Next-Generation Space Internet: Standardization and 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.
7075 Samuel Morse Drive
Columbia, MD 21046

Abstract[1] -- 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 the spacecraft to 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 using IP and IP-based protocols end-to-end.

This paper describes a standards-based architecture for end-to-end IP between orbiting assets and Internet-based hosts. The main elements are: 1) Security provided by a mix of IPSEC and the more bit-efficient SCPS-SP security protocols 2) a MobileIP implementation that takes advantage of scheduled connectivity to reduce overhead and 3) Resource reservation to limit loss and hence increase the goodput across the space-to-ground link. We have also developed a link-layer driver that allows us to run standard space link telecommand and telemetry protocols over Ethernet for easy lab integration of instruments.

In a previous paper [1] we verified the feasibility of the above elements in isolation and quantified the performance enhancements they provided though simulation and rapid prototyping. This paper describes the current state of standardization of the NGSI protocols within the Consultative Committee for Space Data Systems (CCSDS) and the current system demonstration. The NGSI technologies are being tested in a realistic environment, with the emulated instrument, ground station, control center, and principal investigator distributed across the country.

I. INTRODUCTION

Current space missions rely on highly managed communications. There is no direct interaction between users (investigators or data recipients) and spacecraft instruments. All such interactions are filtered through the spacecraft control center and usually involve human processing and intervention. This works well when there are relatively few space assets, where communication with each can be independently scheduled, and where data volumes are predictable.

We believe the current model is *not* well-suited for future sensor webs. Drawing from NASA vision documents for future sensor network requirements, we have derived the following requirements:

1. Direct IP-level connectivity is required between orbiting elements (instruments) and Internet-based hosts (investigators and data consumers).
2. Future orbiting constellations need to support elements in different orbits (e.g. LEO, MEO, GEO) with connectivity among them. In particular, some elements might not always, or ever, have direct connectivity with ground stations.
3. The orbiting elements will share a common communications infrastructure. That is, a single LEO satellite might function as a router for several GEO satellites.
4. Future instruments will be autonomous or semi-autonomous. This will allow the instruments to dynamically vary the frequency and resolution of their measurements in response to conditions.
5. Future instruments will produce extremely large raw data sets that may be passed between spacecraft and processed on orbit.
6. Efficiency of communication over the space-to-ground link is highly desirable. We assume that the data products generated on orbit, even after processing, will be large enough so as to consume all available downlink capacity.

As a result of these assumptions, we propose a completely IP-based system for the ground and space segments, with extensions to increase efficiency.

The rest of this paper is organized as follows. Section II describes the NGSI system architecture. Section III describes the NGSI system architecture in detail, and Section IV our standardization efforts. Section V describes our current system implementation and demonstration. Section VI contains concluding remarks and areas of future study.

II. NGSI SYSTEM ARCHITECTURE

Three main elements are required in order to connect space assets with Internet-based hosts: 1) security, 2) support for IP

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

mobility, and 3) resource reservation. This section begins with the overall system architecture then describes each of the elements in detail.

The NGSI 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. 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. Also, we consider an architecture where there may be multiple space hops between an instrument and the ground station responsible for communicating with it.

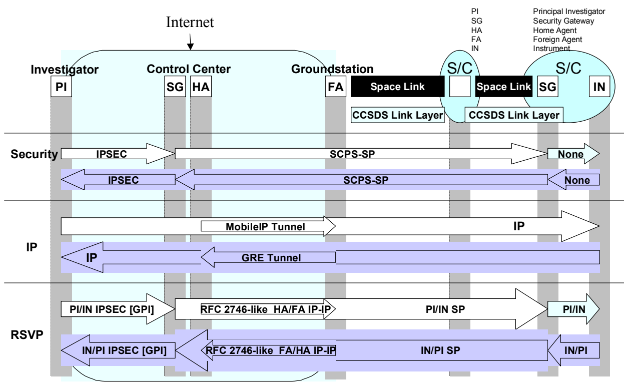

Figure 1: System architecture.

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. While we have not implemented security on board the destination spacecraft in our prototype, its addition would be a simple matter.

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).

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. No changes are required to routers 'in the middle' of the Internet. Even the requirement for an RSVP-capable connection among ground users can be relaxed if the connection from the groundstation to the control center is via a guaranteed bandwidth connection that matches the space-to-ground link bandwidth.

We also assume that each instrument is associated with a particular control center (though different instruments on the same spacecraft could have different control centers). This lets us know how to establish the required MobileIP tunnels and lets Internet-based hosts set up IPSEC associations with fixed endpoints.

A. *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 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 that translate 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. Each end user configures their system to user IPSEC tunnel mode when communicating with instruments, with the control center as the other end of the tunnel. The control center decrypts the IPSEC traffic and re-encrypts using SCPS-SP for transmission to security gateways on board the spacecraft. The security gateways on board the spacecraft can handle SCPS-SP communications with multiple distinct control centers, each with their own encryption.

