

Next-Generation Space Internet: Prototype Implementation

James Noles
Global Science and Technology, Inc.
6411 Ivy Lane, Suite 300
Greenbelt, MD 20770

Keith Scott, Mary Jo Zukoski
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102-3481

Howard Weiss
SPARTA, Inc.
9861 Broken Land Parkway
Columbia, MD 21046

Abstract¹ – Future orbiting sensor constellations will consist of tens, hundreds, or even thousands of spacecraft, each capable of generating enormous amounts of data. Managing these constellations will require that the spacecraft be autonomous, deciding without human intervention what observations to make and asynchronously reporting results. The need for a robust and efficient shared network infrastructure combined with the desire to provide direct connectivity between orbiting instruments and scientists on the Internet argues for connecting these sensor constellations directly to the Internet.

This paper describes our implementation of an architecture to connect sensor constellations to the Internet based on open standards (Internet and CCSDS protocols). The main elements are: 1) Security (access control, authentication, and encryption) provided by a mix of IPSEC and the more 'space-link-friendly' SCPS-SP security protocols 2) a MobileIP implementation that takes advantage of scheduled contacts to reduce the overhead involved in setting up MobileIP tunnels 3) Resource reservation mechanisms that allow applications to ensure that data are not lost and/or are delivered in a timely manner and 4) a link-layer driver that allows us to easily connect ground and space networks.

In a previous paper [1] we verified the feasibility of the above elements in isolation and quantified the performance enhancements they provided through simulation and rapid prototyping. We have recently integrated the elements into an end-to-end prototype, and describe here their implementations. We include information on the standards used, how they have been modified for the particular environment, and the steps required to integrate them into a complete system. We note that the changes to existing standards are minimal and are confined to a small set of hosts that could reasonably be assumed to be under the control of the spacecraft operators; we require no modification to the underlying bearer network, which we assume to be the Internet.

I. INTRODUCTION

A. Communications Paradigms

Current space missions rely on highly managed communications. This works well when there are relatively few space assets, where communication with each can be independently scheduled, and where data volumes are predictable. This model does not hold in the sensor web of

semi-autonomous spacecraft that we see as the future of Earth-observing science.

Orbiting sensor webs of semi-autonomous spacecraft will require advanced networks to support their dynamic and unpredictable communications requirements. Multiple spacecraft with highly efficient protocols will need to dynamically share downlinks to support both "quick-look" data to scientists and to respond to unscheduled events of interest.

The Internet provides a good example of a scalable, robust, efficient, and adaptive network architecture that could support future orbiting sensor webs. Use of Internet Protocols in the space segment is particularly attractive because it provides for easy interconnection with the terrestrial Internet. This will allow scientists more control over their experiments and faster, easier access to their data.

Section II provides an overview of the mechanisms necessary to provide real-time communications between orbiting elements and hosts on the Internet. Section III describes the system architecture, and section IV the implementation details of the four areas of section II. Section V contains concluding remarks and areas of future study.

II. MECHANISMS FOR REAL-TIME COMMUNICATIONS

Four main elements are required in order for us to connect space assets with ground-based PIs: 1) Security, 2) MobileIP, 3) Resource Reservation, and 4) an efficient and space-tuned link layer. This section gives a brief outline of these mechanisms; section III describes in detail their instantiations in the end-to-end prototype.

1) Security

Allowing direct access to space assets from hosts on the Internet requires security. Authentication to ensure that only authorized users are granted access to the space link and encryption to ensure the privacy of science data are both primary concerns. Under joint DoD/NASA sponsorship, a set of Internet protocols were specified for use in bandwidth constrained environments. This work, known collectively as the Space Communications Protocol Suite (SCPS) includes a Security Protocol, known as SCPS-SP [2]. SCPS-SP provides the same security services as its Internet counterpart, IPsec, but with significantly less overhead. Transport-layer performance-enhancing proxies developed as part of the SCPS work can also host security gateways, translating

This work was performed under contract to NASA Jet Propulsion Laboratory in support of ESTO/AIST task AIST-99-0031

between IPSEC and SCPS-SP. We use these gateways to allow terrestrial users to employ IPSEC while maintaining the efficiency of SCPS-SP across the space link.

During the development of the SCPS protocols, the key management and algorithm aspects the security protocol were not addressed. As a result, this work has undertaken the addition of encryption algorithms to the existing SCPS gateways, has done an analysis of key management techniques, and has concluded that the use of the Internet Key Exchange (IKE), despite its overhead, is preferable to the development of a custom key exchange protocol. A version of IKE is being modified to operate with SCPS-SP on the SCPS gateways.

