Embedded Systems' Transport Protocol Choosing for Modelling over the SpaceWire Model

Ilya Korobkov
St-Petersburg State University of Aerospace Instrumentation
190000, St-Petersburg, Bolshaya Morskaya 67, Russia
Ilya.Korobkov@guap.ru

Abstract

Modelling becomes more and more important in the communication protocols development flow. It is a powerful tool in hands of developers. Modelling helps to find the weak spots of standards and fix them. Also it becomes possible to experiment by creating combinations of several specified standards' models that superpose in single executable system model.

This paper gives a review of existing streaming transport protocols, an overview of STP transport protocol modelling over the SpaceWire SystemC model activity and shares the given results.

Index Terms: Modelling, streaming protocols, SystemC, SpaceWire, STP

I. INTRODUCTION

Modelling is extensive used in the development process as a solution to perform detailed check of the specification and verification of the project to the stage of physical implementation of the final product. It allows spending less efforts, time and money [1].

In this paper we consider the collaboration of multiple applications over a SpaceWire Network. For this purpose we did review of existing streaming transport protocols, choose STP protocol and we implemented a layered model of STP protocol for testing of the specification.

This article gives an overview of activity and shares the results.

II. PROJECT OVERVIEW

There are many prospective applications to work over a SpaceWire Network interconnections operate with streaming data: data streams from high-rate sensors, ADCs, video streams input and output, etc. They have some general common features [2]:

- Information flow is generated by the information source continuously.
- Information flow is a sequence of information chunks of fixed and the same length.
- Information chunks are generated by the source, may be periodically with some time interval.

- Information chunks length and generation time interval could change, but being changed they keep it operating for a long period.
- Corrupted and lost in transmission information chunks are not expected to be repeated; in most cases – could not be repeated by the source.
- The receiver cannot stop generation of the information flow by the source instantaneously.
- Support of multiple coherent data streams

Some multimedia applications have the same features.

SpaceWire standard covers three (physical, data-link and network) of the seven layers of the OSI model and does not specify the transport layer [1]. Therefore to implement the simultaneous operation of multiple applications, pursuing different goals over the SpaceWire it is necessary to connect the transport layer with the previously developed SpaceWire model [1].

So there are a lot of transport layer protocols (e.g. streaming, real-time protocols and so on). The short review is presented below.

III. TRANSPORT PROTOCOLS REVEIW

A. Remote Memory Access Protocol

The aim of Remote Memory Access Protocol (RMAP) is to support reading from and writing to memory in a remote SpaceWire node. RMAP can be used to configure a SpaceWire Network, control SpaceWire nodes, and to transfer data to and from SpaceWire nodes [3].

RMAP is a connectionless protocol. RMAP has been designed to support a wide range of SpaceWire applications. Its primary purpose however is to configure a SpaceWire Network, to control SpaceWire units and to gather data and status information from those units [3].

RMAP may be used to configure SpaceWire routing switches, setting their operating parameters and routing table information. It may be used to monitor the status of those routing switches. RMAP may also be used to configure and read the status of nodes on the SpaceWire Network. For example, the operating data rate of a node may be set to 100 Mbits/s and the interface may be set to auto-start mode. For simple SpaceWire units without an embedded processor, RMAP may be used to set application configuration registers, to read status information and to read or write data into memory in the unit [3].

RMAP is efficient for system administration, for setting/checking device parameters, for casual data polling. In regular and intensive data transfer the RMAP request/reply scheme could be of excess in overheads both in communications loads and operation overheads, non-consistent in the stream delivery to its consumer and in pumping data out from sources with limited buffering [2].

B. CCSDS Packet Transfer Protocol

The aim of the CCSDS Space Packet Transfer Protocol (PTP) is to transfer CCSDS Packets across a SpaceWire Network. It does this by encapsulating the CCSDS Packet in a SpaceWire packet, transferring it across the SpaceWire Network and then extracting the CCSDS Packet at the target [4]. The PTP is a connectionless protocol [2].

