLS is unique relative to other link state protocols in two ways. Traditional link state protocols do not use positional information, while PLS does. Furthermore, in PLS, link state information can be proactively disseminated.

ASCoT also uses data-centric communication for proximity routing such as satellites in a close formation or rovers and

0-7803-8870-4/05/$20.00/©2005 IEEE
IEEEAC paper # 1219, Version 3.14, Updated December 10, 2004

sensors on a planetary surface. Such a communication mechanism allows declarative access to sensor data from proximity networks. For example, a PI conducting an experiment is able to issue a query of the form: ”Return an image from this region of Mars when the average temperature is greater than 25 degrees, and the light levels are below 10”. Data-centric communication achieves energy-efficiency in two ways [16]: a) Naming the data allows the system to eliminate different levels of binding and b) Naming the data allows in-network processing of data. Intermediate nodes now have the context to transform the data in several interesting ways: for example to aggregate different data items that perhaps have redundant information (e.g. temperature data from nearby sensors), or reduce information in response to resource constraints (e.g. downsample an image because the image size exceeds available network capacity).

We have implemented both of these mechanisms in the OPNET [3] simulator and integrated our simulator with an STK [4] front end. We have also implemented a query front end which supports multiple, simultaneous one-shot and streaming queries. This provides concurrent and constant access to space assets to the scientists and the mission controllers. An evaluation of PLS using our simulator shows that even with a limited knowledge of future link schedules, PLS can compute the best paths that maximize a given metric.

2. Related Work

Recently there has been an increasing interest in communication infrastructure for the space Internet. We are not aware of any work that addresses route dissemination, path computation, and integrating data centric architecture for the science missions. Following are the major categories of work related to our effort.

Routing in Space

A number of studies [9], [7], [12], [14] propose exploiting the geometry of orbits to compute the next hop towards the destination. These schemes assume that the information relevant to path computation (topologies and orbits and number of satellites) are well-known and programmed into the satellites ahead of time. Given this information, custom algorithms that exploit geometry can be used to compute the paths to the destination at any given time. These studies are limited to various Earth orbits (LEO, MEO, GEO).

MARVIN [29] is a routing protocol for the interplanetary network and leverages predictability of the orbits to compute paths. MARVIN uses an augmented Dijkstra’s algorithm to compute the shortest path according to a given metric. However, they use published ephemeris tables and only address path computation and the algorithm can not rapidly adapt to expected as well as unexpected changes. They propose computing the paths in a centralized node and disseminating the routing table to the nodes. We believe this will not scale to the large number of satellite nodes we envision in the future.

TCP/IP and UDP in Space Networks

There is an overwhelming desire to make any system and protocol co-exist with the IP [24] stack to maintain backward compatibility in the existing infrastructure. However, the case for using TCP [25], the de-facto transport protocol in the Internet, in the space network is not strong. TCP uses end-to-end control mechanisms to probe and adjust to network dynamics. The performance is drastically reduced when the latencies are high, because end-to-end control mechanisms tend to misinterpret latency as a sign of congestion. A number of studies [11], [19], [6] review the impact of high bandwidth-delay product on TCP and conclude that TCP is ill-suited for space like environments where delay will be minutes if not hours. There has been a number of proposals on how to modify TCP to adapt it to the space environment [10]. UDP [23] which does not use end-to-end control might be more applicable in space if one insists on using the conventional networking stack. UDP does hop-by-hop forwarding with no end-to-end guarantees of reliability. ASCoT messages resemble UDP in the sense that hop-by-hop interaction is used to affect end-to-end QoS requirements.

Delay Tolerant Networking

Delay Tolerant Networks (DTN) are characterized by high latency, low data rate, frequent link disconnections, and lack of end-to-end path at a given time [13]. The space internet can be thought of as an instantiation of DTN. In the DTN architecture, nodes that are within the same *region* use the native protocol optimized for that environment. Nodes in different regions communicate to each other using the Bundling protocol [28] through DTN gateways. DTN uses a store and forward paradigm in which bundles might be sometimes stored for a long period of time and forwarded to the next hop towards the destination when the link becomes available. A recent work establishes a taxonomy of different routing protocols that one might use in the context of DTN [17]. Our study could be viewed as fleshing out some aspects of the DTN architecture: the actual routing protocol used in the space backbone, and the networking stack used in proximity networks.

Terrestrial Sensor Networking

Much effort in sensor networking is directed towards collecting environmental data using nodes with low computation, communication, and energy resources. The wireless links often are of poor quality and intermittent. Data centric architectures (Directed Diffusion [16], One Phase Pull [15], GHT [26]) have been proposed and studied extensively in these networks. Directed Diffusion uses data centric naming that eliminates different levels of binding and enables in-network processing of data. These techniques can be adapted to science space missions which are also characterized by nodes with low computation, communication, and energy resources. We propose using protocols such as One Phase Pull (a routing protocol under the Diffusion framework) for routing in the proximity network consisting of a constellation of spacecraft or a group of rovers on the surface of a planet.

For commanding and querying the vehicles, we propose using data centric rather than node centric naming and communication.

While ASCoT borrows the idea of data-centric routing from sensor networks, ASCoT's data-centric routing schemes are likely to be significantly different from those used in sensor networks today. They will be more tightly integrated with position-based routing, will explicitly deal with the significant amount of heterogeneity that exists in space networks (in terms of link bandwidth and node processing capability), and will attempt to leverage some of the predictability of space configurations (e.g. known satellite trajectories) to intelligently route information.

3. Why not Internet Routing?

In this section, we first list the unique challenges in space communication before describing why the standard routing algorithm used in the Internet is not directly applicable in the context of space networks.

Link Characteristics in Space Networks

High latency— EM radiation travels in space at the speed of light. This results in communication between Earth and satellites in LEO/MEO taking a few hundred milliseconds, between Earth and Mars taking several minutes, and between Earth and deep space probes taking a few hours to a few days.

Low throughput— While terrestrial networks boast rates exceeding GBps, satellites communication bandwidth is rarely over a few MBps. Usually the farther away the satellite, the smaller the bandwidth available due to errors that can be attributed to lower signal to noise ratio, effectively ruling out expensive routing algorithms with too much overhead.

Short link duration— Satellite nodes are expected to be visible for a few hours or less depending on the orbit before they get occluded by planets. This is in contrast to wired networks in which nodes are reachable for a few weeks without any interruption. One can use redundancy to effectively increase the duration of link. DSN [2] base stations, for example, use this technique to increase visibility between Earth and distant objects. More generally, however, for deep space communication, limited but largely predictable link durations will be the norm rather than the exception.

Link asymmetry— Data rate usually is not the same on both directions of a link. Satellites are usually provisioned for significantly higher data rates downstream. Future tasking nodes might have opposite characteristics. Currently, most of the traffic between Mars/Earth is earthbound with ratio as high as 1000 to 1 [11]. Routing protocols that use a forward direction route update to establish reverse routes without taking data rates into account are bound to perform badly.

