Scheduling Mechanisms for SpaceWire Networks

Ilya Korobkov, Elena Podgornova, Dmitry Raszhivin, Valentin Olenev, Irina Lavrovskaya
Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
{ilya.korobkov, alena.podgornova, dmitry.raszhivin, valentin.olenev, irina.lavrovskaya}@guap.ru

***Abstract*—The paper describes two mechanisms which solve the problem of time multiplexing for SpaceWire networks. They are a transport-level scheduling mechanism, which is used in the STP-ISS protocol, and SpaceWire-D-SUAI, that is a protocol prototype. The current paper presents an overview of these mechanisms and compares them. In addition, we provide a short analysis of time-division multiplexing in the communication protocols.**

I. INTRODUCTION

Time multiplexing and scheduling are used in network technologies to obtain guaranteed transmission characteristics and to avoid conflicts with simultaneous network resource usage. However, analysis of the transport protocols presented in [1] showed that transport layer protocols officially released for the SpaceWire do not support these mechanisms. We conducted an analysis of different communication protocols that use different approaches to solve the time multiplexing problem. Basing on this research we focused on two mechanisms that are suitable for SpaceWire networks and compared them.

II. TDMA-BASED NETWORK PROTOCOLS

Time multiplexing is actively used in 2G and 3G mobile networks, as well as in some wireless personal networks, such as Bluetooth, ZigBee, Ubiquiti. However, SpaceWire developers are primarily interested in using time division multiplexing in on-board networks.

A. TTCAN

The time-triggered CAN (Controller Area Network) protocol [2] is a higher layer protocol on top of the CAN data link layer. TTCAN (Time Triggered CAN) provides mechanisms to schedule CAN messages in a time-triggered way as well as in an event-triggered way. It allows using CAN-based networks for closed-loop control. Also the real-time performance in CAN-based in-vehicle networks increases with the use of TTCAN.

The time-triggered control and thus synchronization of the involved control units in a network are done via a reference message. All participants of the TTCAN network identify the reference message by its identifier. As soon as the first bit of the frame (Start of Frame: SOF) is recognized, the local time unit is synchronized. The accuracy of the local time units depends only on the physical signal propagation of the bus line and it is thus neglectable. Individual TTCAN participants are configured to know when to send their frames after having received the reference frame. The time between two reference frames is called the basic cycle. Basic cycles are not always identical in order to be able to transmit messages at different periodic frequencies. The system matrix comprises several basic cycles and is repeated indefinitely until the vehicle network is turned off.

B. FlexRay

FlexRay [3] is a fast, deterministic and fault-tolerant bus system for automotive use, based on the experience of Daimler-Chrysler with the development of prototype applications and the developed by BMW byteflight communication system.

FlexRay works according to the TDMA principles. However, the fixed allocation of the bus bandwidth to the FlexRay components or messages by means of fixed time slots has the disadvantage that the bandwidth is not fully exploited. For this reason FlexRay subdivides the cycle into a static and a dynamic segment. The fixed time slots are situated in the static segment at the beginning of a bus cycle. In the dynamic segment the time slots are assigned dynamically.

In order to implement synchronous functions and optimize the bandwidth by means of small distances between two FlexRay messages, distributed components in the communication network require a common time base (global time). For clock synchronization, specific FlexRay messages tagged as synchronization messages are transmitted in the static segment of the cycle. With the aid of a special algorithm, the local clock-time of a component is corrected in such a way that all local clocks run synchronously to a global clock.

C. TTP/C

The TTP/C (Time Triggered Protocol class C) [4] frame consists of one byte header, up to 236 bytes of the payload and 3-byte CRC field. TTP/C implements the time-division multiplexing approach based on MEDL (Message Descriptor List), which shall be loaded into every node. The

ISSN 2305-7254

MEDL contains predefined static data to control when a message shall be sent on or received from the communication channels.

Each MELD string consists of the following main fields: the global time when the message shall be sent or received; the memory location of the message intended to be sent or received; the attributes field that includes the message type (input or output), the message length, etc.

