

# Computer-Aided Design System for On-board SpaceWire Networks Simulation and Design

Yuriy Sheynin, Valentin Olenev, Irina Lavrovskaya,  
Ilya Korobkov  
Saint-Petersburg State University of Aerospace  
Instrumentation  
Saint Petersburg, Russia  
sheynin@aanet.ru, {valentin.olenov, irina.lavrovskaya,  
ilya.korobkov}@guap.ru

Sergey Kochura, Vadim Shkolniy, Dmitry Dymov  
JSC "Academician M.F. Reshetnev" Information Satellite  
Systems"  
Zheleznogorsk, Russia  
{kochura, shkolniy}@iss-reshetnev.ru, dymovdv@mail.ru

**Abstract**—The paper provides an analysis of existing simulation tools for the on-board and local area networks. We overview the main abilities of the existing software and then propose the computer-aided design (CAD) system for SpaceWire onboard networks design and simulation. This CAD system will support the full on-board network design and simulation flow, which begins from the network topology automated generation and finishes with getting the simulation results and statistics.

## I. INTRODUCTION

Evolution of microelectronics has led to the growth of the on-board networks and systems sizes. Modern on-board networks consist of a huge number of computers, telemetry, radio-transmitting and data transmitting devices, scientific instruments and sensors and devices for electricity, heating control, orientation and stabilization of a spacecraft. Interconnection of these systems is done via the on-board network with numerous devices that work at different frequencies and data transmission speeds, transmit different types of data with different intensity. Each data flow has different Quality of Service (QoS) requirements. If there are some errors in a channel, or a device is corrupted, important scientific information could be lost. All these situations need to be simulated and tested on networking models before the assembling and launching a spacecraft. In the current paper, we will overview the existing simulation tools for the on-board and local-area networks and propose the architecture of a new tool for the SpaceWire on-board networks design and simulation.

## II. NETWORK SIMULATION TOOLS OVERVIEW

Network simulators allow researchers to test the scenarios that are difficult or expensive to imitate in real world. It is particularly useful to test new communication protocols or to change the existing protocols in a controlled and reproducible environment. Simulators can be used to design different network topologies using various types of nodes. There are different types of network simulators and they can be compared on the basis of the following features:

- range – from very simple to very complex,
- ability to specify nodes and links between those nodes and the traffic between the nodes,

- ability to specify everything about protocols used to handle traffic in a network,
- graphical user interface – allows users to easily visualize operation of their simulated environment,
- text-based applications – permit more advanced forms of customization;
- programming-oriented tools – providing a programming framework that customizes to create an application that simulates the networking environment to be tested [1].

There are different network simulators with different features. Some of them are commercial, which means that the source code of the software or the affiliated packages is not provided to users. All users have to pay to get a license to use this software or pay to order specific packages for their own specific usage requirements. On the other hand, open source network simulators and their interfaces are completely open for the developers.

Currently there is a number of tools and models that give an ability to simulate the operation of communication networks, but mostly these tools are intended for the Ethernet and Wi-Fi networks. Some of network simulators are overviewed in the current paper.

### A. GloMoSim

Global Mobile Information System Simulator (GloMoSim) is a scalable simulation environment for large wireless and wireline communication networks. GloMoSim uses a parallel discrete-event simulation capability. It simulates networks with up to thousand nodes linked by a heterogeneous communications capability that includes multicast, asymmetric communications using direct satellite broadcasts, multi-hop wireless communications using ad-hoc networking, and traditional Internet protocols.

The node aggregation technique is introduced into GloMoSim to give significant benefits to the simulation performance. Initializing each node as a separate entity inherently limits the scalability because the memory requirements increase dramatically for a model with large number of nodes. With node aggregation, a single entity can

simulate several network nodes in the system. Node aggregation technique implies that the number of nodes in the system can be increased while maintaining the same number of entities in the simulation. In GloMoSim, each entity represents a geographical area of the simulation. Hence, the network nodes which a particular entity represents are determined by the physical position of the nodes [2]. The GloMoSim user interface is shown in Fig. 1.