The basic setup is shown in Figure 2, using a web server as the surrogate satellite instrument.

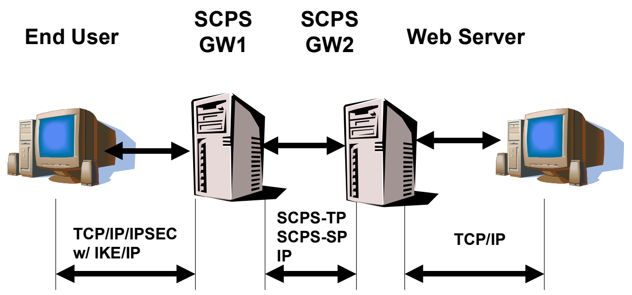

Figure 2: IPSEC/SCPS-SP Security with Trusted Gateways

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. MobileIP Extensions

As orbiting spacecraft communicate with different ground stations, their points of attachment to the terrestrial network change. If the groundstations with which they are communicating are at different locations in the Internet topology (if they are on different IP subnets, e.g.), then the spacecraft appear to Internet hosts like a mobile users, similar to someone with a laptop moving between home, work, and other locations.

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. 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.

In MobileIP, mobile units connect with 'Foreign Agents' that assist in delivering IP packets to the mobiles. Mobiles discover foreign agents by means of router solicitation and router advertisement messages. For spacecraft, these messages would have to be transmitted over the space link, consuming precious bandwidth and time during each contact.

Once it has contacted a foreign agent, the mobile node receives a care-of address 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.

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.

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 handoffs, 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 [4]. 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.

Cisco has recently developed a 'mobile router' product that can provide some of the same benefits as the extensions we have developed here. The mobile router technology is targeted primarily at the automobile and aircraft industries as a way of simplifying MobileIP for travelers. The mobile router is simply a mobile node with routing functionality.

Table 1 shows a comparison of the mobile router and NGSI MobileIP approaches. Because the mobile router is designed to work in 'standard' MobileIP environments where user mobility may not be predictable, it keeps the mobile—foreign agent signaling. This is exactly the overhead we were attempting to avoid in designing the NGSI MobileIP extensions. In addition, the mobile router establishes two IP-in-IP tunnels, the standard one from the home agent to the foreign agent, and a second IP-in-IP tunnel from the home agent to the mobile. This inner tunnel crosses the FA – mobile link, which in our case would be a ground-to-space link.

Table 1: Comparison of NGSI and Mobile Router Features

Feature	Cisco Mobile Router	NGSI MobileIP Extensions
Mobile – FA Signaling (Across the space-to-ground-link)	Yes – Router solicitation / Advertisement and Mobile Registration	NO – MobileIP tunnels configured from the ground before acquisition of signal (AOS)
Per-Packet Overhead between FA and Mobile (Across the space-to-ground link)	20 bytes per packet of IP-in-IP encapsulation	None
Operation in multi-hop constellation environments	No – Router solicitations and advertisements are link layer broadcasts	Yes – MobileIP tunnels initiated on the ground can direct traffic to arbitrary uplinks

We believe that these limit the utility of the mobile router technology to situations where there is only a single hop between ground and spacecraft and where efficiency is not an issue.

C. 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. If too many sources try to use the same resource (link) concurrently, they will congest it and the result will be dropped packets and data loss.

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.

RSVP is an end-to-end protocol for resource reservation, and hence touches all of the other work areas. Here we describe our modifications to the ISI reference implementation of RSVP [5] that allow it to function in the NGSI environment.

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[6] 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 [7]) 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 [8]) 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 [9] in FreeBSD and by the native 2.4-kernel traffic control [10] 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.

III. Standardization

The market for space communications equipment is relatively small, certainly when compared to the Internet. At the same time, NASA and other nations' space agencies are anxious to leverage the interoperability that comes from standardizing space technologies, whether they be communications protocols or hardware specifications.

We are standardizing mechanisms within the Consultative Committee for Space Data Systems (CCSDS)[11]. The CCSDS standardization process currently consists of two 'tracks' as shown in Figure 3: a 'standards' track and an 'experimental' track. The standards track is for technologies that have hard mission requirements, i.e. technologies that missions under development or in the planning stage are willing to commit to for operational use. The experimental track is for technologies that have prospective application but which no mission has yet committed to fly. At the current time a number of projects fall into this 'experimental' category, including most IP-based mission designs.