To determine whether a node is operating correctly the membership service is used. Each bit in the membership field corresponds to a particular cluster node. When a node is allocated for data transmission at a particular time unit, all other nodes in the cluster analyze the input data so as to determine whether this node operates correctly.

D. TTEthernet

TTEthernet [5] is implemented on the basis of Ethernet. It provides time-triggered communications and global clock synchronization as well as a fault-tolerant operation mode. TTEthernet offers three types of traffic classes: Time-Triggered, Rate-Constrained and Best-Effort.

TTEthernet implements the TT class of QoS by the combination of resource reservation in space and time-division multiplexing. Each TTEthernet device in the network shall send TT frames only at predefined points of time to avoid collisions. On the other hand, frames, which are transmitted over different paths, can be sent to the network at the same time. To support this scheme TTEthernet implements clock synchronization mechanism.

In order to prevent error propagation from failed components the fault-tolerant TTEthernet network configuration deploys two independent channels for each connection. Safety-critical TTEthernet controllers shall be able to transmit and receive messages using two communication channels simultaneously.

In order to detect a failure of nodes within a cluster, TTEthernet provides membership service similar to TTP/C.

E. SpaceWire-D

SpaceWire-D protocol is a standard prototype that provides deterministic data delivery over SpaceWire networks [6]. It uses the RMAP protocol for transferring information over the SpaceWire network. The main feature of SpaceWire-D is a scheduling mechanism. SpaceWire-D divides the system operation time into a number of time-slots. A constant number of time-slots composes one epoch.

The receipt of a time-code by a node indicates the start of a time-slot. The time-slot number shall be the same as the time-value of the time-code that indicates the start of a time-slot. The end of a time-slot shall be normally indicated by the arrival of the next time-code.

SpaceWire-D uses the time-code watchdog timer that should be kept in each initiator node to check for the correct arrival of each time-code, early or late arrival of a time-code. In the event of an early or late time-code the initiator shall flag an error to the user application.

A schedule is defined by a schedule table for each initiator node. This table specifies in which time-slot the corresponding initiator is allowed to send RMAP commands. Each initiator node holds a copy of the schedule table. The initiator node may start the transmission of data to any target device during a time-slot in which it is scheduled to initiate RMAP transactions [6].

F. STP-ISS revision 2

STP-ISS [7, 8] is the transport layer protocol that describes informational and logic interaction between onboard devices, packets' formats and packet transmission rules for the SpaceWire network. STP-ISS provides connection-oriented and connectionless data transmission between the nodes of the network with priority, guaranteed delivery, best effort and scheduling quality of service types. The transmitter side has three logical buffers for each type of packet (control commands, urgent messages and common messages). The receiver side has two logical buffers: for connectionless and connection-oriented traffic. In addition, there is a flow control mechanism implemented for the connection-oriented transmission with the guaranteed quality of service. STP-ISS protocol provides the duplicate control commands detection in receiver and gives ability for data resending in case of error detection in the received data. Thus STP-ISS provides reliability, guaranteed services and scheduling. STP-ISS protocol is currently on modeling and testing stage.

III. Scheduling Mechanisms in the SpaceWire Onboard Networks

In this paper we present an analysis of the scheduling mechanism implemented in STP-ISS rev.2 and suggestions for scheduling mechanism for SpaceWire-D protocol. Both these algorithms are based on the SpaceWire time synchronization [9].

A. SpaceWire time synchronization

The scheduling approaches use SpaceWire time-codes broadcasting mechanism. These time-codes contain a six-bit value of system time. Each node and router has its internal six-bit time counter. There is one node or router in a network, which is set as a time-master. It is responsible for time distributing over a network. When the time-master receives a tick from a host-system, it should increment its time counter and send this new time value in a time-code. When a node or a router receives a time-code, it should update its internal time counter with the received time value. This new value should be one more than the time-counter's previous time value. If the received time-code value is equal

to internal counters value, then tick-out signal should not be emitted. When a router receives a time-code with a time value, which is one more than the internal counter's value it increments the counter value and emits a tick signal. This tick signal propagates to all the output ports of the router so that they emit the time-code. In turn, when a router receives a time-code with a time-code value that is equal to the internal counter value, it is ignored. This mechanism helps to prevent circular time-codes propagation over a network. This is the way the time-codes are used to synchronize all the network nodes with the time master's clock [9].