Link dynamics and outages— Wireless links in space come with the benefit of some predictability but also demonstrate significant unpredictability. Satellites and planets follow orbits that can be computed and predicted to a reasonable accuracy. This makes certain links periodic. The Mars rover, for example, communicates with Earth only when it is facing the Earth. The connectivity follows the cycle of day and night on Mars. Many satellites in deep space missions are frequently occluded by planets, moons, and asteroids but these occlusions can be computed ahead of time. This allows us to predict the availability of links to these satellites. However, there can be unexpected link outages caused by terrestrial elements [1] or solar flares [5]. These unexpected outages can last several hours to several days.

Point-to-point communication— Long distance communication is almost always point-to-point; omni-directional antenna is rarely used. Due to the use of high-gain antennas for regular communication, the reception drops off rapidly as one moves from the cone of signal propagation. Satellites steer (passively or actively, mechanically or electronically) their antenna to point to the node for communication.

High Error rates— Interference in space and in the atmosphere will significantly degrade the reliability of links over terrestrial counterparts even with error correction. By trading data rate for error correction, space links can achieve BER rates of 10^{-9} to 10^{-7} compared to fiber optic error rates of $< 10^{-12}$. To-earth frame error rates of 10^{-4} to 10^{-6} can be expected, and weather conditions cause 5% of frames transmitted to be lost. Standard assumptions that packet loss is caused entirely by buffer overruns no longer hold.

Routing in the Internet and MANETs

Routing mechanisms used in the Internet are well understood and highly scalable. BGP [27] is the de-facto standard in inter-domain routing in the Internet. OSPF [21] is used for intra-domain routing. A combination of these two mechanisms have enabled highly reliable and scalable routing in the Internet.

Mobile ad-hoc wireless networks work reasonably well under a high degree of dynamics. Recent widespread adoption of 802.11 wireless networks illustrates that the mechanisms for wireless network operation are stable and reliable to the point that it is starting to challenge the wired infrastructure. In this environment, nodes are constantly moving from one place to another but they are still able to probe for available services wherever they happen to be, associate with a node that is able to provide the best service, and use that node as long as it is available. The capability to detect new services as well as failed services is critical to being able to adapt to the changing environment.

The wireless ad hoc solution for campus-wide scale is not applicable in the space. The wireless part of these networks is usually a single hop network while ASCoT needs to use multiple hops to reach a destination. While there are MANET [8] standards for multiple-hop wireless networks, they are rarely used in practice and have hardly been tested in real world application. Multiple hop wireless networking is a hot topic and several protocols for route discovery have been proposed. AODV [22] and DSR [18] are probably the most well studied proposals. A recent study on an ad hoc vehicular network [20] shows that these protocols perform poorly under high mobility. These protocols are a class of protocols called reactive protocols. They probe the network to find paths to different destinations when asked to route packets. However, due to high mobility, the paths become outdated quickly.

MANET routing protocols (AODV and DSR) assume a broadcast media. In AODV, for example, routes for data forwarding are formed along the propagation path of the query (Route Request messages). This algorithm finds reasonable paths only if the links are symmetric. Both MANET routing protocols and BGP/OSPF do not have mechanism to use periodicity of the links to compute paths. They use only the active links to compute a path. In space networks, often all the links on a path are not active at a given time and messages need to be buffered for long periods of time, in the order of minutes and hours, as opposed to buffering in the Internet in the order of milliseconds. Thus, even though solutions deployed in MANET's and the Internet are known to be highly stable and scalable, they can not be directly used in the context of space networks.

Host-centric Architecture

The Internet uses a host-centric architecture to relay messages from a source to the destination. The source and destination nodes are identified using unique names and addresses. The intermediate nodes know how to relay messages towards a given address. When a source intends to send a message to a destination, it sends a request to a server (commonly a DNS server) to provide the IP address for the given name. This process is called name resolution. Once the name is resolved, i.e., the IP address is known, the source sends the message to the destination IP address. While resolving names using a DNS server might not be a problem for Earth to Space communication, it will be costly in energy as well as latency for Earth-bound messages that originate on Mars assuming such DNS servers will be deployed on Earth. For a rover on Mars to resolve names using a DNS server can take minutes of communication overhead. An alternative is to use explicit IP addresses and use hard-coded sources and destination identifiers in all the layers of the protocol stack. However, this results in a highly inflexible configuration. Furthermore, when a PI asks a query such as "What is the average temperature of the shadowed parts in Gusev crater?", a host-centric approach first collects the IP addresses of all the sensors and rovers in the crater (a process similar to name resolution) then sends query to each of the node in the crater. This multiple lookup and binding seems highly inefficient.

It has been shown that when network communication is data-centric (e.g. when a message specifies its content in terms of attributes; thus a message from a node in a proximity network might contain attributes such as "location", "temperature"), energy-efficiency is obtained in two ways: a) Naming the data allows the system to eliminate different levels of binding and b) Naming the data allows in-network processing of data. The query for the average temperature of the shadowed parts in Gusev crater, for example, now can be flooded in the proximity network and only the nodes that meet the query criteria (in Gusev crater and in shadowed parts) will respond to the query thereby avoiding multiple steps of binding and spending energy transmitting data from nodes that are not in the shadowed parts of the crater. Intermediate nodes also have the context to transform the data in several interesting ways: for example to aggregate different data items that perhaps have redundant information (e.g. temperature data from nearby sensors), or reduce information in response to resource constraints (e.g. downsample an image because the image size exceeds available network capacity). In this way, space systems will be able to reduce the latency of communication compared to a purely IP based approach. Furthermore, using these ideas, space networks can perform hop-by-hop error recovery because the data is self-identifying.

In this section, we surveyed different networking technologies that have been deployed in the Internet and explained how they can not be directly applied in the space internet. There is an overwhelming desire to make any new deployments compatible with the current infrastructure and host-centric architecture. However, we believe that the science missions can benefit greatly from data-centric approach advocated by the ASCoT architecture.

4. ASCoT ARCHITECTURE

ASCoT is a routing and scheduling substrate for flexible tasking and coordination among space assets. Unlike existing solutions for terrestrial routing, ASCoT is specifically designed to surmount challenges unique to the space domain and offers an asynchronous API for flexible tasking and coordination that can adapt to the dynamics of the environment. More specifically, it addresses the following requirements:

- Scaling to several hundred nodes including orbiting constellations, rovers, static sensor pods etc.
- The ability to deal with message propagation latencies in the order of tens of minutes to hours.
- Connectivity changes due to physical events, such as planetary occlusions or solar flares.
- Heterogeneous and asymmetric link bandwidths, as well as widely varying link error characteristics, resulting in the need for Quality of Service (QoS).
- The ability to name data in order to make it self-identifying.

