STREAMING TRANSPORT PROTOCOL FOR SPACEWIRE NETWORKS

Session: SpaceWire Standardisation

Long Paper

Yuriy Sheynin, Elena Suvorova, Felix Schutenko

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: sheynin@aanet.ru, suvorova@aanet.ru

Vladimir Goussev

SIC "ELVEES", Moscow
vgoussev@elvees.com

ABSTRACT

The basic SpaceWire Protocol Stack, standard ECSS-E-50-12C, covers a set of layers, from PHY to Network level. The Transport level framework is under standardization. It specifies general structure of the Transport PDU, claims that a variety of Transport protocols can be specified and work simultaneously in a SpaceWire interconnection. Transport layer protocols are in development with a couple of them standardised: RMAP, implementing remote memory access paradigm, and the CCSDS packet transfer protocol.

For Transport protocols we discuss a variety of choices between Connectionless (CL) and Connection-oriented (CO) protocols. The RMAP protocol is considered as a case study of a CL protocol. It 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.

Many prospective applications to work over SpaceWire network interconnections operate with streaming data: data streams from high-rate sensors, ADCs, video streams input and output, etc. Some applications require support of multiple coherent data streams.

An outline for a new CO-type transport protocol – Streaming Transport Protocol (STP) is presented. We consider features that characterised streaming transport connection and consider the selected set of connection parameters, data packet parameters and additional data flow information.

The STP is implemented in our designs as a proprietary protocol. After its demonstration and trial it could be proposed for standardization by the SpaceWire community.

1. Introduction

According to the OSI 7-layer Reference Model, the transport layer is the lowest layer that operates on an end-to-end basis between two or more communicating hosts. The service of the transport layer use application entities. Communication between peer entities consists of an exchange of Protocol Data Units (PDUs). Application peers communicate using Application PDUs (APDUs),

while transport peers communicate using Transport PDUs (TPDUs). Interfaces between adjacent layers are provided with Service Access Points (SAP), by which the upper layer applies with its request to the lower one with data and control units that are the Service Data Units (SDU) of the layer in consideration. For the Transport layer it is the Transport Service Data Unit (TSDU) (used to be informally called a message), Figure 1.

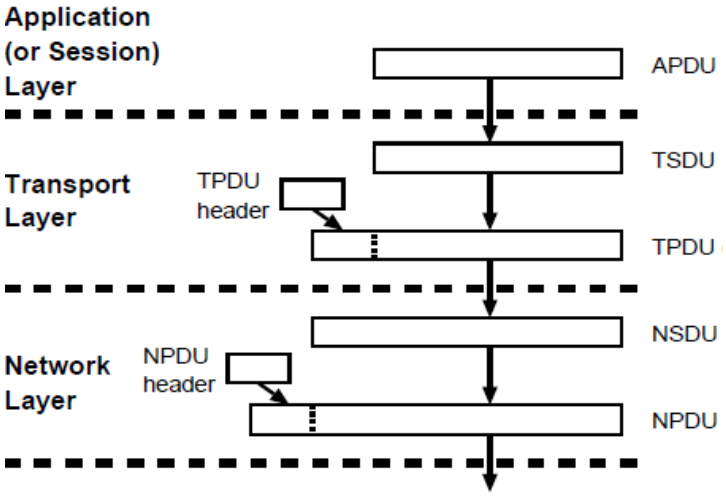

Figure 1.

The basic SpaceWire Protocol Stack, standard ECSS-E-50-12C, covers a set of layers, from PHY to Network level. The Transport level framework is officially added by the three new standards, [1, 2,3]. The ECSS-E-ST-50-51C introduces the Transport layer in the SpaceWire protocol stack and gives the SpaceWire transport protocol identification; the SpaceWire Transport layer is defined to be multiprotocol layer that supports simultaneous operation of multiple transport protocols running in a SpaceWire network. The next two standards specify the first two standardised transport protocols: the ECSS-E-ST-50-52C SpaceWire specifies the RMAP (Remote memory access protocol) transport protocol, the ECSS-E-ST-50-53C SpaceWire specifies the CCSDS packet transfer protocol transport protocol to run over SpaceWire interconnection.