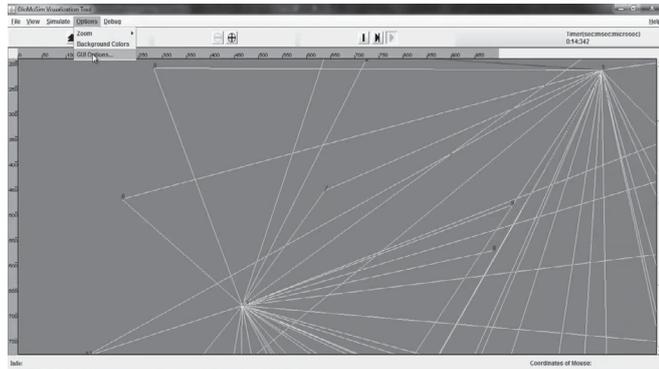


Fig.1. GloMoSim user interface

It is built using a layered approach that is similar to the OSI seven layers network architecture. Standard APIs are used between the different simulation layers, to allow integration of models developed at different layers by different people. GloMoSim has drifted away from creating each of the OSI layers as a separate entity to representing each node as a single entity, with each layer being represented only by standard APIs to initialize, finalize etc. This not only allows sharing of memory areas that all OSI layers need to access, but also allows for better performance, scalability and ease of programming use. GloMoSim is thus perceived to be modular, easy to use and flexible, besides maintaining a high degree of detail [1].

**B. QualNet**

QualNet is the commercial flavor of GloMoSim, and has additional implementations of layers/modules and features like GUI based analysis tools. It runs on all common platforms (Linux, Windows, Solaris, OS X) and is specialized in simulating all kind of wireless applications. It has a quite clear user interface while also offering an easy to use command line interface [3]. The QualNet user interface is shown in Fig. 2.

QualNet is composed of the following components:

1) *QualNet Architect*: a graphical scenario design and visualization tool. In Design mode, it is possible to set up terrain, network connections, subnets, mobility patterns of wireless users, and other functional parameters of network nodes. User can create network models by using intuitive, click and drag operations, customize the protocol stack of any of the nodes, specify the application layer traffic and services that run on the network. Visualize mode gives an ability to perform in-depth visualization and analysis of a network scenario designed in Design mode. As simulations are running, users can watch packets at various layers flow

through the network and view dynamic graphs of critical performance metrics. Real-time statistics are also an option, where you can view dynamic graphs while a network scenario simulation is running.

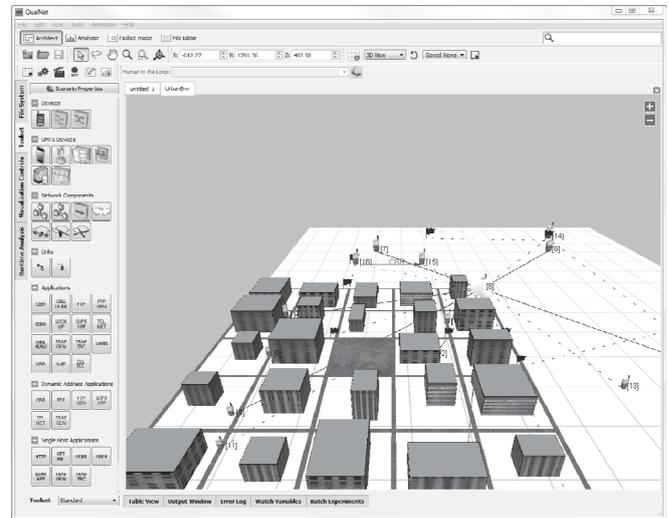


Fig.2. QualNet user interface

2) *QualNet Analyzer*: a statistical graphing tool that displays hundreds of metrics collected during simulation of a network scenario. You can choose to see pre-designed reports or customize graphs with their own statistics. Multi-experiment reports are also available. All statistics are exportable to spreadsheets in CSV format.

3) *QualNet Packet Tracer*: a graphical tool that provides a visual representation of packet trace files generated during the simulation of a network scenario. Trace files are text files in XML format that contain information about packets as they move up and down the protocol stack.