ASCoT expects the underlying system to provide a variety of information and services to the ASCoT middleware. This includes:

• Navigation information – characteristics of the links available and nodes on the other end, including position (current and expected), bandwidth, reliability, latency, etc.
• Current position of the node
• Local status (power, health, load of transmission queues, etc.)
• Ability to send and receive messages, with the underlying layers handling all framing and encoding procedures.

Applications interface with the ASCoT middleware using the Diffusion API: *interest* and *data* messages, filters, and their corresponding callback functions. When a PI issues a query to the system, the query is translated into an interest message and handed to the ASCoT middleware. Whenever there is data available that matches the query, data is forwarded to the callback function. The applications on rovers and sensor nodes indicate the types of queries they are interested in by installing filters with a set of attributes. Arrival of matching queries is announced to the application using a callback for the filter. Without filters, the default behavior is to receive all interest messages. A node can then decide queries to which it will and can respond. Sensor nodes and rovers respond to queries using data messages. The ASCoT layer ensures that there exists an interest for that type of data before forwarding it towards the querying node.

ASCoT uses a protocol called Positional Link-trajectory State (PLS) routing to compute paths to the destination nodes. PLS is an adaptation of Link State Routing (LSR) to the context of space networks. The basic idea behind our proposed approach is that link trajectories, together with link attributes such as latency, data rate and error characteristics are disseminated throughout the network. Using this information, each node can then independently compute a path to a given location by computing a shortest path tree (using a modified Dijkstra's algorithm) spanning the network. The next hop for a message is then determined by looking up the next edge leading to the destination in this spanning tree. As long as change in neighborhood information is infrequent, and the total number of nodes (and links) participating in link state routing is small, the protocol is known to work efficiently. The path computation can also take into account link characteristics in order to satisfy any Quality of Service requirements that a particular message (or conversation) might have.

PLS departs from traditional link-state routing in one very important way. In PLS, the information disseminated throughout the network is the trajectory of nodes in space, and the availability, of the link endpoints. We intend PLS to be run only on mobile space assets. A mobile space asset is a satellite or a base station on planets that are moving large distances with respect to potential communicating partners. We have a qualitative definition of "large distance". We consider base stations on Earth, and the satellites around Earth and Mars as mobile space assets. By our definition, then, a Mars rover would prefer to relay all its messages through a powerful base station on Mars, so a Mars rover would not participate in PLS routing. On the surface of Mars, sensor and rover networks will use Diffusion [16] based data-centric routing. Diffusion provides naming and aggregation framework for data generating ad hoc networks such as sensor networks. Diffusion can use different routing protocols such as One Phase Pull, Two Phase Pull, and Push. Combination of naming, framework for aggregation, and data-centric routing protocol makes this light-weight substrate well-suited for proximity networks. On Earth, one would just use IP based routing to exchanges messages with the base stations such as DSN base stations.

To adapt LSR to the unique demands and constraints of the space internet, we propose that nodes participating in PLS routing exchange the following set of information about each link that they can form with their neighbors: $\{u, p(u), v, t, attributes(u \rightarrow v)\}$. Here u is the sender, $p(u)$ is some description of its position, v a node with which it can communicate, t, expressed as a list of {start,stop} tuples is a set of time intervals (also mentioned as contacts in [17]) in which the link $(u \rightarrow v)$ is available. $Attributes(u \rightarrow v)$ are a set of metrics that capture link quality information, such as bandwidth, latency, error rate etc. This information is flooded reliably throughout the network, and is used by nodes to compute paths. The nodes exchange this information every few hours to a few days. The frequency of update must be kept high enough so that link updates are communicated in due time but low enough to ensure conservation of networking and energy resources and to prevent accumulations of large numbers of these updates. Once link information about neighbors is available, the nodes compute the "best metric" trees for each QoS metric.. Examples include computing the least latency or most bandwidth paths two nodes. The algorithm uses the time intervals of link availability to predict and ensure links are available when a packet reaches those links on its way to its destination.

How does PLS address the constraints of space enumerated in Section 3? While space links have high latency and space assets are continuously mobile, PLS takes advantage of their relative predictability by distributing information about link availability throughout the network ahead of time. Thus, even with long delays, the information about link availability is known to PLS nodes when needed (PLS deals with unexpected link failures as well, in a manner described in the next section). Furthermore, information about link data rate asymmetry and link quality are distributed in PLS, so nodes can make intelligent routing decisions.

Now we describe the three key components of ASCoT routing: (1) dissemination of link information, (2) computing the paths (routing tables) on interplanetary links and (3) forwarding messages on the interplanetary links using the routing and using gradients in the proximity networks. The base station bridges the interplanetary network with the proximity networks using message translation.

```
v.receive_linkstate (links)
  for all l in links
     if exists(known_links, l.(u->v))
         known_links[u->v] = l
     else
         known_links.add(l)

v.link_refresh()
  for all l in known_links
    if (l.age > LINK_TIMEOUT)
       l.delete()
    l.age++
  send(known_links)
```

Table 1. PLS link information dissemination algorithm

Link information dissemination

PLS leverages the largely predictable trajectories of space assets and distributes the link information before the links become available (this is not the case with mobile ad hoc networks). When a link comes up, nodes at each end of the link exchange their link state information; in this manner, whenever nodes could have used a link for communication, they have the necessary information to compute paths. Initial knowledge of the link's existence is obtained either from a lower stack layer or through previously known links during link state dissemination. Over a period of time, nodes accumulate enough link state information to compute the shortest paths to each node in the network. Table 4 shows a simplified pseudocode that describes the dissemination algorithm.

PLS requires bi-directional links but asymmetric up/down bandwidths do not pose a problem because bandwidth in each direction is accounted for in the link state information and taken into consideration while computing the paths.

Path computation

Traditionally link state protocols use Dijkstra's shortest path algorithm to compute a path to the destination. The algorithm expects a set of links, link weights, and the root node as inputs. To compute the shortest path from the root node, all the links are assigned a weight of 1. Thus, the algorithm computes the shortest path (in terms of number of hops) to all the destinations in the network. This algorithm assumes that all the links are valid and active at all times. However, in space networks many links are intermittent or periodic. In order to use Dijkstra's algorithm in this environment, we need to use predicted link connectivity (part of link state entry for a link) and make sure the link is valid when the packet arrives at the link. In other words, we need to make the algorithm take into account future topology schedules. Furthermore, observe that different link attributes disseminated with PLS can be used to compute paths with different QoS.