PTP features and services [4]:

- PTP provides the capability to transfer CCSDS Space Packets between onboard users of a SpaceWire Network.
- The CCSDS space packets may be of variable length or fixed size at the discretion of the user and may be submitted for transmission at variable intervals. The composition of the CCSDS space packet is under the responsibility of the user application and is not checked by PTP.
- Unidirectional (one way) data transfer service.
- Asynchronous Service. There are no predefined timing rules for the transfer of service data units supplied by the service user. The user may request data transfer at any time it desires, but there may be restrictions imposed by the provider on the data generation rate.
- Unconfirmed Service: the sending user does not receive confirmation from the receiving end that data has been received.
- Incomplete Services. The services do not guarantee completeness, nor do they provide a retransmission mechanism.
- SDU format: the service does not check the format of the submitted CCSDS Space packet.
- Non sequence Preserving Service. The sequence of service data units supplied by the sending user may not be preserved through the underlying network.

C. Real-time Transport Protocol RTP

The Real-time Transport Protocol (RTP) is the Internet-standard protocol for the transport of real-time data, including audio and video. RTP is designed for end-to-end, real-time, transfer of stream data. RTP is regarded as the primary standard for audio/video transport in IP networks and is used with an associated profile and payload format [1]. RTP supports data transfer to multiple destinations through multicast [5].

RTP features [12]:

- RTP provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. But RTP itself does not provide any mechanism to ensure timely delivery. It needs support from lower layers that actually have control over resources in switches and routers.
- RTP doesn't assume anything about the underlying network, except that it provides framing. RTP is typically run on the top of UDP to make use of its multiplexing and checksum service, but efforts have been made to make RTP compatible with other transport protocols, such as ATM AAL5 and IPv6.
- RTP does not offer any form of reliability or flow/congestion control. It provides timestamps, sequence numbers as hooks for adding reliability and flow/congestion control, but how to implement is totally left to the application.
- RTP is a protocol framework that is deliberately not complete. It is open to new payload formats and new multimedia software. By adding new profile and payload format specifications, one can tailor RTP to new data formats and new applications.

D. Real-Time Streaming Protocol

The Real-Time Streaming Protocol (RTSP) is a client-server multimedia application protocol to enable controlled delivery of streamed multimedia data over IP network. It provides Video Cassette Recorder (VCR) style remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning. Sources of data include both live data feeds and stored clips [6].

RTSP features [12]:

- RTSP is an Application level protocol with syntax and operations similar to HTTP, but works for audio and video. It uses URLs like those in HTTP.
- An RTSP server needs to maintain states, using SETUP, TEARDOWN and other methods.
- RTSP messages are be carried out-of-band. The protocol for RTSP may be different from the data delivery protocol.
- Unlike HTTP, in RTSP both servers and clients can issue requests.
- RTSP is implemented on multiple operating system platforms, it allows interoperability between clients and servers from different manufacturers.

E. Resource Reservation Protocol

The Resource Reservation Protocol (RSVP) is a transport layer protocol that allows data receiver to request a special end-to-end quality of service for its data flows. Real-time applications use RSVP to reserve necessary resources at routers along the transmission paths so that the requested bandwidth can be available when the transmission actually takes place. RSVP is a main component of the future Integrated Services Internet which can provide both best-effort and real-time service [7].