4) *QualNet File Editor*: a text editing tool.

5) *QualNet Command Line Interface*: Command line access to the simulator [4].

Therefore, QualNet has a lot of benefits and useful components, but unfortunately, it is a very expensive solution.

**C. OPNET**

OPNET is a registered commercial trademark and a name of product presented by OPNET Technologies incorporation. It became one of the most famous and popular commercial network simulators. Because of its use for a long time in the industry, it has become mature and has occupied a big market share. OPNET claims to be the fastest simulation engine among leading industry solutions. It has a wide variety of niche simulators for the wired/wireless areas. It also has many of wired/wireless protocol and vendor device models with source code, and allows object-oriented modeling of components. Modeling environment is an hierarchical one and has a slightly more complex method of definition of nodes as finite state machines. They also have an optional System-in-the-Loop to

interface simulations with live systems. The simulator is flexible and, therefore, allows integration with other libraries and simulators. With help from a rich suite of integrated, GUI-based debuggers and analyzers the setup, configuration can be done. The OPNET user interface is shown in Fig. 3.

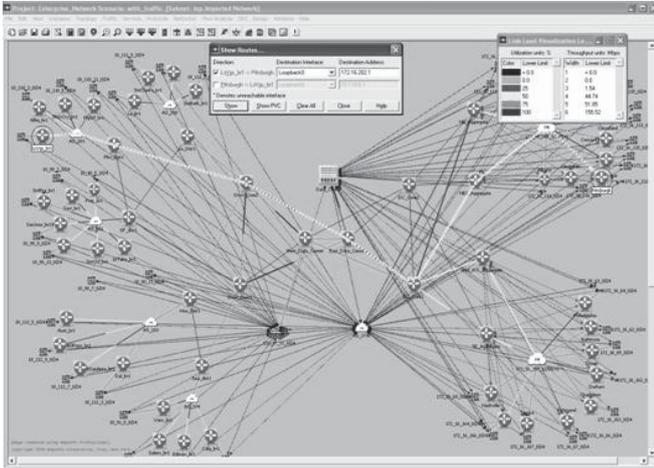


Fig.3. OPNET user interface

OPNET inherently has three main functions: modeling, simulating, and analysis. In modeling, it provides intuitive graphical environment to create all kinds of models of protocols for modeling. For simulating, it uses different advanced simulation technologies and can be used to address a wide range of studies. For analysis, the emulation results and data can be analyzed and displayed very easily for convince of its users. User friendly graphs, charts, statistics, and even animation can be generated by OPNET [1], [5].

D. NS-2

NS-2 is one of the most widely used network simulation tools in the research community and is available as freeware. Being an object-oriented discrete event simulator that follows the layered approach, NS-2 is accompanied by a rich set of protocols. Beside this NS-2 is also an emulator, and can talk to real networks. However, to its disadvantage, it has a large footprint, and is not very scalable. NS-2 also ranks low on the flexibility and ease of use fronts. In addition, the process of new protocols implementation is complex. It uses C++ and OTcl script language. The NS-2 user interface is shown in Fig. 4.

NS-2 separates control path implementations from the data path implementation. The scenes can be changed easily by programming in the OTcl script. When a user wants to make a new network object, he can either create the new object or assemble a compound object from the existing object library, and plumb the data path through the object. This plumbing makes NS-2 very powerful. Another feature of NS-2 is the event scheduler. In NS-2, the event scheduler keeps track of simulation time and release all the events in the event queue by invoking appropriate network components [6].