To achieve these goals, we need to make the following main changes to Dijkstra's algorithm. First, instead of hop-count, minimize the desired QoS metric. Second, modify the algorithm to account for time-varying graphs. The property that our algorithm satisfies is that a path between two nodes never uses, at any time t, a link that is not active at t. To ensure this property, we modified Dijkstra's algorithm so that it does not add a link l to the shortest path tree if l is not valid between times $t1$ and $t2$ where $t1$ is the time at which a packet would arrive at the leaf of the tree *before* adding a new link and $t2$ is the time at which a packet would arrive at the end of the tree *after* adding a new link. Table 4 shows the pseudocode for the augmented Dijkstra's algorithm. This pseudocode uses *sum* as the aggregation operator and latency as the link metric. One can think of using different aggregation operators and metrics with the same algorithm. For example, *product* aggregation operator for link reliability, *sum* for energy metric, *min* for maximum bandwidth, etc. Path computation builds a routing table that lists each node that participated in PLS message exchange and the next hop on a path that maximizes a given metric. Table 4 shows a sample routing table.

At times, we would like packets to wait if better links are going to be available in the future. This reminds us that routing decision is not simply building a routing tree; we need to employ intelligent scheduling to meet the application's QoS requirement. At this stage, we remark that scheduling might effect routing decisions, but we leave the design of the scheduler as future work.

Computational complexity is of concern in a routing protocol because forwarding must happen using outdated paths while new paths are being computed. CPU cycles are expensive because of energy scarcity in space missions and the variety of maintenance, bookkeeping, and science tasks the CPU must perform. Memory is equally precious. Of late, planetary rovers, for instance, boast a few hundred megabytes of memory. There is competition for memory usage from science data, system data, communication buffers, and diagnostic data. Asymptotic computational complexity of Dijkstra's algorithm is polynomial in the product of number of nodes and edges in the network. The proposed modification to Dijkstra's algorithm does not change its asymptotic complexity. In the $relax$ procedure, for a given link, using $link\_available$ call, we iterate though all the available intervals. This adds a multiplicative factor to the asymptotic complexity of the overall algorithm. But we expect a small number of available intervals for a given link. Further, this list can be sorted to minimize search time. If the number of nodes is large, it might not be feasible to store the entire tree in the memory in addition to the link state information. This is made all the more difficult since the tree varies with the send time of the packet. An alternative would be to retain link state information but build the routing tree on-demand. This approach is not suitable when there are frequent forwarding requests. In a system that gets sparse forwarding requests, this might save memory for other important tasks.

Message Forwarding using PLS

As mentioned in the previous section, the interplay between routing and scheduling could become quite complex. The

```
// returns true if link l is available at time t
link_available(t, l)
  for all i in known_links[l].intervals
    if (t >= i.start) and (t < i.stop)
      return true
  return false

// compute the latency metric for a given link
latency_weight(u, v, arrival_time)
  for all i in known_links[u->v].intervals
    if i.start <= t && i.stop >= arrival_time
      return link_latency(u,v,now());
    if i.start >= t
      return link_latency(u, v, i.start) +
                 (i.start - arrival_time);

// updates the tree if d[u] + (u,v) is shorter
// (according to a given metric) than d[v]
relax(u, v, metric_weight, agg_op)
  newdv = agg_op(d[u], metric_weight)
  if (d[v] > newdv) and
      link_available(arrival_time[u], (u,v))
    d[v] = newdv
    pi[v] = u
    arrival_time[v] = arrival_time[u]
          + latency_weight(u, v, arrival_time[u])

// initialize Dijkstra variables
initialize(G, s)
  for each vertex v in V[G]
     d[v] = inf
     pi[v] = nil
     arrival_time[v] = nil
  d[s] = 0
  arrival_time[s] = 0

// compute the best metric tree
// from source s in graph G
dijkstra(G, w, s)
  initialize(G, s)
  s = {}
  q = V[G]
  while (q != {})
    u = extract_min(q)
    S = S union u
    for each vertex v in adj[u]
      relax(u, v,
        latency_weight(u,v,arrival_time[u]), sum)
```

Table 2. Algorithm to compute PLS paths.

Metric: Latency			
Destination	Next hop	Trajectory	Path Metric
Mars_base1	Earth_relay2	t1	1400 ms
Mars_base3	Earth_relay2	t1	1402 ms
Metric: Bandwidth			
Destination	Next hop	Trajectory	Path Metric
Mars_base1	Earth_relay4	t3	600 Kbps
Mars_relay2	Earth_relay3	t4	400 Kbps

Table 3. Example routing table for a DSN node on Earth

```
find_next_hop(destination, metric)
  entries = get_routing_entries(metric)
  return closest_entry(entries, destination)
```

Table 4. Algorithm to find the next hop towards the destination

current version of the ASCoT router instead performs a simple task: once routing table is populated for a given metric, lookup the best next hop towards the destination and buffer the packet until the link becomes available.

Most ASCoT queries have a geographic location or a region as the target. When a message is sent from a workstation on Earth to a region on Mars, the message is routed to the base station on Mars using forwarding tables constructed by PLS. The algorithm (Table 4) looks for a node that is geographically closest to the destination location.

Note that this is a greedy forwarding strategy with an assumption that a node closest to the destination location will know how to forward the message to the nodes in the destination. The link updates close to Mars might not have already propagated to nodes closer to Earth. In this case, the nodes closer to Mars will have a more up-to-date information on which links and proximity network can be used to forward messages to the nodes in Gusev crater.

Not all the nodes in the network run PLS. Then, how can messages be forwarded to these sensors and rovers? The sensors and rovers on the surface will be a part of a proximity network (such as an 802.11 subnet); the node closest to destination might have the means to leverage the proximity network to reach the destination. For example, the closest PLS node to Gusev crater might be a Mars base station with two sets of communication hardware to function as message gateway. Thus, the router will forward the message to the *base station* closest to the intended destination; the base station then will use proximity network to reach the rovers and sensors.

Message switching

The base station in a proximity network acts as a message gateway. It decodes the data-centric name for the target, encodes it as an attribute along with other constraints for the query, and uses Diffusion to harvest data. The node, using Diffusion semantics, translates a query into an interest message, floods the network and sets up gradients in the network using the One Phase Pull [15] protocol. Sources, when they generate data, inject messages into the network as a set of attribute-value tuples. Gradients guide data to the base station by matching attributes in the data message to that of the gradients established by the interest messages. Data not matching the gradients are dropped at the first hop. Data filtering and aggregation happen in the network, close to the site where data is generated, rather than transporting them to Earth only to find out most of the data needs to be filtered out.