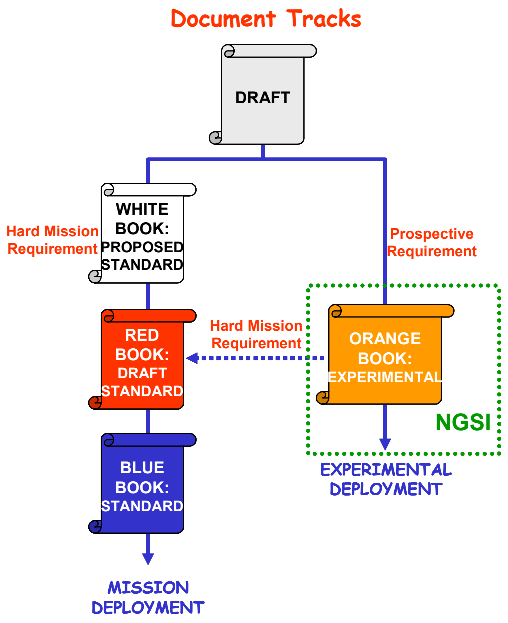

Figure 3: CCSDS Standardization Tracks

The goal of the experimental track is to provide a forum to flesh out how advanced technologies can be applied to missions and to develop concepts of operations. If the technology proves compelling, the proposed standards can be reviewed and quickly moved to and along the standards track.

The NGSI work described here is being standardized in the experimental track, under the Next Generation Space Internet working group within CCSDS.

IV. System Demonstration

To demonstrate the feasibility of the proposed approach, we first constructed a lab demonstration of the proposed technologies. This year we have moved the demonstration out of the lab so that the various elements, PI, control center, ground station, and satellite are geographically and topologically distributed. The emulated PI is located in Greenbelt, MD, the control center in Columbia, MD, the ground station in McLean, VA, and the simulated satellite is at the Jet Propulsion Laboratory in Pasadena, CA.

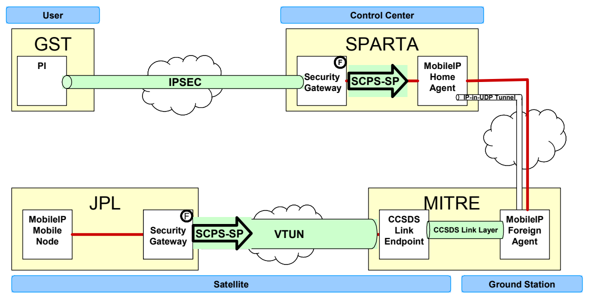

Figure 4: NGSI Distributed Demonstration

The various machines have all been 'hardened' with security measures, showing that the NGSI technologies can function in highly secure environments.

Splitting the simulated satellite between JPL (on the West coast) and MITRE (in Virginia) is a result of the CCSDS link layer implementation. The CCSDS link layer emulator allows the demonstration to use real space link telementry and telecommand protocols, but the implementation we have requires that the two ends of the link be on the same Ethernet segment. A future version of the demonstration will use VTUN's ability to tunnel Ethernet over IP to shift the space link endpoint to JPL.

V. Conclusions and Future Work

One of the biggest challenges to deploying the architecture described here is the inclusion of RSVP. RSVP is needed both to increase efficiency over the space link and to provide an automated resource arbitration mechanism that frees human operators from being involved in bandwidth provisioning. The difficulty is that ISPs do not generally support users sourceing RSVP requests through their networks. A solution to this for TCP-based traffic would be to acquire a guaranteed-bandwidth VPN service between ground stations and control centers that would pass the RSVP signaling. Provided that the VPN has capacity equal to or greater than the space link, there will be no loss in this portion of the network. The security gateway in the control center could then terminate the RSVP signaling. There is no danger of (unrecoverable) loss between the control center and the PI because the SCPS security gateway implementation also contains a transport-layer gateway. The security gateway actually terminates TCP connections coming from the satellite and starts separate ones with Internet-based hosts. Thus any losses incurred between the control center and Internet-based hosts would incur retransmissions from the control center, *not* from the instrument (and hence not across the space link).

The next step in advancing the Technical Readiness Level (TRL) of the NGSI technologies will be to field them on a suitable flight demonstration. The difficulty of porting the required protocols to a flight environment will range from trivial to moderate depending on the operating system of the target spacecraft.

This work is nearing the end of the standardization process in the CCSDS experimental track. The results, both specifications and code, will be maintained by CCSDS so that future missions will be able to review them and quickly convert them to full international standards.

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] MobileIP Implementation: Dynamics - HUT MobileIP, http://www.cs.hut.fi/Research/Dynamics/.

[5] Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," RFC 2205, September 1997, Proposed Standard.

[6] Terzis, A, Krawczyk, J., Wroclawski, J., and Zhang, L., "RSVP Operation Over IP Tunnels," RFC2746, January 2000.

[7] http://sites.inka.de/sites/bigred/devel/cipe.html

[8] 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.

[9] Kenjiro Cho, *The Design and Implementation of the ALTQ Traffic Management System*, Ph.D. dissertation, Keio University. January 2001.

[10] 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

[11] http://www.ccsds.org