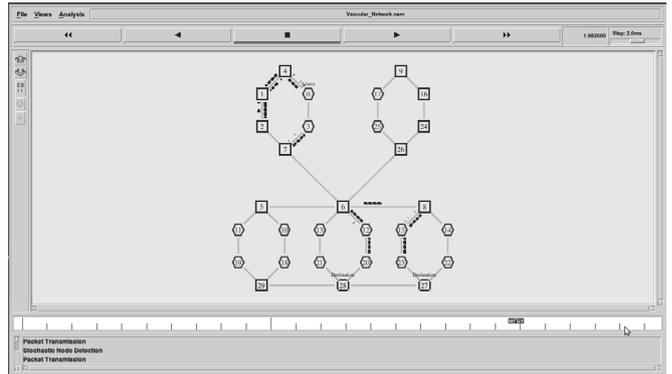


Fig.4. NS-2 user interface

E. NS-3

Similar to NS-2, NS-3 is also an open source discrete-event network simulator which targets primarily for research and educational use. NS-3 is licensed under the GNU, GPLv2 license, and is available for research and development. NS-3 is designed to replace the current popular NS-2. NS-3 is a new simulator and it is not backward-compatible with NS-2. The major differences lying between NS-3 and NS-2 is that it has different software core (C++ and Python), protocol entities are designed to be closer to real computers, lightweight virtual machines are used. NS-3 is developing a tracing and statistics gathering framework trying to enable customization of the output without rebuilding the simulation core [1], [7]. The NS-3 user interface is shown in Fig. 5.

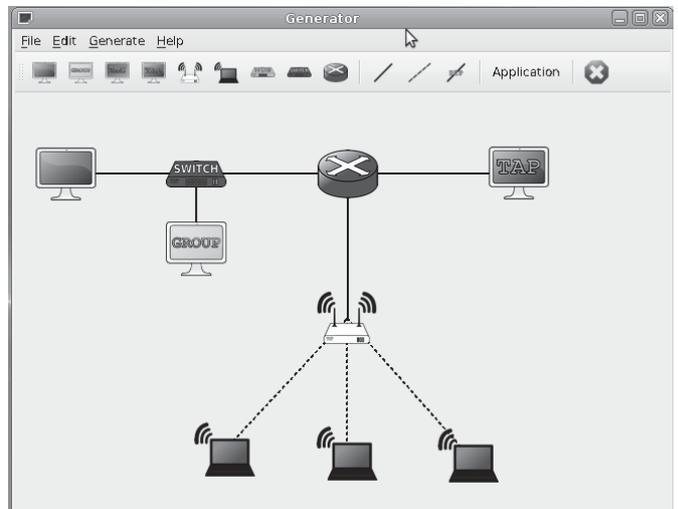


Fig.5. NS-3 user interface

F. OMNet++

OMNet++ is a C++-based discrete event simulator for modeling communication networks, multiprocessors and other distributed or parallel systems. OMNet++ is public-source, and can be used under the Academic Public License that makes the software free for non-profit use. Components of OMNET++ are defined by nested hierarchical modules in a simple text based language which is easy to learn, while being

very expressive. OMNet++ offers an easy to use GUI for graphical network editing, animation and configuring simulation runs. OMNet++ has a basic output analyzer, which can display collected statistics in graphical formats. However, not many OSI-related models are implemented. Nevertheless, its base infrastructure is very extensible, and it is easy to modify. This offsets the lack of implemented models to a certain extent. The models or modules of OMNet++ are assembled from reusable components as OMNet++ is designed to provide a component-based architecture. Modules are reusable and can be combined in various ways which is one of the main features of OMNet++. The OMNet++ user interface is shown in Fig. 6.

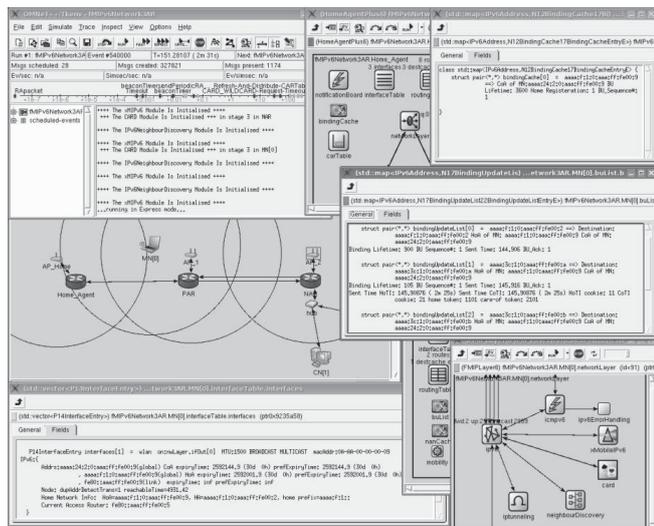


Fig.6. OMNet++ user interface

The OMNet++ approach significantly differs from that of NS-2, the most widely used network simulator in academic and research circles: while the NS-2 (and NS-3) project goal is to build a network simulator, OMNet++ aims at providing a rich simulation platform, and leaves creating simulation models to independent research groups [1], [8].