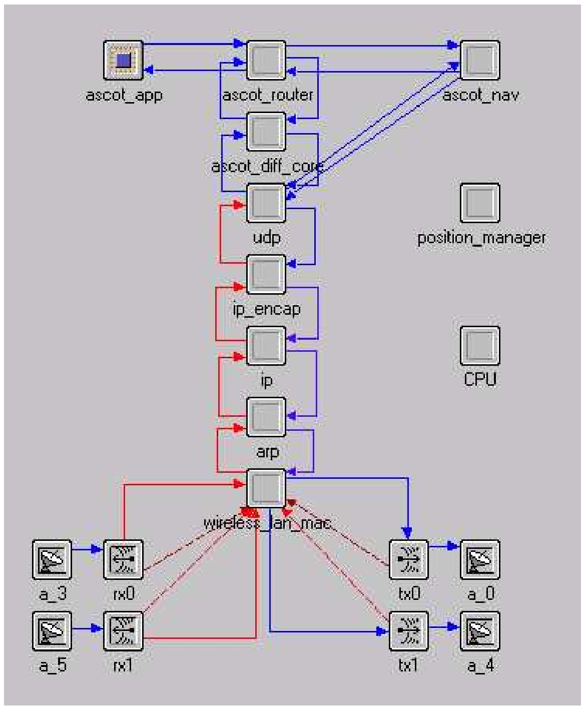


Figure 1. ASCoT Implementation in OPNET.

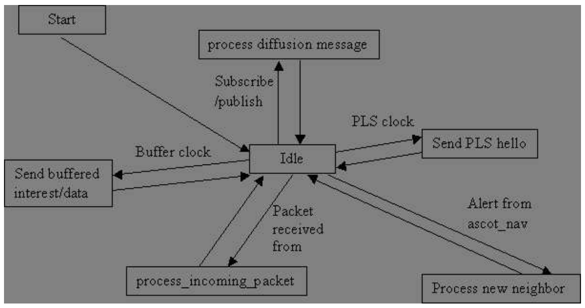
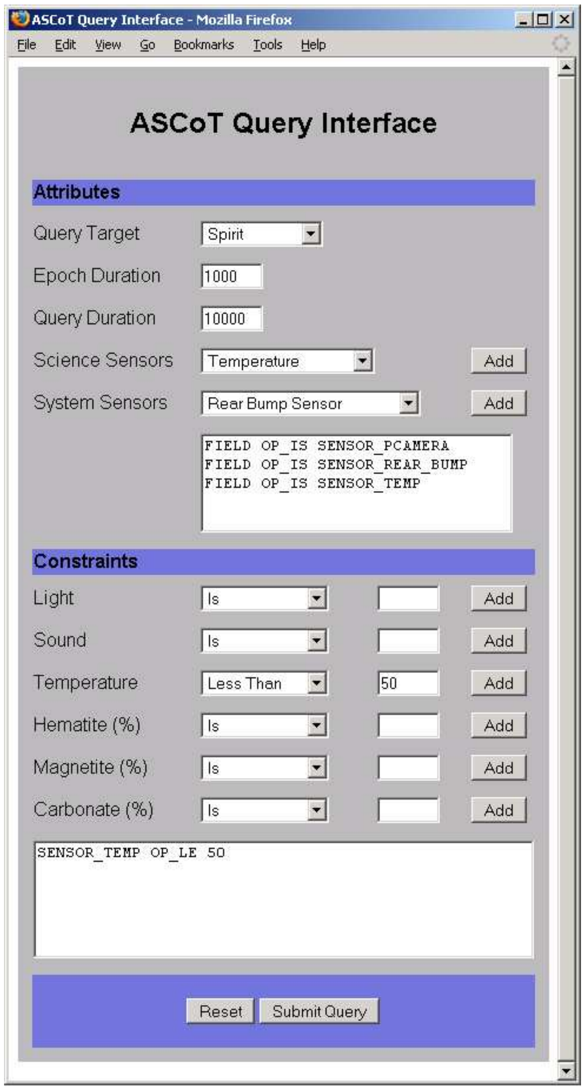


Figure 2. ASCoT Router state diagram showing major functionalities.


ASCoT Query Interface - Mozilla Firefox
File Edit View Go Bookmarks Tools Help

ASCoT Query Interface

Attributes

Query Target	Spirit	
Epoch Duration	1000	
Query Duration	10000	
Science Sensors	Temperature	Add
System Sensors	Rear Bump Sensor	Add

```
FIELD OP_IS SENSOR_PCAMERA
FIELD OP_IS SENSOR_REAR_BUMP
FIELD OP_IS SENSOR_TEMP
```

Constraints

Light	Is		Add
Sound	Is		Add
Temperature	Less Than	50	Add
Hematite (%)	Is		Add
Magnetite (%)	Is		Add
Carbonate (%)	Is		Add

```
SENSOR_TEMP OP_LE 50
```

Figure 3. ASCoT Query Interface.

5. Implementation structure

We implemented ASCoT in OPNET. There are three basic services PLS must perform: (a) Dissemination of link state information, (b) Path computation and (c) Message forwarding. Here we will describe the overall ASCoT implementation and explain where and how these functionalities are implemented.

An ASCoT node consists of the following processes: ascot_app, ascot_router, ascot_nav, position_manager, ascot_diff_core, ascot_mac and a set of OPNET built-in processes: udp, ip_encap, ip, arp, and transmitter and receiver antenna modules. Figure 1 is a screenshot from OPNET node model view that shows how these processes are interconnected. The screenshot shows a node model with two sets of antenna. In our scenario, a base station on Mars is a node of this type. The base station communicates with satellites using one set of antenna and uses the second set to communicate with sensors and rovers on Mars on a different carrier frequency. Satellite and relay nodes have a single set of antenna.

The ASCoT application is implemented in the ascot_app process model, which behaves differently depending on the type of node. Nodes that send out queries are called sink nodes and the nodes that send out sensor data are called source nodes. On a sink node, ascot_app process is responsible for generating interests and receiving data matching that interest. On a source node, this process receives interests and sends out data matching the interest. Ascot_app allows an external interface for injecting queries and receiving data as well. We have implemented this using a webpage based query injection and data display (Figure 3).

Most of the intelligence of routing is built in the ascot_router process. There is a flag that determines the type of routing to enable. The current version of ascot_router supports three types of routing: (1) gradient-based One-Phase-Pull, (2) geographic greedy routing, and (3) PLS routing. This provides the ascot_subscribe (used for sending a query to the network) and ascot_send (used for sending data in response to a query) interfaces to the ascot_app. Figure 2 shows the state machine of the router.

The router initiates link information dissemination every time PLS clock expires. When a PLS hello message is sent out, a chain of PLS update messages are triggered which results in nodes discovering all the links in the network. Before an interest or data message is sent out, link availability information is used to buffer that message until the link becomes available. A periodic buffer clock triggers a message transmission if links are available for buffered messages. Packets received from diff_core get forwarded to the application or sent back down to be forwarded to other nodes.

Path calculation is implemented as an external function to which router passes in the list of all known links. The calculate path function then creates a graph, runs the shortest path algorithm with a given metric and returns the next hop on a path to the node closest to the destination.

Diff_core provides a layer for future scheduling additions to ASCoT. Currently diff_core acts as a pass through layer. Position_manager manages object positions using position vector generated by STK. Ascot_nav sends alerts to ascot_router whenever a new neighbor is discovered or an existing neighbor is no longer responding to beacons.