Besides the standardised transport protocols the ECSS-E-ST-50-51C leaves a space for proprietary protocols also. With the 8-bit PIDs (Protocol Identifiers) coding the codes in the range 240 to 254 (0xF0 to 0xFE) could be used for particular non-standard protocols, one could develop and implement in its products. Reasons for developing new protocols could be specific requirements of particular projects and missions or lack of required for some transport services features and characteristics. In such cases a new protocol could be developed and implemented in addition to standardised ones as a proprietary transport protocol. With its implementation, which could be considered as proof of concept, and substantiation that similar features could not be covered with reasonable efficiency by the standardised protocols the new protocol cloud be standardised also by the SpaceWire WG.

Transport services can be divided into two types: connection-oriented and connectionless. A connection-oriented (CO) service provides for the establishment, maintenance, and termination of a logical connection between transport users. A transport service user generally performs three distinct phases of operation: connection establishment, data transfer, and connection termination. A connectionless (CL) service provides only one phase of operation: data transfer. A connectionless (CL) service provides no T-Connect and T-Disconnect primitives exchanged between a user sender and the transport sender, but gives only one phase of operation: data transfer.

As services are CO/CL specified, transport protocols could be classified CO or CL as well. The distinction depends on the establishment and maintenance of state information, a record of

characteristics and events related to the communication between the transport sender and receiver. A transport protocol is CO if state information is maintained between transport entities. If no state information is maintained at the transport sender and receiver, the protocol is CL. A CL protocol is based on individually self-contained PDUs often called datagrams that are exchanged independently. Each datagram contains all of the information that the receiving transport entity needs to interpret it.

The standardized Transport layers protocols RMAP and CCSDS PTP are connectionless protocols. More particular, the RMAP protocol could be classified as a transaction-oriented protocol, [4].Transaction-oriented protocols follows an asymmetrical model (i.e., client and server), short duration, low delay, few data TPDUs, and the need for no-duplicates service. Transaction-oriented protocols attempt to optimize the case where a user sender wishes to communicate a single APDU (called a request) to a user receiver, who then normally responds with a single APDU (called a response). Such a request/response pair is called a transaction. The RMAP protocol 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, [5].

Many prospective applications to work over SpaceWire network interconnections operate with streaming data: data streams from high-rate sensors, ADCs, video streams input and output, etc. They have different features for which CL-class, transaction-oriented protocols like RMAP is not efficient. It motivated us in development of a new CO-type transport protocol for SpaceWire networks – the Streaming Transport Protocol (STP).

2. Streaming Transport Protocol (STP) features

The Streaming Transport Protocol (STP) is aimed for processing with stream-oriented information flow sources. Such types of sources could be found in many space systems and spacecraft payloads, e.g. ADC with high sampling rates, ADC with preprocessing, video cameras, SAR sensors, high-rate instrument sensors, etc. They have some general common features:

- 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.

Additional feature one can find in many applications is that a receiver, e.g. the payload data processing unit, quite often deals with a set of similar sources that form a set of data streams. Moreover, some applications require support of multiple coherent data streams and this feature is to be supported by the transport service also.

Such a set of features justifies development of the tailored for it protocol, we call the Streaming Transport Protocol, STP. The STP provides for applications the connection-oriented transport service that fits the target information flow features. Continuous data generation and sending with stable features do not require a mode control per data chunk. The logic link between a source and

the receiver and its mode of operation could be set once for a long period, thus omitting overheads for per PDU transmission and delivery mode control. The connection-oriented service and CO transport protocol look quite natural here.

The STP is developed as the CO 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.

Like the RMAP, the 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. As distinct from RMAP, which requires a read command to be send to the slave to initialise transmission of data PDU from it, the 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).

The STP forms its PDU form the SDU that is supplied by the Application layer through the STP SAP. The SDU is enveloped and transmitted in the single STP PDU; STP does not use packing/unpacking of an SDU into fragments. An SDU is reformatted into a PDU that is transferred to the SpaceWire Network level for transmission in a single packet. 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 the STP could be done only by termination the connection and establishment of a new transport connection with modifies parameters.