The detailed description of the STP-ISS rev.2 scheduling mechanism is presented in the next subsection.

B. STP-ISS scheduling mechanism

The first scheduling solution which we consider in this paper is the scheduling mechanism that is implemented in the new STP-ISS rev.2 transport protocol.

According to this scheduling mechanism, there is a single schedule for the whole SpaceWire network. It gives an opportunity for the node to send data only during particular time-slots. The schedule, time-slot duration (D_{TS}) and the number of time-slots in one epoch (N_{TS}) are set during the configuration phase and are stored in each node of the network. An epoch is a constant number of time-slots. For example, an epoch can consist of 10, 20, 64 or more time-slots, but it should contain at least 2 time-slots. The scheduling table defines one epoch. The epoch duration (D_E) is calculated in the following way:

$$D_E = D_{TS} \cdot N_{TS} \quad (1)$$

If the time-slot duration value changes, the epoch duration value D_E should be calculated and updated.

Each node has a time-slot timer (T_{TS}) which counts duration of the current time-slot for a node. Synchronization according to the scheduling mechanism is performed once in an epoch.

The time-slot timer T_{TS} should be set to the time-slot duration D_{TS}.

Each node is permitted to send packets during particular time-slots in accordance with the schedule. At the end of its time-slot, the node should stop the data transmission. However, the transmission stops only after the current packet is transmitted to the network. If any other node has data for transmission, but it is not scheduled for transmission during the current time-slot, then this node should wait for its time-slot.

STP-ISS protocol defines the time-code relevancy window, which is used to identify if the received time-code is relevant or not. This parameter defines a number of time-slots (K) in the end of the one epoch and in the beginning of the next epoch during which the received time-code is considered as relevant. The time-code relevancy window is shown in Fig. 1. Time-code relevancy window is a configuration parameter and should be set during the configuration phase. Its value should be even and could be defined individually for each node.

The time-slot timer expiration in the last time-slot of an epoch and reception of a relevant time-code indicate the beginning of a new epoch, in which the time-slot counter C_{TS} will count time-slots starting from zero. When a node gets a time-code, it does not analyze the time-code number. The beginning of a new epoch is associated with the fact of the time-code reception.

There are two possible synchronization cases, which can occur during network operation:

- the next time-code is received during first *K/2* time-slots of the epoch;
- the next time-code is received during last *K/2* time-slots of the epoch.