The node model uses a standard IP-based stack just as a one-hop connection service. One could substitute any connection service depending on the platform. All the routing intelligence is built in the ascot_router and IP is used to deliver a message to the next hop that ascot_router has determined to be on the best path to the destination. The current simulator uses IP-based connection service because we wanted to run ASCoT in a scenario as close as possible to the reality. Satellites will have standard IP-stack on them and it is desirable to build a service that is compatible with existing services.

We had to build our own Media Access Control (MAC) layer because OPNET does not have a MAC layer model that can handle large latency interplanetary links without heavy modification. Because of assumptions on maximum latency, the default MAC often behaved unpredictably when it was asked to deliver a packet on a link with latency in the order of thousand seconds The chatty nature of the standard MAC protocol was also entirely inappropriate. The MAC we wrote provides a very basic delivery mechanism and works for any range of latencies and any number of transmitters/receivers. While this MAC would be insufficient for a general purpose use, it is adequate as a basic delivery mechanism.

Low level link negotiation, such as antenna pointing and link characterization is not modeled (the required data is taken from STK). Trajectory and link availability calculations are also taken from STK and are provided to the OPNET code through the navigation module which simulates a run-time environment. All coordinate translation is currently performed in STK (OPNET operates solely in an Earth-centered fixed coordinate system). In a real implementation, these calculations would be performed by the onboard computer.

While we place ASCoT on top of UDP and IP, this is done primarily for compatibility with other likely space assets. The framing, addressing and error checking services that these modules provide could be easily achieved through other means.

Demo Scenario

The scenario with which we demonstrate our current routing capabilities envisions a PI requesting data directly from a rover on Mars. Through a web interface, the user selects the type of data desired and the application software running on his workstation sends the request to Mars. In the simulation, the originating point of the request is the Madrid DSN station: we do not model the communication between the DSN node and the PI. We assume that communication will be handled through standard internet channels. If the DSN station does not have direct visibility of Mars at the time, the request is relayed through one or more Earth satellites, then through a Mars orbiter (if necessary) before it reaches the base station on Mars. The base station serves as a gateway for the two rovers in its vicinity (Spirit and Opportunity) that will provide the data. The topology of this scenario vaguely resembles current Mars rover communication scenarios, but with more intermediate nodes than currently in use to demonstrate the power of PLS routing. For example, instead of choosing the DSN dish that has Mars in view (which may be busy with other tasks) all requests are sent from Madrid, requiring the use of Earth relays.

Demonstrated ASCoT Features

This scenario highlights four strengths of ASCoT.

1. Its ability to deal with heterogeneous hardware. The Earth and Mars relay satellites, as well as the Mars base station, may utilize completely different transmission hardware. As long as they have a form of the IP stack and ASCoT running on top, the communication occurs seamlessly.
2. The reliability of the protocol in the face of dynamic network topology, short link duration and long link latencies. As relay stations become occluded or occupied with tasks of higher priority (or orientation requirements force them to cut the current link), ASCoT automatically selects a different path that uses orbiters that become available.
3. PLS routing exploits future link information to predictively route on paths that become available just as the message travels along, and buffers messages as it waits for the

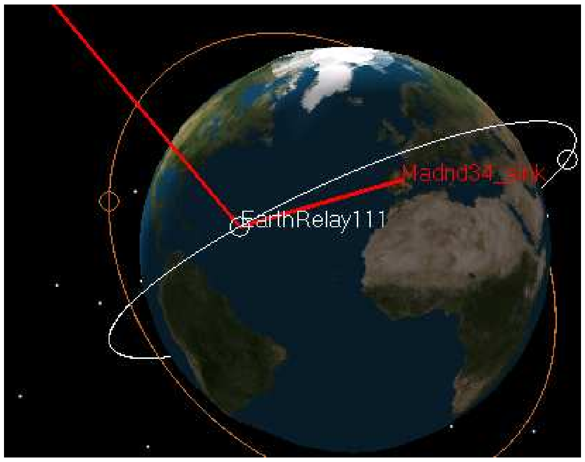


Figure 4. Earth View of the network. Data is streaming from Mars, and is being routed through EarthRelay111. The thickness of the line indicates approximately the amount of bandwidth being used.

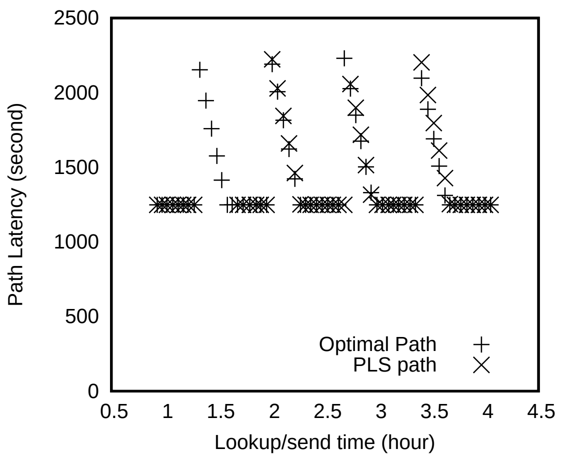


Figure 6. Path latencies. Planetary and satellite positions at lookup time impact path latency.

links to come up if necessary.
4. Automatic and efficient path discovery and link information distribution that allows PLS path computation to occur. Several parameters allow this behavior to be tuned to the current network state.

Simulation Components

We now describe various simulation components in our demonstration scenario.

PI—The PI specifies the source and constraints of the data using a web interface. In the demo, six types of data are available: a mini-Thermal Emission Spectrometer (image), Panoramic camera (image), microscopic imager (image), light, sound, and temperature. The simulation manager accepts the data from the web page and passes it to the application process inside the OPNET simulation.

DSN—The Madrid DSN node participates in PLS and uses it to select the best path to the selected destination. If Mars is not visible at the time of the query, the node will attempt to route through an Earth satellite. If none of those are currently visible (but are anticipated) it will buffer the interest and send it out at a later time. We used a carrier frequency of 7.5GHz and a downlink and uplink bandwidths of 0.2 Mb/s and 10Kb/s respectively on these links.

Earth Orbiters—Six satellites in the MEO orbit can relay the signal to and from the vicinity of Mars. There are times when none of these satellites are in view, and the DSN station must buffer the interest. The antenna characteristics for both Earth satellites and the DSN station are chosen such that they can transmit to any Mars satellite that is in line of sight. Figure 4 shows the earth view of satellites and the DSN station while data is coming from Mars.

Mars Orbiters—Three satellites in Areosynchronous and five satellites in moderately inclined lower Mars orbit are available for relaying the packets to and from the base station that is near the two data collection rovers. This redundant (and not very realistic) set up was chosen to allow the possibility of more complicated paths.

Mars Base station—All communication with the rovers happens through the base station. It has two sets of network interfaces: one to talk to the Mars orbiters using 7.5 GHz frequency and another to communication with the surface rovers using the 802.11 radio. By going through a dedicated base station, the rovers can ensure continuous accessibility and further reduce their communication power requirements.