2) MobileIP

Mobile IP, as specified in RFC 2002 [3], was designed to permit mobile agents to move randomly through the Internet while still receiving datagrams at a fixed address. A moving spacecraft, connecting first to one ground station and then to another, fits the definition of a mobile agent. It is natural, then, to want to apply MobileIP to the spacecraft environment. However, the protocol overhead required for each ground station connection is not conducive for real time communication. On the other hand, spacecraft do not move randomly. Contacts between spacecraft and ground stations are scheduled, with *a priori* agreement of established state. Our work takes advantage of this agreement and has implemented extensions to MobileIP, allowing for real-time user-to-payload interaction.

3) Resource Reservation

Because of the expense and scarcity of bandwidth between space assets and the ground, preventing data loss and subsequent retransmission is of great concern. This is particularly difficult in the case where multiple semi-autonomous spacecraft need to share communications resources in a controlled manner and in an automated fashion. The Resource reSerVation Protocol (RSVP) used in the Internet can prevent data loss due to congestion by allowing flows to reserve bandwidth and buffer space in intermediate routers. Our work with RSVP has been to adapt it to our environment and to integrate it with the other work packages, specifically MobileIP and the security gateways mentioned above.

4) Efficient Data Link Layer for Space Communications

Conventional CCSDS links transport relatively statically scheduled streams of CCSDS data units encapsulated in virtual channels that may be transferred over several physical links to a control or processing center. In an internetworking environment, however, the virtual channels will transport dynamically changing streams of Internet Protocol packets that are extracted and routed at each spacecraft in a network before being routed to the ground. We have developed an efficient driver for packing, transporting, and extracting IP

packets for each space link in the Internetwork. This driver implements the CCSDS Telecommand and Telemetry [4, 5, 6] standard mechanisms for multiplexing and demultiplexing IP packets into/from CCSDS data links.

III. SYSTEM ARCHITECTURE

The general system architecture is shown in figure 1, where different horizontal bands represent different views of the system. At the top are the physical components: Investigators, a Control Center, a Ground Station, and possibly multiple spacecraft between the ground station and the destination. These represent the minimum set of hosts required to make the system function, and the set of hosts on which the software we have developed runs. Note that we make no assumption about the networks connecting the control center to either investigators or the groundstation. One advantage of our architecture is that these can be completely open networks, including the Internet.

The ‘security’ band highlights the security aspects of the architecture, namely a set of security gateways that translate between IPSEC and the more bit-efficient SCPS-SP protocols. The IP band highlights the need for two IP tunnels. One tunnel handles forward (from the PI to the spacecraft) traffic, and the other ensures that reverse traffic is routed through the security gateway at the control center. The bottom band illustrates how the different flows appear to RSVP. In particular, a single unidirectional flow has four states: encapsulated in IPSEC, encapsulated in SCPS-SP and tunneled inside a MobileIP tunnel, encapsulated in SCPS-SP, and “native” (unencrypted and not tunneled) While we have not implemented security on board the destination spacecraft in our prototype, its addition would be a simple matter.

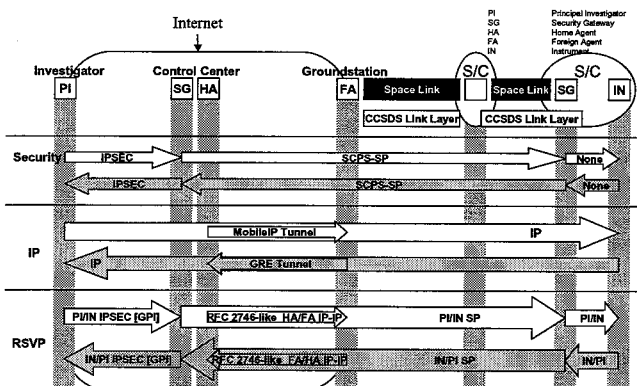


Fig. 1: System Architecture

For our proof-of-concept demonstration we chose Intel-based machines running the FreeBSD (4.2) and Linux (2.4.x) operating systems to emulate the various system components shown in Fig. 1. Each machine was equipped with three 100Mbps network interfaces, one of which was used to control the processes running on the machine and the other

two of which supported test traffic. One of the machines adjacent to the simulated space links also hosted a software delay and error injector (NISTNet).

IV. IMPLEMENTATION DETAILS

This section discusses the implementation details of our prototype system. We note here that the basic requirements to implement the architecture are: security gateways at the control center and onboard each spacecraft requiring security; a MobileIP Foreign Agent in each groundstation; and an RSVP-capable network connecting the groundstation to the Internet users.