With the STP idea of the host-receiver and slave-transmitter a STP transport connection is a point-to-point connection. A transport connection supports connection with a single slave and unicast service. However, it is considered that the master can have multiple transport connections with multiple slaves. Though they would be separate connections, they could be considered as correlated ones. From the master node (host) application point of view it could be a set of flowing together data streams, in some cases – coherent streams. Thus we have here an opposite to the multicast transmission case, so to say Inverse Multicast – a many-to-one transmission, [6]. To support of multiple coherent data streams feature the STP introduces a special field in data packets for coherence alignment of the incoming data streams in the receiver.

3. STP phases

3.1. Connection establishment

A Connection establishment launches the transport session between the host and a slave and builds the transport connection between them. The transport connection is asymmetric logical connection for transmission of packets from the source (slave) to the recipient (master). In the opposite direction the master sends control PDUs (commands) to the slave. On the transport connection (TC) establishment the TC parameters are set that would be in operation for the whole TC lifetime. The cost associated with the connection establishment would be amortized over a connection's lifetime.

Initiator of connection is the receiver (master). For the STP avoiding false connections is important, so 2-way or 3-way handshake mechanisms with explicit exchange of control TPDUs are needed. When the underlying network service provides a small degree of loss, a 2-way-handshake mechanism may be good enough to establish new connections without significant risk of false connections. The SpaceWire interconnections have rather low BER. However high robustness requirements for the space grade onboard interconnections shift the level of risk that is acceptable; it motivated us to move to the 3-way handshake. Three-phase protocol is used by the STP for connection establishment, Figure 2.

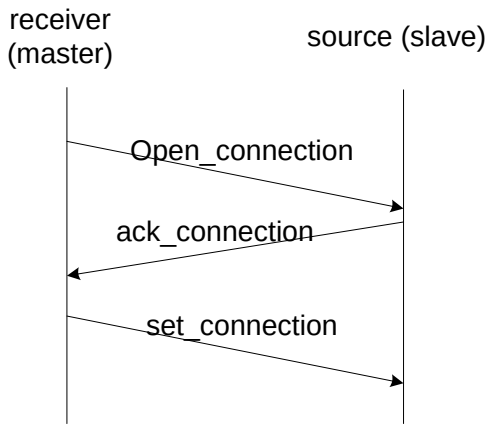

Figure 2.

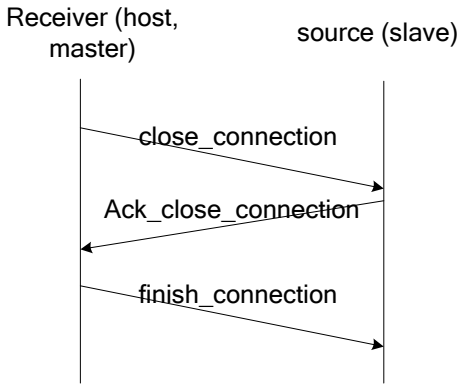

Figure 3.

The master sends an Open Connection to the slave, which responds with an Ack_Connection. The procedure is completed with a Set_Connection. No user data is carried on the connection establishment TPDUs. The 3-way-handshake is needed to prevent false connections that might result from delayed TPDUs.

3.2. Data transfer

Data PDUs could be transferred after the transport connection has been established. Permission for data transmission is sent by the receiver that should send a *Start_transfer* command to the receiver. A data PDU is generated by the slave and is sent to the master by the transport connection between the slave and the master.