RSVP features [12]:

- RSVP flows are simplex.
  RSVP distinguishes senders and receivers. Although in many cases, a host can act both as a sender and as a receiver, one RSVP reservation only reserves resources for data streams in one direction.
- RSVP supports both multicast and unicast, and adapts to changing memberships and routes.
  RSVP is designed for both multicast and unicast. Since the reservations are initiated by the receivers and the reservation states are soft, RSVP can easily handle changing memberships and routes. A host can send IGMP (Internet Group Management Protocol) messages to join a multicast group. Reservation merging enables RSVP to scale to large multicast groups without causing heavy overhead for the sender.
- RSVP is receiver-oriented and handles heterogeneous receivers.
  In heterogeneous multicast groups, receivers have different capacities and levels of QoS. The receiver oriented RSVP reservation requests facilitate the handling of heterogeous multicast groups. Receivers are responsible for choosing its own level of QoS, initiating the reservation and keeping it active as long as it wants. The senders divide traffic in several flows, each is a separate RSVP flow with different level of QoS. Each RSVP flow is

homogeneous and receivers can choose to join one or more flows. This approach makes it possible for heterogeneous receivers to request different QoS tailored to their particular capacities and requirements.

- RSVP has good compatibility.
  Efforts have been made to run RSVP over both IPv4 and IPv6. It provides opaque transport of traffic control and policy control messages in order to be more adaptive to new technologies. It also provides transparent operation through non-supporting regions.

F. Stream Control Transmission Protocol

The Stream Control Transmission Protocol (SCTP) is a reliable transport protocol that provides stable, ordered delivery of data between two endpoints (much like TCP) and also preserves data message boundaries (like UDP). However, unlike TCP and UDP, SCTP offers such advantages as multi-homing and multi-streaming capabilities, both of which increase availability [8]. SCTP is a message-oriented protocol.

Features of SCTP include:

- Multihoming support in which one or both endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between redundant network paths.
- Delivery of chunks within independent streams eliminate unnecessary head-of-line blocking, as opposed to TCP byte-stream delivery.
- Path selection and monitoring select a primary data transmission path and test the connectivity of the transmission path.
- Validation and acknowledgment mechanisms protect against flooding attacks and provide notification of duplicated or missing data chunks.
- Improved error detection suitable for Ethernet jumbo frames.

SCTP can multiplex multiple logical streams over a single session, supports both reliable and best-effort delivery modes, and provides fail-over across a group of redundant endpoints. SCTP has limitations reflecting its telecommunications focus, however [9]:

- SCTP streams cannot be created mid-session, only negotiated “en masse” at session initialization, limiting their utility for ephemeral or transaction-oriented activities.
- SCTP implements only one receive window per session rather than one per stream, so the receiver cannot accept data on one stream while applying back-pressure to others, further limiting their independence and usefulness to all but fixed-rate (e.g., telecom) applications.

G. Structured Stream Transport Protocol

The Structured Stream Transport Protocol (SSTP) is transport protocol designed to address the needs of modern applications that need to juggle many asynchronous communication activities in parallel, such as downloading different parts of a web page simultaneously and playing multiple audio and video streams at once [9].

Features of SSTP [13]:

- Multiplexes many application streams onto one network connection

- Gives streams hereditary structure: applications can spawn lightweight streams from existing ones
  - Efficient: no 3-way handshake on startup or TIME-WAIT on close
  - Supports request/response transactions without serializing onto one stream
  - General out-of-band signaling: control requests already in progress
- Both reliable and best-effort delivery in a semantically unified model
  - supports messages/datagrams of any size: no need to limit size of video frames, RPC responses, etc.
- Dynamic prioritization of application's streams
  - e.g., load visible parts of a web page first, change priorities when user scrolls
- Optional end-to-end cryptographic security comparable to SSL
- Peer-to-peer communication across NATs via hole punching
- Implemented as a library that can be linked directly into applications like SSL for easy deployment
- Doesn’t support multihoming/failover

H. Datagram Congestion Control Protocol

The DCCP (Datagram Congestion Control Protocol) is a transport protocol that provides bidirectional unicast connections of congestion-controlled unreliable datagrams. DCCP is suitable for applications that transfer fairly large amounts of data (e.g. streaming media, Multiplayer online games, Internet telephony), but can benefit from control over the tradeoff between timeliness and reliability [10].