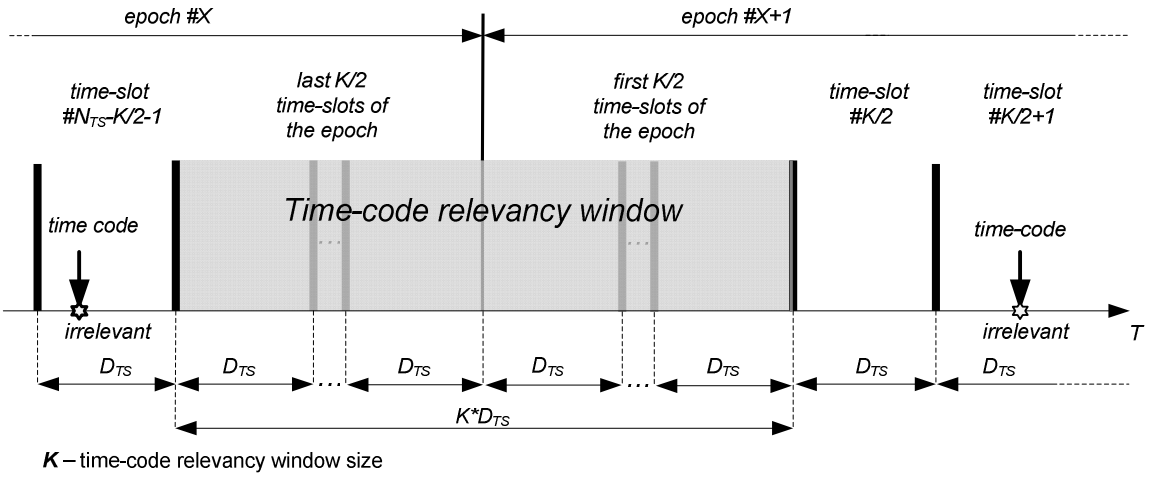

Fig. 1. Time-code relevancy window

Considering the node functionality, the abovementioned cases mean that the internal time counter and the time counter of the time-master are not synchronized. This means that the node should start the synchronization process.

Fig. 2 shows the case when a node started a new epoch and an expected time-code is received during the first *K/2* time-slots of the new epoch. In this case, the node should stop the time-slot timer T_{TS} and calculate a new value for the time-slot duration. The D_{TS_new} value is calculated according to the equation (2):

$$D_{TS\_new} = D_{TS} + \frac{\Delta t}{N_{TS}} \quad (2)$$

where Δt is the time value which passed from the beginning of the current epoch.

Subsequently, the node updates the epoch duration value according to the equation (3).

$$D_E / D_{TS\_new} \rightarrow N_{TS} \quad (3)$$

The newly calculated value is applied to the T_{TS} timer for all time-slots in the epoch.

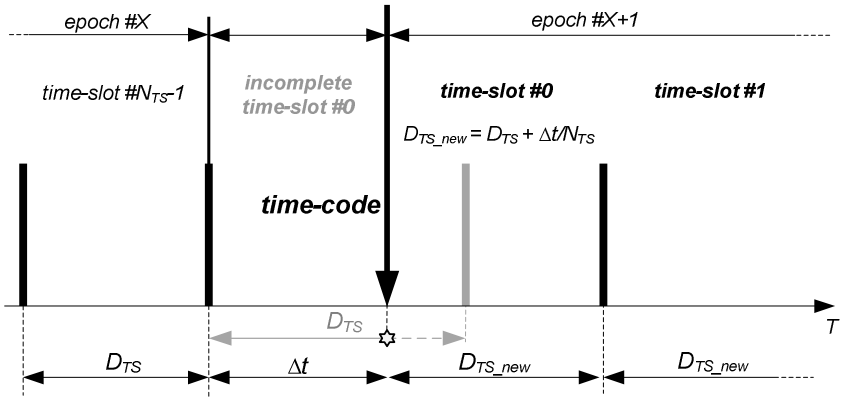

Fig. 2. Time-slot timer value correction (the time-code received during the first time-slot of the epoch

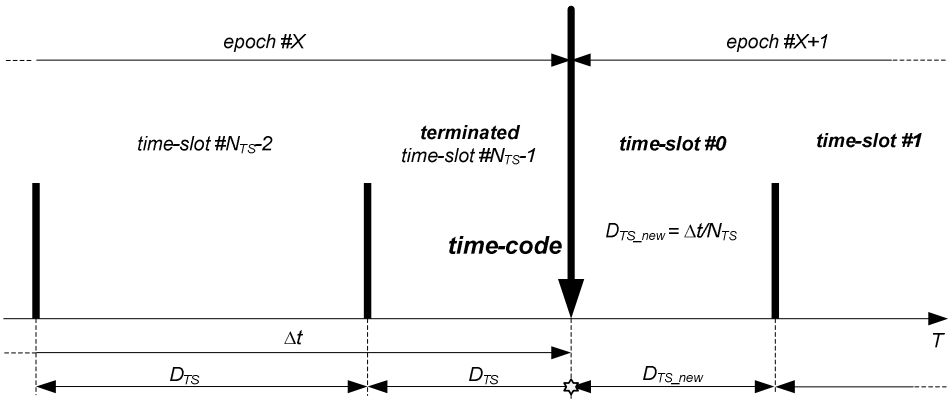

Fig. 3. Time-slot timer value correction (the time-code received the last time-slot of the epoch)