A. Security

Once relegated to back-burner status in the IT and network communities, security has recently become a hot topic due to ubiquitous Internet connectivity and ubiquitous Internet security breaches. From the outset, the SCPS design, specification, and development took security into account. As a result, the SCPS Security Protocol (SCPS-SP) was developed.

At the time SCPS-SP was being designed, there was not a finished IPsec protocol specification. There were several earlier attempts to define Internet security protocols, most notably by the DoD in their *Security Protocol at Layer Three* (SP-3) specification. However, none of those security protocols took into account bandwidth constraints; all of them assumed large amounts of terrestrial bandwidth. Conversely, SCPS and the SCPS Security Protocol were being designed for use in limited bandwidth environments with high delays and high errors. As a result, SCPS-SP became a “light-weight” cousin of SP-3 and the emerging IETF IPsec.

The reference implementation of the SCPS protocols includes a transport-layer gateway that can convert from Internet protocols to SCPS versions of those protocols. As part of this work, the SCPS gateways were extended to provide full security interoperability between IPsec and SCPS-SP. Tests were performed over the Internet with British collaborators using Internet-standard IPsec to a SCPS gateway which in turn used SCPS-SP to communicate with a peer SCPS

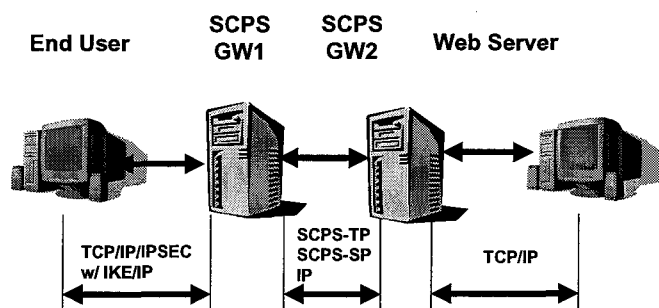


Figure 1: IPsec to SCPS-SP Security Interoperability gateway. The second gateway then removed all security for

delivery to a simulated on-board instrument (or in this test case, a web server). The setup is shown in fig. 2.

The SCPS-SP specification is algorithm neutral. That is, it specifies the protocol but does not specify the algorithms to be used to provide security. In the SCPS-SP reference implementation, for testing purposes, a simple XOR algorithm was used in place of an actual encryption algorithm. For this project we have added two encryption algorithms to the SCPS-SP reference code and the SCPS gateway: the (former) U.S. Data Encryption Standard (DES), and a stronger variant known as triple DES (3DES).

Another aspect of security that was not addressed in the earlier SCPS work was key management and key distribution. Testing of SCPS-SP was carried out with manually distributed and installed cryptographic keys. For large-scale use, an automated means of distributing and managing keys is necessary. In order to provide key management for the space environment, a tradeoff study was performed to examine what key management protocols were already in use, their strengths and weaknesses, and whether or not they could be used or adapted for the space community. Alternatively, if none of the existing key management protocols could be used, then a custom protocol would have to be designed.

The Internet community has developed a protocol known as the Internet Key Exchange (IKE). This was one of the leading contenders for use in the space environment from several perspectives. First, it is the Internet standard and therefore implementations would exist via the open-source community as well as commercially off-the-shelf. Its adoption would also allow complete interoperability between ground-based and space-based assets requiring cryptographic keys. However, IKE is a “heavy-weight” protocol with a considerable amount of overhead and it uses several round-trip messages to perform its work. As a result of the key management analysis study, it was determined that although IKE had considerable overhead, it was invoked infrequently and therefore the overhead would be bearable over constrained bandwidth links.

As a result of the analysis, the version of IKE written for FreeBSD is being modified for use with SCPS-SP in the SCPS gateway environment. The *Racoon* implementation of IKE assumes the use of an internal, kernel-owned key cache that is not used by the out-of-kernel SCPS protocols. As a result, changes to both SCPS-SP and *Racoon* are being made to allow the key exchange to occur and for the SCPS gateway to obtain the keys for use in secure communications.

B. Extensions to MobileIP

As stated in the Introduction, MobileIP is a natural “fit”, conceptually, to a space-ground communications system. And while certain characteristics of such a system limit the direct applicability of the protocol, other characteristics – namely, *a priori* knowledge of contact periods, ground station

addresses, etc. – provide for minimal changes to overcome the limitations.