DCCP provides the following features, among others [14]:

- An unreliable flow of datagrams, with acknowledgements.
- A reliable handshake for connection setup and teardown.
- Reliable negotiation of features.
- A choice of TCP-friendly congestion control mechanisms, including, initially,TCP-like congestion control (CCID 2) and TCP-Friendly Rate Control (CCID 3). CCID 2 uses a version of TCP’s congestion control mechanisms, and is appropriate for flows that want to quickly take advantage of available bandwidth, and can cope with quickly changing send rates; CCID 3 is appropriate for flows that require a steadier send rate.
- Options that tell the sender, with high reliability, which packets reached the receiver, and whether those packets were ECN marked, corrupted, or dropped in the receive buffer.
- Congestion control incorporating Explicit Congestion Notification (ECN) and the ECN Nonce.
- Mechanisms allowing a server to avoid holding any state for unacknowledged connection attempts or already-finished connections.
- Path MTU discovery.

DCCP implements congestion control for UDP-style best-effort communication, in the process incurring much of the same protocol complexity as TCP without providing reliable delivery, security, or other high-level features when they are desired [9]

I. Streaming Transport Protocol

The Streaming Transport Protocol (STP) is a new connection-oriented transport layer protocol [11]. STP is aimed for processing with stream-oriented information flow sources [2]. It performs set-connection control, initialization of connection parameters and data flow control [1].

Features of STP include:

- Connection-oriented protocol
  STP is developed as the connection-oriented transport protocol for regular data stream transfer from the source as the slave (with or without internal buffering). Interaction between the source and the recipient, the Transmitter and the Receiver is based on the establishment and maintenance a logical connection between these transport users. The master initiates establishment of the session, with setting logical connection – the transport channel, and setting its mode of operation and parameters. The session will be in operation until it will be terminated by the transport connection endpoint – the master [2].
- Multiple coherent data streams
  To support of multiple coherent data streams feature STP introduces a special field in data packets for coherence alignment of the incoming data streams in the receiver [2]. STP supports 2^{16}-1 [11] coherent data sources that can synchronous transmit data flows.
- Periodical continuous data transfer
  STP is an asymmetric protocol with the master and the slave(s). The master is the recipient and the slave is the source of the data stream to be transmitted. STP initialises data transmission ones for a long period of operation. After it the source (the slave) will send data PDUs one by one in accordance with the set for the transport connection parameters. The source governs itself the moments of data PDUs transmission (on data availability, on its generation time interval, etc.), without per PDU requests from the master (receiver) [2].
- Fixed length of transmitted data
  Transmitted by the transport connection PDUs have the fixed size that has been set in the transport connection establishment phase. Changes in any mode and parameters, the size included, in STP could be done only by termination the connection and establishment of a new transport connection with modifies parameters [2].
- In-order data delivery feature
  To control the in-order delivery of the STP PDUs and to reconstruct the initial PDUs order STP includes the ordinal number of the SDU in the STP data packet format [2].
- Data flow control
  STP Flow control mechanism uses the receiver crediting End-to-End Flow Control. The receiver (master) issues credits *n* in the number of packets it has buffer space for. The transmitter can send no more than the number of packets it has credits for. The packet size is defined in the transport connection parameters that are set in the Connection establishment phase and is known to both sides of the transport connection [2].

We have compared all considered protocols on the required features of the highly asynchronous applications, which operate with streaming data and work over the SpaceWire Network. The comparison is shown in the table I.