Let us consider the second case. The time-code is received during the last *K/2* time-slots of the epoch (see Fig. 3). In this case, the node should terminate the current epoch and calculate a new value of the time-slot duration. For this purpose, the node takes the current Δt value passed from the beginning of the epoch and calculates the new time-slot duration according to the equation (4):

$$D_{TS\_new} = \frac{\Delta t}{N_{TS}} \quad (4)$$

The newly calculated value is applied to the T_{TS} timer for all time-slots in the epoch.

If the T_{TS} timer of the last time-slot of an epoch expires simultaneously with the time-code reception, then there is no need to correct the time-slot duration value.

In order to detect significant time counters skew, the STP-ISS protocol should keep a history of received irrelevant time-codes. Reception of three irrelevant time-codes in three consecutive epochs means that the internal

time counter and the counter of the time master are significantly asynchronous. Reception of the third irrelevant time-code should determine the beginning of a new epoch. The node should stop the time-slot timer T_{TS} and wait for reception of the next time-code. In this new epoch the node should not send data until reception of the next time-code. After reception of the time-code the node should update the time-slot duration value D_{TS}. Then the node can continue data transmission according to the schedule. The new time-slot duration value should be calculated according to the equation (5):

$$D_{TS\_new} = \frac{\Delta t}{N_{TS\ c}} \quad (5)$$

where Δt is the time value, that passed from the moment of the third irrelevant time-code reception till the reception of the next time-code.

Once this mechanism was developed, we implemented a SystemC [10], [11] model of the SpaceWire network in order to check the correctness and efficiency of scheduling mechanism. An example of a network topology that we used for the scheduling simulation is shown in Fig. 4. Nodes of the SpaceWire network have numbers from 0 to 23; switches have numbers from 0 to 3. Nodes and switches are connected via SpaceWire channels. Node 0 is a SpaceWire time-master (TM). The detailed description of the implemented SystemC model and simulation parameters you can find in [12]. Although the scheduling mechanism has been slightly improved the simulation results are almost the same.

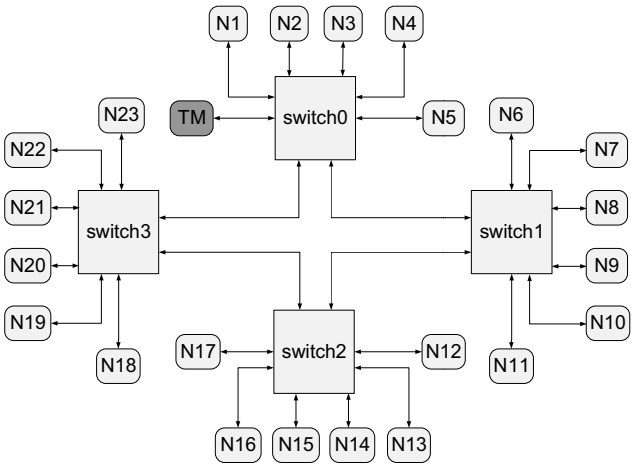

Fig. 4. SpaceWire network topology

Simulation results are provided in Fig. 5. The diagram clearly shows that in case of simulation without scheduling mechanism, the number of packets sent by nodes is non-uniformly distributed. The STP-ISS scheduling mechanism provides data transmission with uniform distribution. However, in comparison with simulation without scheduling mechanism, the mean-square deviation of packets number decreases from 53.25 to 5.51.