G. SSFNet

SSFNet (Scalable Simulation Framework) is defined as a “public-domain standard for discrete-event simulation of large, complex systems in Java and C++”. The SSFNet standard defines a minimalist API (which, however, was designed with parallel simulation in mind). The topology and configuration of SSFNet simulations are given in DML files. DML is a text-based format comparable to user friendly graphs: charts, statistics, and even animation can be generated by OPNET XML, but it has its own syntax. DML can be considered the SSFNet equivalent of NED, however it lacks expressing power and features to scale up to support large model frameworks built from reusable components. SSFNet also lacks OMNet++’s INI files, all parameters need to be given in the DML. The SSFNet user interface is shown in Fig. 7.

SSFNet has four implementations: DaSSF and CSSF in C++, and two Java implementations (Renesys Raceway and JSSF).

There were significantly more simulation models developed for the Java versions than for DaSSF.

H. REAL

REAL is a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks. It provides users with a way of specifying such networks and to simulate their behavior. It provides around 30 modules (written in C) that exactly emulate the actions of several well-known flow control protocols (such as TCP), and 5 research scheduling disciplines (such as Fair Queuing and Hierarchical Round Robin). The modular design of the system allows new modules to be added to the system with little effort. Source code is provided so that interested users can modify the simulator to their own purposes.

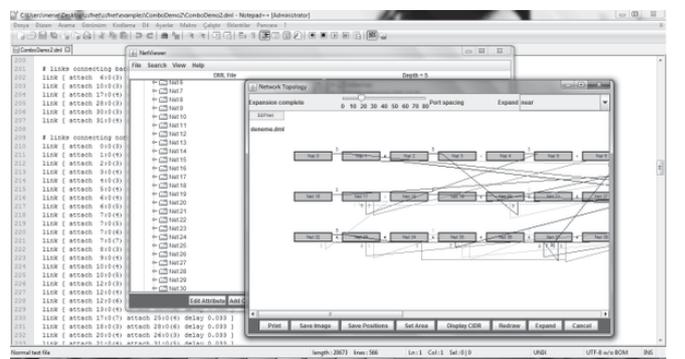


Fig.7. SSFNet user interface

The simulator takes as input a scenario, which is a description of a network topology, protocols, workload and control parameters. It produces as output statistics such as the number of packets sent by each source of data, the queueing delay at each queueing point, and the number of dropped and retransmitted packets. REAL includes a graphical user interface written in Java. The GUI allows users to quickly build simulation scenarios with a point-and-click interface [9]. However, REAL is pretty much old simulator, so mostly it is not used for the serious modern projects.

I. J-Sim

J-Sim (formerly known as JavaSim) is a component-based, compositional simulation environment, implemented in Java. J-Sim is similar to OMNet++ in that simulation models are hierarchical and built from self-contained components, but the approach of assembling components into models is more like NS-2: J-Sim is also a dual-language simulation environment, in which classes are written in Java, and glued together using Tcl (or Java). The use of Tcl in J-Sim has the same drawback as with NS-2: it makes implementing graphical editors impossible. In fact, J-Sim does provide a graphical editor (gEditor), but its native format is XML. Although gEditor can export Tcl scripts, developers recommend that XML files are directly loaded into the simulator, bypassing Tcl. This way, XML becomes the equivalent of OMNet++ NED. However, the problem with XML as native file format is that it is hard to read and write by humans. Simulation models are provided in

the Inet package, which contains IPv4, TCP, MPLS and other protocol models.