The basic idea of standard MobileIP is that if the mobile node is away from its home, a care-of address is associated with it that provides information as to its current point of attachment to the Internet. The mobile node registers this care-of address with its home agent, who tunnels datagrams destined for the mobile agent to this care-of address. MobileIP specifies a preferred method of acquiring a care-of address through foreign agents, where the foreign agent acts as the endpoint of the tunnel, decapsulates received datagrams, and delivers them to the mobile node. This setup, while straightforward, may be too time- and bandwidth-consuming in the limited resource environment of spacecraft communication.

Consider the spacecraft to be a mobile node, the ground station as foreign agent, and the control center as home agent. Then making use of the a priori knowledge of state, the ground station (in its role as foreign agent) can act as a proxy for the mobile node. Knowing that the spacecraft is about to make contact, the ground station pre-registers with the control center and sets up the tunnel. The objective is for datagrams destined for the spacecraft to arrive as the space-ground contact is established. (Note: This is an application of the Just-in-Time scheduling used in many manufacturing operations.) From its perspective, the spacecraft assumes foreign agent and tunnel functionality will be in place and prepares any outgoing datagrams for download.

For connection termination, the next ground station will begin the proxy registration slightly before the current station reaches loss-of-signal (LOS). This prevents an undue number of dropped datagrams due to misrouting – which can result if the current tunnel is still in place after LOS is achieved. If the next tunnel is established slightly before its ground station makes contact, the station will queue the data.

We have incorporated these modifications to the base protocol of the Dynamics – HUT MobileIP system [7]. This is a Linux implementation of Mobile IP, which we installed on a Red Hat 7.1 distribution. The mobile node, foreign agent and home agent were run on different machines. The IP-in-IP tunneling support was loaded and configured as a kernel module.

C. Resource Reservation

RSVP is an end-to-end protocol for resource reservation, and hence touches all of the other work areas. This section describes the modifications to the ISI reference implementation of RSVP [8] that allow it to function in the architecture described above.

The security gateways present a special challenge to RSVP, as the RSVP design did not anticipate cases where the IP protocol field changes in transit. To preserve reservations as flows change security measures, the RSVP daemons on the

security gateways were modified to manipulate the IP protocol fields of RSVP filterspec messages. Thus RSVP messages that arrive reserving IPSEC flows leave reserving SCPS-SP flows, and vice versa.

The MobileIP tunnels used to forward data between home and foreign agents also required changes to the RSVP daemons running on those hosts. RFC 2746[9] defines mechanisms to allow RSVP to operate over IP tunnels, but no suitable implementation existed for the Linux operating system (we chose the HUT MobileIP implementation, which runs under Linux). We thus plan to implement the tunnel functionality described in RFC2746 in the Linux RSVP implementation. This requires an IP-in-UDP encapsulation mechanism so that RSVP could distinguish between tunneled flows (all of which have the same source and destination IP addresses – those of the tunnel endpoints).

The Crypto IP Encapsulation (CIPE [10]) distributed with recent Linux kernels provides IP-in-UDP encapsulation (with optional encryption). However the interface required to set up and manage CIPE tunnels is significantly different from that used for managing the IP-in-IP tunnels that the HUT MobileIP daemon expects. In addition, CIPE was designed for VPN-type applications, where manual administration of tunnels is feasible. For our application we needed a system capable of dynamic tunnel management via an application programming interface (API) rather than a configuration file.

We thus chose to modify the IP-in-IP driver to include an extra UDP header. Issues with this approach include changing the Maximum Transfer Unit (MTU) to avoid IP fragmentation, the ability of RSVP to correctly identify the interface (since the IP-in-UDP interfaces are not *physical* interfaces) and the performance of traffic control over these interfaces. The MTU issues are easily solved at the endpoints or the transport-layer gateways, and the Linux RSVP implementation does recognize the virtual drivers as interfaces.

We found that the performance of the various traffic control mechanisms varied widely depending on the operating system and link layer. The goal was to characterize the traffic control characteristics to determine their impact on overall performance. In particular, there are known issues with the class-based queueing (CBQ [11]) mechanisms used in both the Linux and FreeBSD implementations that can cause them to under-perform, especially when examined over short time periods. For a deployed system we would be relying on the traffic shaping of the routers (e.g. Cisco, Juniper, ...), which is considerably better than the Linux or FreeBSD implementations.

For our system, CBQ was provided by the altq package [12] in FreeBSD and by the native 2.4-kernel traffic control [13] in Linux. As a first measure, we simply examined the traffic control mechanisms' abilities to shape traffic. This provides the foundation for resource reservation, as we would like to

be able to provide minimums for both reserved and non-reserved traffic, as well as to allow each to borrow unused bandwidth from the other.