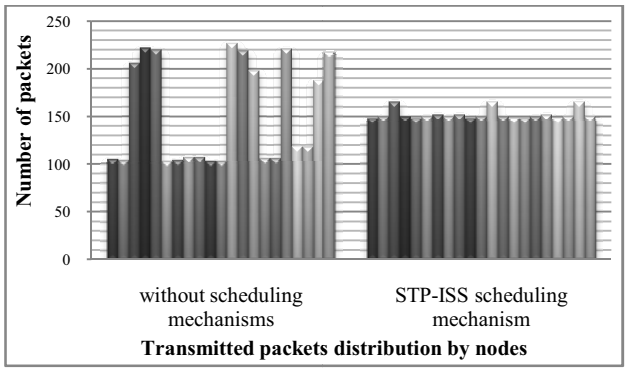

Fig. 5. Comparison of the simulation results for modeling without scheduling mechanism and STP-ISS scheduling mechanism modeling

Simulation results prove that the scheduling mechanism gives ability to uniformly distribute the network traffic among all nodes of the network.

C. Suggestions for SpaceWire-D scheduling mechanism

Another solution of the time multiplexing problem is a sublayer protocol between network and transport layers. It can be implemented in any of these two layers. This sublayer protocol is a set of services that works over SpaceWire and is compatible with any transport protocol. General mechanisms of this sublayer are presented below. We will call SpaceWire-D with those suggestions SpaceWire-D-SUAI.

1) Epoch ogranization: Each new time code indicates the beginning of a new time slot. Number of time slots in the epoch can vary from 2 to 64. The question is – how the node should determine the slot number basing on the time-code value.

Let a network member know a total number of time-slots in the epoch. It increments a slot's counter on receiving of the next valid time-code. This counter is reset to zero when the maximum value is reached. The disadvantage of this method is loss of transparency that is pawned in SpaceWire-D draft: the time-slot number is equal to the received time-code value.

Router synchronization problem arises if two routers were turned on at different time moments, as it is often done in real equipment. When the second router receives the first (his) time code, it would be treated as the start point for the first time-slot. At the same time this time-code would be subsequent for the first router that started before the second one, and the routers come out of synchronization. The problem is shown in Fig. 6: the router K1 is the time-master, it distributes time-codes, the epoch consists of 4 time-slots. The second router K2 turns on two slots later and receives a time-code with value “3”. However, it cannot unambiguously determine a place in the epoch of the current time-slot using the incoming time-code value, because time-code value is not equal to time-slot number.

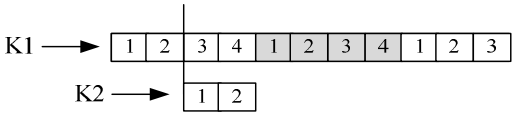

Fig. 6. Time-slot counter

Restriction on the multiplicity is another approach to epoch organization. Let us limit the value of time-slots in the epoch by the values that are the power of two: 2, 4, 8, 16, 32, and 64. All routers should know the value of power *n,* for two time-slots *n*=1, for 4 slots *n*=2, for 8 slots *n*=3 etc. All network participants can identically determine time-slot number while receiving new time-code if value *n* is pre-defined for them.

2) Port Guardian: A lot of network technologies, described above, suppose port guardian mechanism in order to protect the network from faulty nodes that try to transmit data at appropriate time-slots. Often such a "watch dog" is implemented as a separate device or chip in order to increase fault tolerance. Port guardian guarantees that the node would not transmit data during wrong time-slots and eliminates «babbling idiot» problem. Port guardian mechanism is supposed to be included to SpaceWire routers or nodes in order to improve network fault-tolerance of a deterministic SpaceWire network.

The Fig. 7 shows a SpaceWire network with time division multiplexing support. Network routers, marked as «Net guard», store scheduling table and permit data transfer for nodes only at proper time moments. The central part of the router is the standard SpaceWire router.

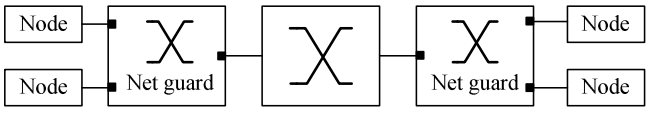

Fig. 7. Network with network guardians

The Fig. 8 shows «Net guard» router's port. This router has the ability to block transmission that violates the predetermined during configuration time scheduling table.