Surface Rovers—The scenario is set up such that the two rovers (Spirit and Opportunity) can talk only to the base station, but those links are available continuously. Upon receiving a query from the PI, the rovers start transmitting the data. Filters registered in the diffusion core ensure that only matching data is sent as a response. Figure 5 shows the Mars view with rovers and orbiters when messages are being sent to Earth.

6. Simulation Experiments

In this section, we compare the quality of the best possible route an engineer can manually program on to a node with the path automatically discovered by PLS. Existing routing solutions require manual configuration. Network designers compute ahead of time when links will be available and configure each node manually to receive or transmit messages at those times. PLS on the other hand automates this process. It automatically discovers the link schedules and propagates them. It also communicates to other nodes any unexpected changes in link schedules. This automated system relies on messages exchange with other nodes over lossy links orchestrated with an asynchronous protocol. We predict that PLS

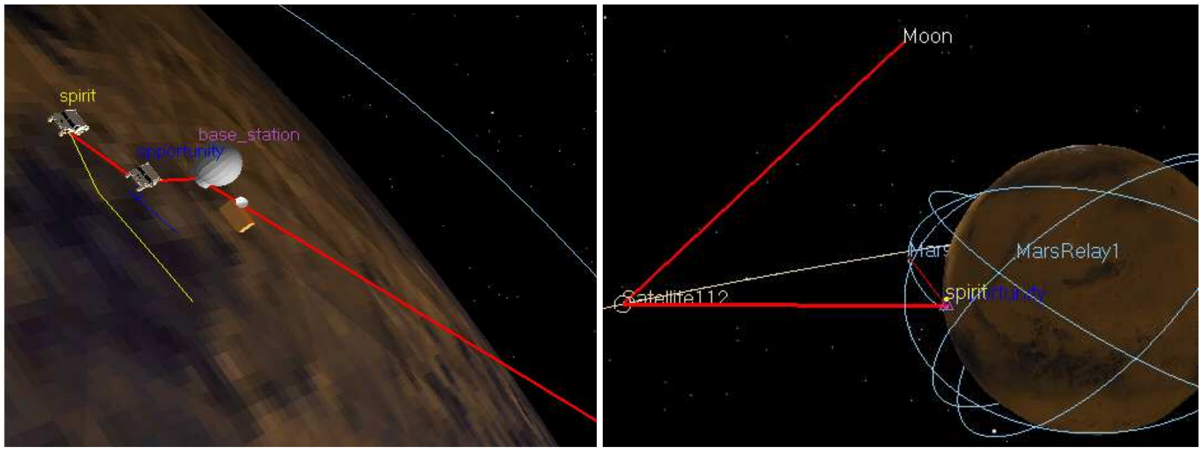


Figure 5. The Mars View of a data transmission. Data is sent from Spirit through Opportunity, the base station, and Satellite112. This illustrates a path that does not involve buffering at any intermediate points, and the latency of which is effectively the propagation delay between Earth and Mars (1250s at the date simulated). An interest that has been buffered on MarsRelay5 is now also being transmitted to the base station.

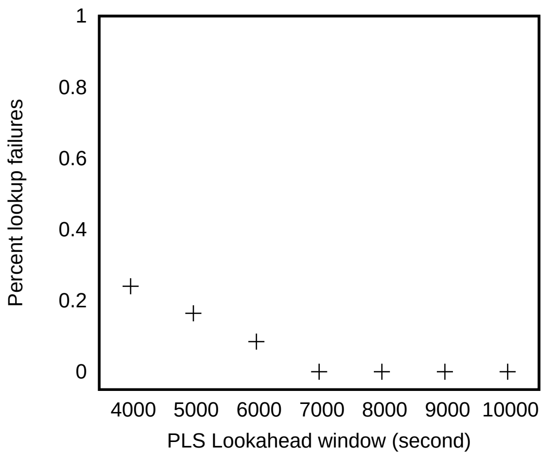


Figure 7. Lookup failures with different lookahead windows.

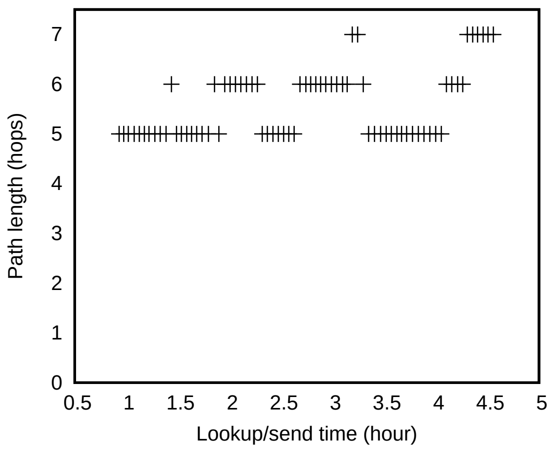


Figure 8. Path lengths (number of hops) on a path selected by PLS with a lookahead window of 7000 seconds

will find a path almost as good as the best path even with this incomplete information.

For our comparison study, we define PLS path to be the path discovered by PLS. We define Optimal path to be the best path considering complete information about future link availability. PLS uses a limited lookahead of link schedules to compose a PLS path update message. This limited lookahead simulates the finite knowledge of future information that is achievable. Furthermore, due to collisions and errors, some PLS messages will be dropped. Optimal path computation uses the entire link schedules generated by STK to compute a path.

We were able to demonstrate that a limited lookahead of 7000 seconds is enough for PLS to be able to compute the path from sink on Earth to a rover on Mars at all times (Figure 7). Furthermore, whenever PLS discovers a path, the path is always the least latency path (Figure 6). We verified this by comparing the PLS path with the Optimal path for each lookup. The intuition is that to compute the least latency path we need not look too far into the future because a path that uses link that is available far into the future, instead of links available in the near future, can not form the least latency path. Thus, optimal path computation never uses link schedules for distant future, rather it forms a path using the links that are available in the near future. In reality some path update messages might get dropped which will result in PLS computing suboptimal paths.

Even with a few relay satellites, PLS was able to compute paths with latencies close to the straight-line propagation delay between the sink and the target rover. Due to planetary and satellite dynamics, sometimes the links between Mars and Earth relay satellites can get occluded. The peaks in the graph is due to occlusion which forces PLS to queue the packet before it can be forwarded to the next hop. PLS was able to automatically find the satellites relays to form a path that provided a near optimal latency performance. These peaks are artifact of the scenario rather than PLS. We also found that paths as long as seven hops were being used to deliver the interest messages (Figure 8).