TABLE I
COMPARISON OF STREAMING REAL-TIME PROTOCOLS ON APPLICATION FEATURES

Name of protocol	Continuous data transfer	Periodical data transfer	Mutli-streaming	Flow Control	Connection-oriented protocol	Small overheads for data transfer
RMAP	-	-	-	-	-	-
CCSDS PTP	+	-	-	-	-	-
RTP	+	-	+	-	-	-
RTSP	+	+	+	+	-	-
RSVP	-	-	+	+	+	-
SCTP	-	-	+	+	+	-
SSTP	+	-	+	+	-	-
DCCP	+	-	+	+	+	-
STP	+	+	+	+	+	+

So STP covers all requirements of SpaceWire applications to data transfer over the basic SpaceWire Networks, which are not supported efficiently by others transport layer protocols [2]. Therefore we used STP to do the simultaneous operation of multiple space applications over the SpaceWire Network. We implemented the STP model for testing ability of the STP transport protocol work with SpaceWire protocol stack. But STP is still in the process of development, so the model's task is to help making important design decisions and to verify already specified mechanisms [1].

Whereas several applications should have possibility to transfer streaming data simultaneously by means of we developed management unit for transport protocol model. It is *STP_manager*. It allows to route data stream between applications and proper transport protocol model. Also for testing joint models work we developed model of Application level. It is *App_STP*. It performs an application role and it can simultaneously work with STP transport protocol by means of *STP_manager* [1].

The total structure of our model is shown at the fig. 1.

All created models are presented in details below.

IV. MODELS DESCRIPTION

A. STP SystemC Model

We developed the STP model in SystemC modelling language according to the STP specification. The structure of STP model is shown at the fig. 2 [1].

STP model consists of four main blocks: *data_up_wrapper*, *command_manager*, *checker*, *data_low_wrapper*. *Data_up_wrapper* performs communication functions between STP model and Application level. *Command_manager* is responsible for generating header and body of STP command. This block is divided into two parts: transmit (TX) and receive (RX). *Checker* is used to calculate and check CRC field in STP commands. In the CRC error case *Checker* should send corresponding notification to Application level. This module is divided into transmit and receive parts also. This model is event-oriented too and all blocks interact with each other via ports and

interfaces. STP model can work simultaneously at host and slave mode. Let's consider how it works [1].