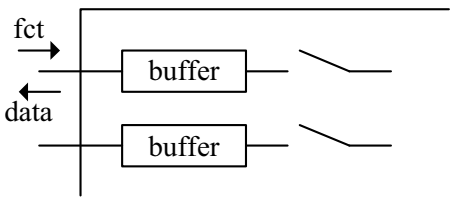

Fig. 8. Port guardian

3) Shadow master: An important task for the time division multiplexing protocol is to provide a fault-tolerance mechanism for the time markers distribution [13]. Loss of a single time-code with «master-slave» time synchronization leads to two time-slots loss; full time-master failure leads to the absolute network closedown.

Shadow master is one of possible solutions to increase time distribution fault-tolerance. This backup master can send time-codes in addition to the primary time master. It should be noted that this is a violation of the normative part of SpaceWire standard, that says that only one channel interface in the whole network should actuate an active tick signal. The Shadow master continuously checks the status of the primary time master by controlling the validity of incoming time code (Fig. 9). If the shadow master does not receive valid time code within a certain predefined time, it would start time codes distribution itself.

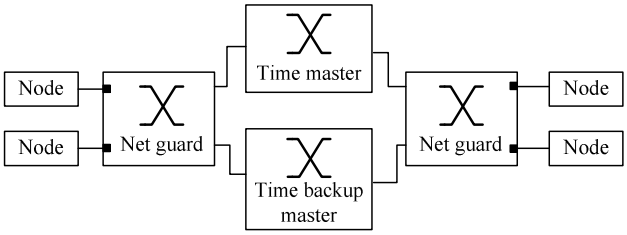

Fig. 9. Time backup master

4) Dynamic and static segments: It is necessary to integrate scheduled traffic and event-triggered traffic to effectively utilize psychical resources of the network. An epoch is divided into two parts for it; those parts are used for data transmission of scheduled or event-triggered traffic (Fig. 10).

Scheduled data transmission goes during static segment, flow control manages epoch division into slots. During dynamic segment all TDMA mechanisms are switched off, network runs at “classical” SpaceWire mode. The last slot in the epoch is designed to clean the routers' buffers of data, which has not been sent, to transmit an EEP or EOP symbol and to prepare the conversion to the static segment.

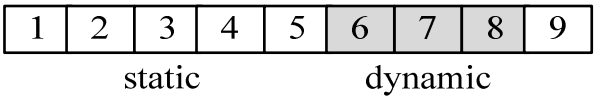

Fig. 10. Static and dynamic segments

IV. Comparison of the STP-ISS rev.2 Scheduling Mechanism and SpaceWire-D-SUAI Protocol

We presented two solutions of the time multiplexing problem. The first one is scheduling mechanism for the STP-ISS rev.2 protocol, the second one is SpaceWire-D-SUAI sublayer protocol. Position of these methods in conformance with the OSI [14] model you can see in Fig. 11.

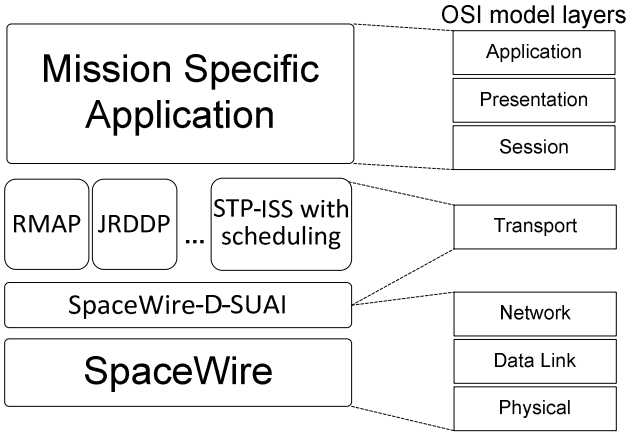

Fig. 11. Conformance of scheduling mechanisms with OSI model

Both these methods have their advantages and disadvantages. We summarized them in the Table I.