The fact that J-Sim is Java-based has some implications. On one hand, model development and debugging can be significantly faster than C++, due to existence of excellent Java development tools. However, simulation performance is significantly weaker than with C++, and, moreover, it is not possible to reuse existing real-life protocol implementations written in C as simulation models [8]. Example of J-Sim gedit application is shown in Fig. 8.

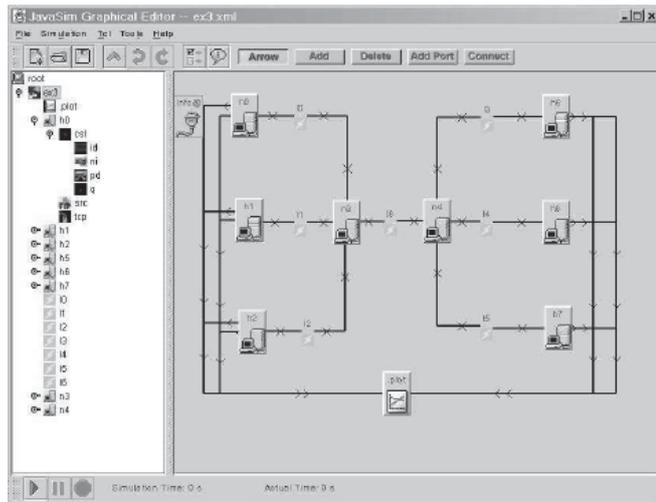


Fig.8. J-Sim gedit user interface

### III. SPACEWIRE NETWORK SIMULATION TOOLS

Most of considered network simulation tools are related to Ethernet and Wi-Fi networks simulation and unfortunately, none of these tools can model SpaceWire networks. However, this overview gives a good vision for the development of a tool for the SpaceWire simulation, because it shows the useful abilities and mechanisms, user-friendly GUIs and additional features for the network operation analysis. In addition, let us consider some tools that are specially developed for SpaceWire networks modeling. Most of them are based on the OPNET simulation framework. For these purposes OPNET was adapted for the SpaceWire and updated with a list of specific modules and network elements.

This way was chosen by Thales Alenia company, which implemented MOST (Modeling of SpaceWire Traffic) [10] for the European Space Agency. MOST based on the OPNET toolkit dedicated to network modeling. The MOST library contains SpaceWire nodes, routers and links which are selected by the user to build the SpaceWire network topology thanks to drag & drop actions. Configuration is done at the network level thanks to a set of attributes attached to each network component that can be tuned by the user. In MOST the Building Block (BB) concept is used to identify different communication layers in the node layout and it is in order to offer to user a flexible simulation tool. For example, the “SpaceWire standard BB” (or CODEC) is a SpaceWire node interface, which ensures node physical connection to the

SpaceWire network with the data transmission management. This “SpaceWire standard BB” is clearly separated from the protocol “RMAP BB”. As a result, one BB can be enhanced anytime without impacting others BB. Example of MOST capabilities is shown in Fig. 9.

OPNET offers a tool to analyse simulation output. It provides various ways to display each type of data. SpaceWire traffic analysis is done based on observation of statistic parameters such as: the end-to-end delay, the bottleneck observation, packet size, packet latency or jitter, number of sent and received packets, evaluation of the sustained bandwidth and also buffers occupation. Observables are selected by the user and it can be chosen to observe the data at the node level (internally) or at the network level [9].

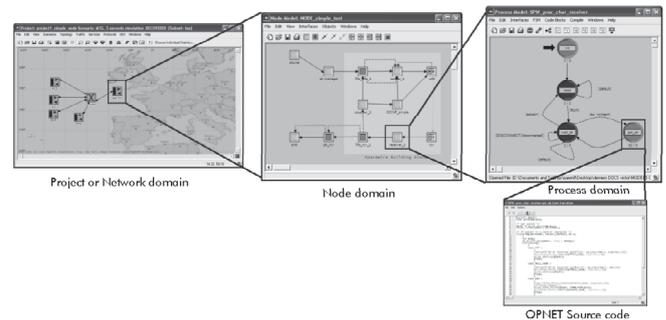


Fig.9. MOST user interface

Similar solution has been developed by Sandia National Laboratories (SNL) [11], but it gives less abilities than MOST, because it does not have an option to insert errors to the transmitted data, which is not good for testing and verification. The SNL team also used extension features within OPNET Modeler to create a set of general purpose modules representing different network elements or basic building blocks for SpaceWire networks simulation. The modules include models of SpaceWire nodes, routers, broadcast servers, and links. These modules can be arranged to represent networks during the design stage. Then, these networks can be analyzed for the desired behavior.