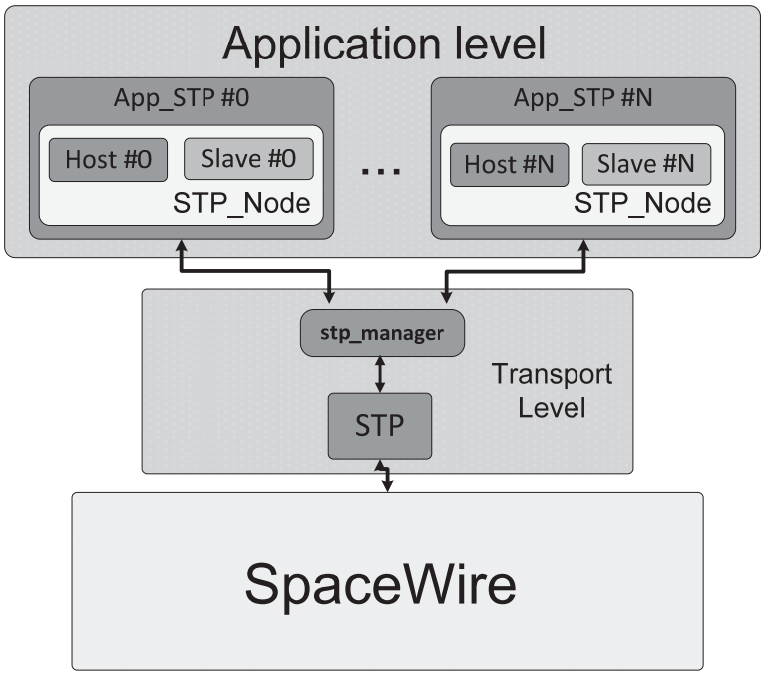

Fig. 1. Total SystemC model architectural diagram

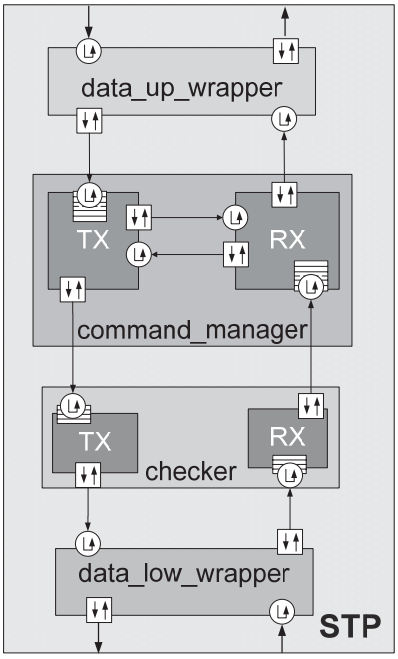

Fig. 2. STP model architectural diagram

STP model gets the open connection request from an application (host) [1]. Open and close connection mechanism between host and slave is shown at the fig. 3.

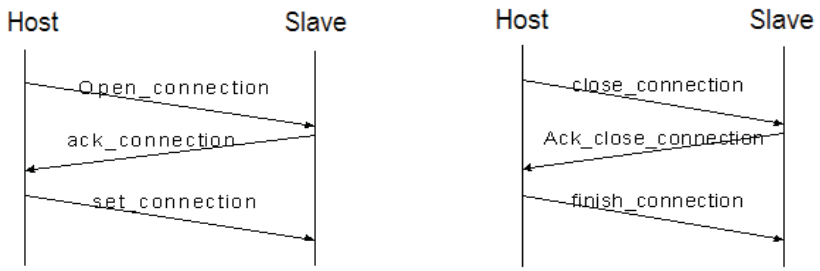

Fig. 3. Message sequence chart for open and close STP connection

Host generates an *open_connection* command depending on the received request. Then it calculates the CRC and sets it to command header and body. Then host sends this command to the SpaceWire Network symbol-by-symbol. When the command reaches the slave node, slave node assembles the command and then checks the CRC fields. If CRC field is invalid, the command would be ignored. Otherwise, command will be analyzed in details. If it is necessary, slave node will save all useful information about connection parameters from this command. After that it generates corresponding response command – *ack_connection*. And this command goes to host node via the SpaceWire Network. Host receives the response, analyzes it and generates another response command *set_connection*. If slave node receives correct response, connection is successfully set. Such mechanism allows decreasing of the reliability of connection establishment between applications [1].

After that slave node notifies the application about successful connection establishment. Application sends request with data for transmission to remote application (host). Slave generates data commands and then sends them with specified frequency. In addition slave will send data cargo according to the number of credits, which have been specified during the set connection transaction. If host buffers are full, host can notify slave to stop the data transfer. And later host can inform slave to start transfer data again. So STP protocol allows data flow controlling [1].

B. STP_manager

Since our task was the realization of several applications for the transport protocol, we developed the *STP_manager*. This unit route the data stream from the STP model to the set of applications. These applications are connected to manager via a number of ports. [1] The structure of STP manager is shown at the fig. 4.

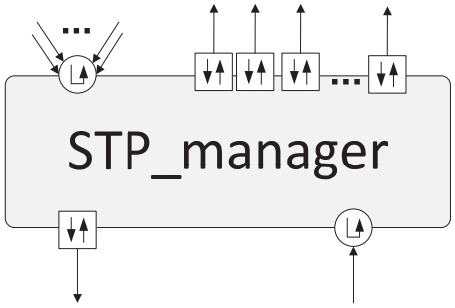

Fig. 4. STP manager SystemC model architectural diagram

Any received primitive should be analyzed. The manager should find the destination address of the particular application and store it in memory. Then a list of primitives should be sent to the application via appropriate port. STP has its own unique sequence of primitives, the manager has to control flow of primitives to detect the end of current packet. Several simultaneously operating applications can send the data to the manager that in turn sends them to the STP model [1].