For plain, non-encapsulated IP traffic, the Linux cbq provided much better performance, able to shape traffic to within a few hundred kilobits per second of the target rate, and provided good borrowing properties between classes. The altq implementation was much coarser, generally over-limiting bandwidth by several megabits per second. Linux traffic control performance dropped dramatically when used over an encapsulating interface (either the IP-in-UDP or the CCSDS links). In these cases linux tc's ability to rate control traffic approached altq's. This should not pose significant problems at low data rates, but could significantly impact performance as data rates increase past around 10Mbps.

D. Ethernet Like Link Layer for CCSDS Links

The most common link layer interface for Internet communication is Ethernet. In this implementation a Linux driver for CCSDS was developed that behaves exactly as an Ethernet interface from the viewpoint of the network layer. This driver segments and inserts IP packets into fixed length CCSDS frames and transports them in a virtual channel to a companion driver at the other side of the link. The companion driver then extracts and reassembles the packets and passes them to the local network layer entity as if it were a local Ethernet.

In the laboratory the CCSDS driver uses Ethernet to transport the CCSDS frames across the link. The driver is layered such that the underlying Ethernet link can be easily replaced with other link protocols. For instance, the Ethernet link can be replaced with a serial interface to radio frequency equipment for implementation in a space to space or space to ground implementation. This architecture allows applications to be developed in the laboratory for common Ethernet implementation and then transferred to a CCSDS based link

environment with no changes.

V. CONCLUSIONS AND FUTURE WORK

We have presented an overview of the prototype implementation for the Next Generation Space Internet architecture.

Significant areas of future work remain. These include shearing off the bottom half of the CCSDS link driver to allow it to run over serial interfaces (for easy connectivity to uplink devices) and evaluating system performance at higher data rates.

REFERENCES

- [1] Noles, J., Scott, K., Weiss, H., and Zukoski, M.J., Next Generation Space Internet, ESTC conference, August 2001, College Park, MD.
- [2] Space Communications Protocol Standards - Security Protocol, CCSDS 713.5-B-1, CCSDS, May 1999.
- [3] C. Perkins, "IP Mobility Support", RFC 2002, October 1996.
- [4] Telecommand Part 1 -- Channel Service, CCSDS 201.0-B-3, June 2000.
- [5] Telecommand Part 2 -- Data Routing Service, CCSDS 202.0-B-3, June 2001
- [6] Packet Telemetry, CCSDS 102.0-B-5, November 2000
- [7] MobileIP Implementation: Dynamics - HUT MobileIP, <http://www.cs.hut.fi/Research/Dynamics/>.
- [8] Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," RFC 2205, September 1997, Proposed Standard.
- [9] Terzis, A, Krawczyk, J., Wroclawski, J., and Zhang, L., "RSVP Operation Over IP Tunnels," RFC2746, January 2000.
- [10] <http://sites.inka.de/sites/bigred/devel/cipe.html>
- [11] Floyd, S., and Jacobson, V., "Link-sharing and Resource Management Models for Packet Networks," IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, August 1995.
- [12] Kenjiro Cho, *The Design and Implementation of the ALTQ Traffic Management System*, Ph.D. dissertation, Keio University. January 2001.
- [13] Hubert, Bert, Maxwell, Gregory, van Mook, Remco, van Oosterhout, Martijn, Schroeder, Paul B., and Spaans, Jasper, "Linux Advanced Routing & Traffic Control HOWTO", <http://lartc.org>

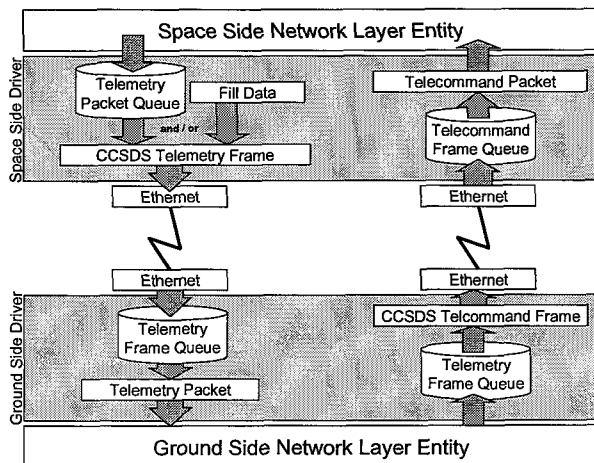


Figure 3: CCSDS Link Driver