7. Conclusions and Future Work

ASCoT is a new data-centric and position-based routing architecture for future space science missions. These missions involve an increasingly large number of satellites which render the current manual and static routing practice ineffective because those techniques do not scale with the number of nodes. ASCoT dynamically discovers and computes the best paths to the destinations that are identified by locations rather than nodes. ASCoT departs from traditional address-centric communication and uses a data-centric architecture to enable energy efficient and low latency operation.

We have hinted that routing and scheduling might interact in complex ways. As we design scheduling and resource allocation strategies, we will continue to study those interactions. Further, we intend to do more simulation experiments with more scenarios with more planets and satellites. We also plan to study the routing state and route stability to evaluate the PLS paths.

Our study shows that even with a limited knowledge about the future, a dynamic approach such as ASCoT can discover paths that can be used to forward messages successfully and efficiently. Our first exploration of integrating link state style routing with data-centric routing looks promising.

8. Acknowledgements

This work is supported in part by the National Aeronautics and Space Administration (NASA) through the NASA Space Communications Project (NAS3-03089). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA.

References

[1] Australian storm delays mars rover's work, cnews. `http://cnews.canoe.ca/CNEWS/Space/2004/01/21/320180-ap.html`.

[2] Deep Space Network. `http://deepspace.jpl.nasa.gov/dsn/`.

[3] OPNET Modeler, OPNET Technologies, Inc. `http://www.opnet.com/`.

[4] Satellite Took Kit, Analytical Graphics, Inc. `http://www.stk.com`.

[5] Solar flare hobbles japanese communications satellite. `http://www.space.com/news/kodama_down_031029.html`.

[6] Ozgur B. Akan, Jian Fang, and Ian F. Akyildiz. Performance of TCP protocols in deep space communication networks. *IEEE Communications Letters, vol. 6, no. 11*, pages 478 – 480, November 2002.

[7] Ian F. Akyildiz, Eylem Ekici, and Michael D. Bender. MLSR: A Novel Routing Algorithm for Multilayered Satellite IP Networks. *IEEE/ACM Transactions on networking, Vol. 10, No. 3, June 2002*.

[8] S. Corson and J. Macker. RFC 2501: Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, January 1999.

[9] M. De Sanctis, E. Cianca, and M. Ruggieri. IP-based routing algorithms for LEO satellite networks in near-polar orbits. In *IEEE Aerospace Conference, 2003*.

[10] R. Durst, G. Miller, and E. Travis. TCP Extensions for Space Communications. *Proceedings of the second annual international conference on Mobile computing and networking, White Plains, NY USA*, page 15, 1996.

[11] Robert C. Durst, Patrick D. Feighery, and Keith L. Scott. Why not use the Standard Internet Suite for the Interplanetary Internet? Interplanetary Internet Study Seminar, California Institute of Technology, 25 October 1999, `http://www.ipnsig.org/reports/TCP_IP.pdf`.

[12] Eylem Ekici, Ian F. Akyildiz, and Michael D. Bender. A distributed routing algorithm for datagram traffic in leo satelitte networks. *IEEE/ACM Trans. Netw.*, 9(2):137–147, 2001.

[13] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM Press, 2003.

[14] H. Chang and B. Kim and C. Lee and Y. Choi and S. Min and H. Yang and C. Kim. Topological Design and Routing for Low-Earth Orbit Satellite Networks. In *Proceedings of IEEE GLOBECOM*, pages 529–535. IEEE, 1995.

[15] John Heidemann, Fabio Silva, and Deborah Estrin. Matching Data Dissemination Algorithms to Application Requirements. Technical Report ISI-TR-571, USC/Information Sciences Institute, April 2003.

[16] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, pages 146–159, Chateau Lake Louise, Banff, Alberta, Canada, October

2001. ACM.

[17] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Networking. In *Proceedings of the ACM SIGCOMM Conference*, Portland, USA, August / September 2004. ACM.

[18] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[19] T. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking, vol. 5 no 3*, pages 336 – 350, July 1997.

[20] Christian Lochert, Hannes Hartenstein, Jing Tian, Holger F, Dagmar Hermann, and Martin Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2003*, June.

[21] J. Moy. RFC 2328: OSPF Version 2, April 1998.

[22] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, USA, February 1999.

[23] J. Postel. User Datagram Protocol. *Network Information Center RFC 768*, August 1980.

[24] J. Postel. Internet Protocol. *Network Information Center RFC 791*, September 1981.

[25] J. Postel. Transmission Control Protocol. *Network Information Center RFC 793*, September 1981.

[26] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mob. Netw. Appl.*, 8(4):427–442, 2003.

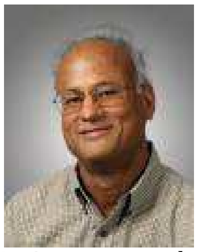
[27] Y. Rekhter and T. Li. RFC 1771: A Border Gateway Protocol 4 (BGP-4), March 1995. Obsoletes RFC1654.

[28] K. Scott and S. Burleigh. Bundle protocol specification. *Internet Draft, Delay Tolerant Networking Research Group*, July 2004.

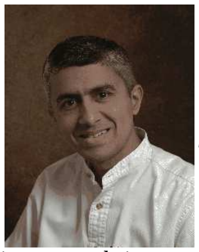
[29] Archana Sekhar, Manoj B. S., and Siva Ram Murthy. MARVIN: Movement-Aware Routing oVer Interplanetary Networks. In *The First International Conference on Sensor and Ad Hoc Communications and Networks SECON, October 2004*.

Omprakash Gnawali, USC. *Omprakash Gnawali received his S.B. and M.Eng. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology. He is a Ph.D. student in Computer Science at the University of Southern California. His research interests include computer networks, distributed systems, and wireless sensor networks.*

Mike Polyakov, LM-ATC. *Mr. Polyakov received his MEng in computer science from Cornell University. Since then he has worked at Lockheed Martin under Dr. Bose on the ASCoT and DyMND projects, focusing on the modeling and simulation aspect of the projects.*



Dr. Prasanta Bose, LM-ATC. *Dr. Bose received his PhD in computer science from University of Southern California and MS in Computer Science from University of Massachusetts. He has worked at Texas Instruments, NASA JPL and was an Assistant Professor at the George Mason University prior to joining the ATC. Dr. Bose leads the Networked Embedded Autonomy Technology (NEAT) group in ATC, Lockheed Martin Space Systems. He is the PI of the NASA Autonomous Space Communications Technology (ASCoT) project and the DARPA DyMND project focused on developing robust coordination services for in-network control and coordina-tion of large collection of resource constrained and wireless networked mobile air and space assets. He is also the Co-PI on a NASA Living with Star (LWS) project investigating distributed and collaborative science infrastructures.*



Ramesh Govindan, USC *Ramesh Govindan received his B. Tech. degree from the Indian Institute of Technology at Madras, and his M.S. and Ph.D. degrees from the University of California at Berkeley. He is an Associate Professor in the Computer Science Department at the University of Southern California. His research interests include Internet routing and topology, and wireless sensor networks.*