TABLE I. FEATURES OF STP-ISS SCHEDULING MECHANISM AND SPACEWIRE-D-SUAI

Scheduling / Features	STP-ISS scheduling mechanism	SpaceWire-D-SUAI
Epoch size	Not limited	Limited
Traffic scheduling	Schedules only STP-ISS protocol traffic	Schedules traffic for all transport protocols
"Babbling idiot" problem	No	Yes
Supports the "silent mode" during which all blocked packets will reach their destinations	Possible in case of appropriate schedule	Yes

V. CONCLUSION

In this paper we suggested and compared two distinct mechanisms solving the time multiplexing problem for SpaceWire networks. The STP-ISS scheduling mechanism does not limit the epoch size, so we are not restricted in creating the network schedule, but we can schedule transmission of only STP-ISS protocol data. SpaceWire-D-SUAI can schedule traffic of all transport protocols and it has no "babbling idiot" problem, but the epoch size is limited by the values that are the power of two: 2, 4, 8, 16, 32 and 64. Both these mechanisms can support the "silent mode" that allows the routers to clean buffers of data.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under the contract RFMEFI57814X0022.

REFERENCES

[1] V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov, "Analysis of the Transport Protocol Requirements for the SpaceWire On-board Networks of Spacecrafts", *Proceedings of 15th Seminar of Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program*; Saint-Petersburg: Saint-Petersburg University of Aerospace Instrumentation, 2014. pp. 65-71.

[2] Fuehrer, T., Mueller, B., Hartwich, F., and Hugel, R., "Time Triggered Communication on CAN (Time Triggered CAN-TTCAN)," SAE Technical Paper 2001-01-0073, 2001.

[3] A. Hanzlik, "A Case Study of Clock Synchronization in FlexRay", Research Report 31/2006 Technische Universitat Wien, Institut fur Technische Informatik, 2006.

[4] H. Kopetz, "Real-Time Systems. Design Principles for Distributed Embedded Applications", Kluwer Academic Publishers, Boston, 1997.

[5] A. Ademaj, H. Kopetz, P. Grillinger, et al., "Fault-Tolerant Time-Triggered Ethernet Configuration with Star Topology", available on-line.

[6] S. Parkes and A. Ferrer-Florit, *SpaceWire-D – Deterministic Control and Data Delivery Over SpaceWire Networks*, Draft B. April 2010.

[7] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, S. Kochura, S. Openko, D. Dymov "STP-ISS Transport Protocol Overview and Modeling", *Proceedings of 16th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program*; Oulu: University of Oulu, 2014. pp. 185-191.

[8] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov "STP-ISS Transport Protocol for Spacecraft On-board Networks", *Proceedings of 6th International SpaceWire Conference 2014 Program*; Greece, Athens, 2014. pp. 26-31.

[9] ESA (European Space Agency). Standard ECSS-E-50-12C, *Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization*. Noordwijk: ESA Publications Division ESTEC, 2008.

[10] D. Black, J. Donovan, B. Bunton, A. Keist, *SystemC: From the Ground Up*, Springer, 2010.

[11] Esperan, "Introduction to the SystemC Tutorial". Web: http://homepages.cae.wisc.edu/~ece734/SystemC/Esperan_SystemC_tutorial.pdf

[12] V. Olenev, E. Podgornova, I. Lavrovskaya, I. Korobkov, N. Matveeva, "Development of the transport layer scheduling mechanism for the onboard SpaceWire networks", *Proceedings of 16th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program;* Oulu: University of Oulu. pp. 164-170.

[13] W. Steiner, R. Maier, D. Jameux, A. Ademaj "Time-triggered services for SpaceWire", *Proceedings of the 2nd International SpaceWire Conference 2008 Program*; Japan, Nara, 2008. pp. 121-129.

[14] A.S. Tanenbaum, *Computer Networks*, Fifth Edition; Prentice Hall, 2011.