V. Application level simulation

During the complex models step-by-step development it is necessary to have a testing system. It should help to find bugs in implementation. Besides it should do all necessary requirements for model maintenance – incoming data processing, logging and so on. And also the most important purpose of such a system is a conformance testing of the developed model [1].

Thereby we developed testing unit. It represents application layer model and it has the following functionality [1]:

- generation of different kinds of traffic;
- received data processing;
- response traffic generation;
- error and critical situations handling.

The Application level has module *App_STP*. It performs an application role and it can simultaneously work with *STP* by means of *STP_manager*.

VI. Conclusion

The result of our work is the review of existing streaming transport protocols, we have chosen STP for modeling and we have implemented the executable system model of STP protocol. The model satisfies to all specification requirements. It consists of all above mentioned models and units: *STP*, *STP_manager*, *App_STP*.

And the next result is that we have found 2 problems at the STP specification during simulation:

1. if multiple hosts set connection with the same value of connection ID with the same slave, then the slave will not recognize from which of hosts packets arrive.
2. if a slave does not handshake a connection establishment with host, because parameters of connection are not good for him or they are incorrect, then a host will infinite try to open connection with the parameters and a link will be littered by host's requests of open connection.

We have shared the found problems with STP workgroup and we have provided possible solutions of the problems. STP protocol is still in the process of development. Our executable system model allows to make the research of STP transport protocol over the SpaceWire Network and it helps to develop and improve the STP specification.

References

[1] V. Olenev, I. Korobkov, N. Martynov, A. Shadursky "RMAP and STP protocols modelling over the SpaceWire SystemC model", *Proceedings of 8th Seminar of Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program (p. 111),* Lappeenranta Univesity of Technology (LUT), Saint-Petersburg, 2010

[2] Y. Sheynin, E. Suvorova, F. Schutenko, V. Goussev, “Streaming Transport Protocols for SpaceWire Networks”, *International SpaceWire Conference 2010*, Saint-Petersburg University of Aerospace Instrumentation (SUAI), Saint-Petersburg, 2010
[3] ESA, standard ECSS-E-ST-50-52C , “SpaceWire - Remote memory access protocol”, *Publications Division ESTEC*, Noordwijk, The Netherlands, February 5, 2010
[4] ESA, standard ECSS-E-ST-50-53C , “SpaceWire – CCSDS packet transfer protocol”, *Publications Division ESTEC*, Noordwijk, The Netherlands, February 5, 2010
[5] H. Schulzrinne et al, “RTP: A transport protocol for real-time applications”, RFC 3550, July 2003
[6] H. Schulzrinne, A. Rao, R. Lanphier, “Real Time Streaming Protocol”, RFC 2543, *Internet Engineering Task Force*, March 1999
[7] Braden, R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC1633, ISI, MIT, PARC, June 1994
[8] R. Stewart et al, “Stream control transmission protocol”, RFC 2960, October 2000
[9] MIT, CSAIL, Parallel & Distributing Operating Systemc Group, SSTP specification, “Structured Stream Transport Preliminary Protocol Specification”, November 23, 2007
[10] E. Kohler, M. Handley, S. Floyd, “Datagram congestion control protocol (DCCP)”, RFC 4340, March 2006
[11] SUAI, STP specification, “Streaming Transport Protocol”, was presented at International SpaceWire Conference 2010, 2010
[12] Chunlei Liu, “Multimedia over IP: RSVP, RTP, RTCP, RTSP”, Handbook of emerging communications technologies, (pp. 29-46), RC Press, Inc. Boca Raton, FL, USA, 2000
[13] MIT, CSAIL, Parallel & Distributing Operating Systemc Group, “Structured Stream Transport”, http://pdos.csail.mit.edu/uia/sst/
[14] E. Kohler, S. Floyd, “Datagram Congestion Control Protocol (DCCP) Overview”, ICIR, July 9, 2003