Data transfer at the transport layer requires some flow control by which the recipient would not be flooded by the incoming PDSU flow and the underlying interconnection would not be blocked by packets that the recipient cannot intake. It can be done by preventing a transport sender from sending data for which there is no available buffer space at the transport receiver, or by preventing too much traffic in the underlying network. The SpaceWire interconnection doesn't have a standardised feature for preventing traffic overflow at the Network layer (though some implementations could have it). The STP Flow control mechanism uses the receiver crediting End-to-End Flow Control (E2E FC). 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 TC

The master can send the $n = 0$ that means the credit an unlimited number of packets. In fact, it is switching of the credit-based E2E FC. In many applications the receiver (the master) knows the PDU flow rate that could be generated by the source and is quite sure that could process the incoming data PDU flow in its regular mode of operation. The transmitter will send a packet after a packet by the TC without waiting for anything from the receiver to continue this process. The receiver shall receive and take them away from the TC. In case it cannot do it after some time, it

can stop the PDU flow by sending Stop control PDU to the transmitter. The transmitter should stop sending data PDU immediately as soon as it receives this control PDU. It is realised that a set of data PDUs could be left in the TC (in the underlying interconnection), which has been send in the interval between the moment when the receiver decided to stop transfer and the moment when the transmitter has received the Stop command. The receiver is obliged to take out all the left after its Stop command issue packets; the receiver can use them if it has buffer place for some of them or throw them away, these packets are considered to be lost.

The STP provides the ordered transport service. As an ordered service it preserves the user sender's submission order of data when delivering it to the user receiver (in-order delivery). It never occurs that a user sender submits two pieces of data, first A, then B, and A is delivered after B is delivered.

The STP considers ordering as providing a linear order of SDU generation events. For a SDU generation event in the source the strict order relation is defined to the events of all other PDUs' generations. The wall clock time of an event is not considered by the STP protocol FSM.

The STP understands that a basic SpaceWire interconnection does not guarantee in-order delivery of the sent packets. To control the in-order delivery of the STP PDUs and to reconstruct the initial PDUs order it includes the ordinal number of the SDU in the STP data packet format.

The STP has been developed in the general scope of the SpaceWire evolution, following its basics, compact implementation included. Thus the reordering of PDUs is limited by some number of *k* packets (a TC parameter). Outside the window of *k* packets a limited in-order delivery is provided, the reconstruction of the initial packet order at the receiver side is not guaranteed. Instead, the violating the order packets are discarded; their places in the ordered SDU sequence, which is returned to the upper layer, are filled by the default filler SDU (a TC parameter).

The STP provides a not guaranteed PDU delivery. An Error Control is provided, but corrupted or lost packets are not reconstructed or retransmitted. It corresponds to the nature of many steaming data applications. The receiver does not inform the transmitter of receive errors and do not request to resend PDUs. The source does not keep a sent by it PDU and do not retransmit it. However, the upper layer at the receiver side is informed about errors in the forwarded to it SDU flow.

3.3. Connection termination

Closing of the session and termination of the current transport connection with the slave is initiated by the master. For connection termination the STP uses a three-phased protocol that is illustrated by Figure 3. When the session and the TC are closed all its parameters are reset.

Two 2-way-handshakes are used, one for each direction of data flow. The master transport entity sends a *Close_connection* to its peer entity. The slave (transmitter) stops to send next data PDUs and then acknowledges the disconnect request by *Ack_close_connection*. The connection is terminated when all the incoming data flow is received by the master(receiver) – a sequence of PDUs in the TC finished by the *Ack_close_connection*. It ensures graceful TC termination, in normal operation no data in transit will be lost. Next the master confirms that it has received the acknowledgement and considers the TC is closed. After both sides have come to the "Closed" state for the TC, they become ready for establishment of another connection between them.

4. STP packet formats.

The STP uses three basic packet formats

- Packet-Command with parameters
- Packet-Command without parameters
- Data packet

4.1. STP Command packets

List of the STP commands (command PDUs) is presented by the Table 1.

Table 1. STP commands

Code	Command	With/without parameters
0000	open_connection	With parameters
0001	ack_connection	Without parameters
0010	set_connection	Without parameters
0011	close_connection	Without parameters
0100	ack_close_connection	Without parameters
0101	finish_connection	Without parameters
0110	start_transfer	Without parameters
0111	stop_transfer	Without parameters
1000	credit_transfer	Without parameters
1001 - 1111	Reserved	--

Commands without parameters are simple and compact (SpW header plus 5 bytes). In fact, commands with parameters are not used in the data transfer phase at all; all the possible for this phase commands are without parameters. The format of packet-commands without parameters is represented at the Figure 4.

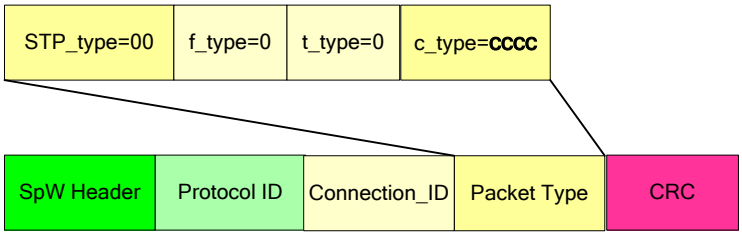

Figure 4

For the STP it is considered that the master can have multiple transport connections simultaneously; so the commands contain connection identification. Connection identifier is placed in Connection_ID field; its value could be from 0 to (2^{16} – 2), with the FFFF intended for special goals. Thus the STP master can have up to (2^{16} – 1) TCs with different slaves.

The Figure 5 represents general format of packet-command with parameters.

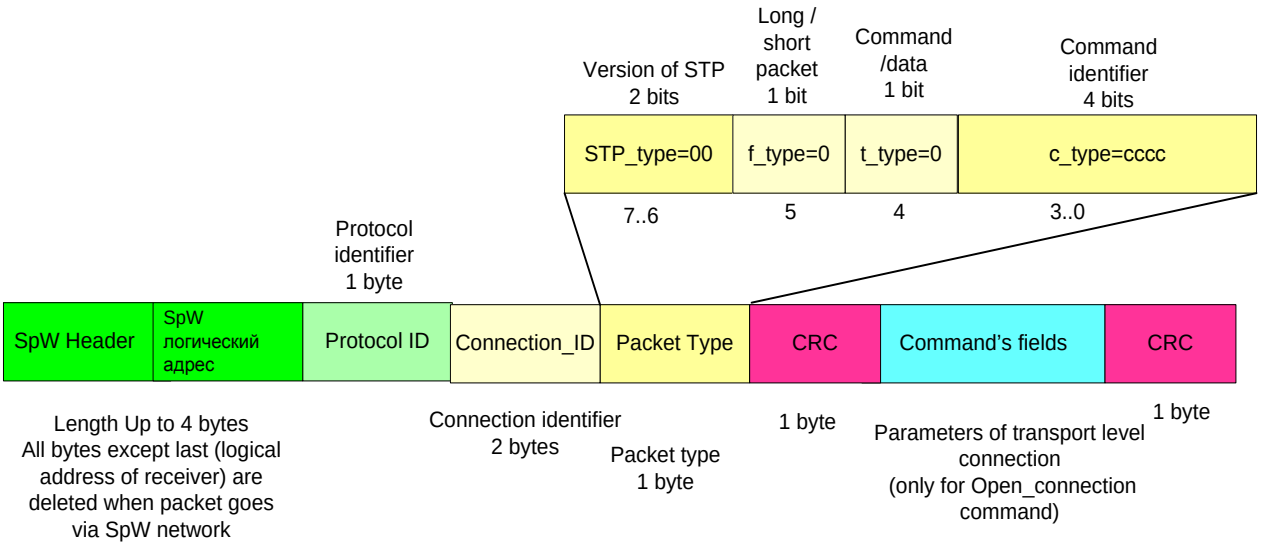

Figure 5

4.2. STP Data packets

Format of data packets represented at the Figure 6. The solid lines mark the boundaries of the 32-bit words (informative).

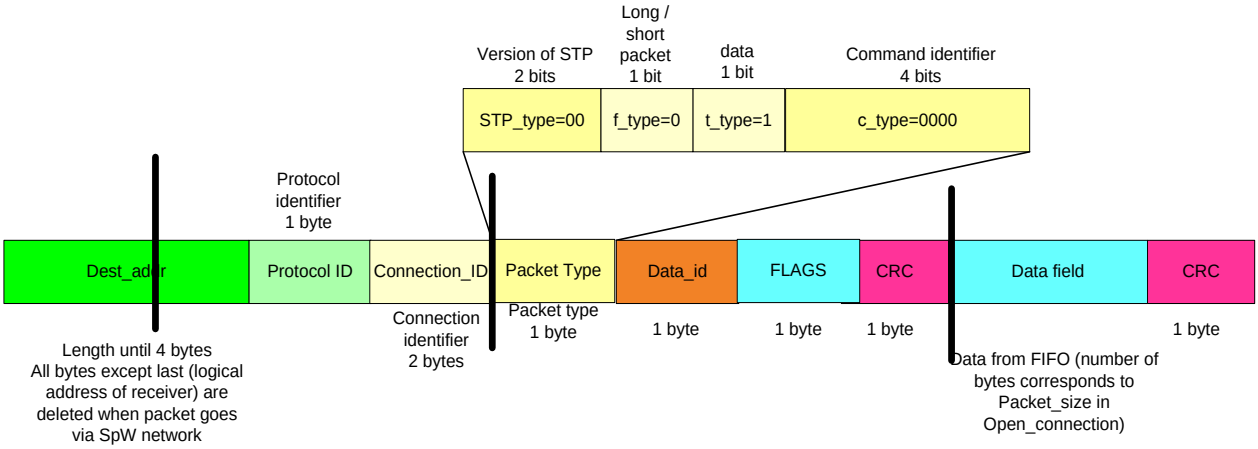

Figure 6

The *Data_ID* field (1 byte) is used for data identifier. Data identifier is generated by the source. Data identifier of every next data PDU is incremented by one; after 255 is the data identifier 0. If the transmitter received *Stop_transfer* command, after the following *Start_transfer* the Data_ID of first data packet will be 0.

The sFLAGs field is not specified in the STP; it could be used for flags that are generated and processed at the Application level. For instance, it could be used by the Application level protocols for its piggybacked control information transfer.

Conclusion

The Streaming Transport Protocols covers the streaming data transfer over the basic SpaceWire networks, which are not supported efficiently by the standardized Transport layer protocols.

The STP could be implemented over the basic SpaceWire interconnections with existing switching routers. Like the RMAP it could be implemented in nodes, e.g. processor-based nodes, in peripheral microcontroller nodes, in software. However, a hardware implementation of the STP is also quite feasible and can give better throughput and latency characteristics. Different STP implementation profiles could be specified also around its core functionality giving more cost-efficient specialization for particular applications with strict resource constraints. The STP is implemented in our designs as a proprietary protocol. After its demonstration and trial it could be proposed for standardization by the SpaceWire community.

Further developments could cover alternative, from the master to slaves, and bi-directional data streams transfer that are not covered by the current STP specification version. More interesting features, e.g. real-time coherent data streams stamping and correlation, could be built in STP with rely on the future SpaceWire 2.

References

1 ECSS-E-ST-50-51C SpaceWire protocol identification. ECSS Secretariat, ESA-ESTEC 2010, 5 February.

2 ECSS-E-ST-50-52C SpaceWire - Remote memory access protocol. ECSS Secretariat, ESA-ESTEC 2010, 5 February.

3 ECSS-E-ST-50-53C SpaceWire CCSDS packet transfer protocol. ECSS Secretariat, ESA-ESTEC 2010, 5 February.

4 Braden, R. Extending TCP for transactions – concepts. RFC 1379 (1992. Nov.).

5 Shutenko F., Suvorova E., Yablokov E., Gillet M. Low Power Protocols Development and Implementation. Proceedings of 6-th Seminar of Finish-Russian University Cooperation in Telecommunications, FRUCT 2009, pp.113-121

6 Gorbachev S., Sheynin Yu. Transport layer protocol for transputer networks. “Nauchnoe Priborostroenie” (Scientific Instrumentation), 1994, No 3-4. pp. 55-60 (in Russian).