The second ability of the tool is an in-depth analysis of the accurate distribution of system time across the SpaceWire network. To accomplish this task, the SNL team developed a packet broadcast mechanism that would lay upon the standard SpaceWire protocol. A representative SpaceWire network was constructed within the OPNET Modeler simulation environment. Based on this network representation, several simulations were executed to study the behavior of the network with respect to packet transmission time, jitter, and the accuracy of distributed system time.

SpaceWire models provide a generalized tool for examining network behavior and different network designs. The user interface is very similar to MOST because of the OPNET base.

However, there is a tool that is not based on the OPNET. It is VisualSim SpaceWire modeling, developed by MIRABILIS Design Company [12]. VisualSim is intended for end-to-end system-level design. The graphical nature of the product and

the availability of the parameterized library, SmartBlocks, make the tool easy to use, adopt and learn. VisualSim combines DSP, Analog, Protocols and Digital Architecture in a single simulation model. SmartBlocks are graphical representations of hardware, software and networking components at queuing, performance, transaction and cycle-accurate levels of abstraction. VisualSim can be used for performance trade-offs using metrics such as bandwidth utilization, application response time and buffer requirements. Architecture analysis of arbitration algorithms, component sizing, software instruction optimization, hardware-software trade-offs and system coverage. Therefore, VisualSim gives an ability to test the real hardware SpaceWire devices, but it is not applicable for prototyping of real onboard networks on early stages of the project. The VisualSim user interface is shown in Fig. 10.

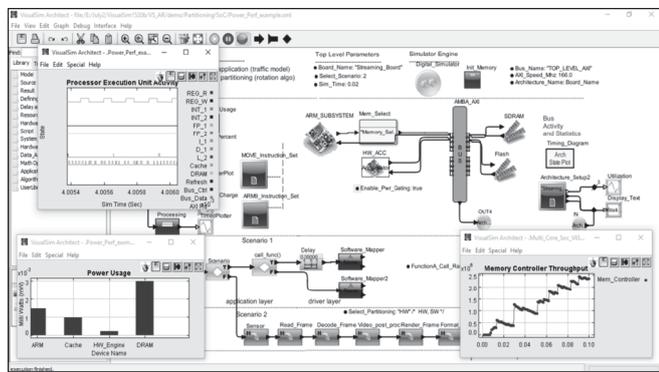


Fig.10. VisualSIM user interface

Consequently, there are only three tools that give an ability to build SpaceWire networks and to simulate their operation. Two of them are based on the OPNET and use its capabilities. Moreover, all these three tools are commercial and rather expensive.

The SUAI team has good experience in network modeling and building SpaceWire network models. In addition, we have a tool, which is able to simulate basic SpaceWire networks and transport-layer protocols – DCNSimulator [14].

The Digital Communication Network Simulator (DCNSimulator) is a tool for design, system-level simulation and analysis of networks. DCNSimulator is based on Qt and SystemC. It consists of the simulation engine and libraries of network components. The simulation engine is a general part that could work for simulation of any network. Libraries of network components are specific for particular network standards and could represent network components at various details levels – from general virtual components to cycle-accurate models of particular devices. Simulated device models are implemented in C++. Application software algorithms could run at end nodes thus generating realistic traffic for the simulated network. The simulator also allows users to design networks graphically in MS Visio. The DCNSimulator runs in Windows and does not require any other third party software for its operation. The DCNSimulator user interface is shown in Fig. 11.

The DCNSimulator with its library completely supports SpaceWire networks. It implements all levels of the SpaceWire standard (excluding signal and physical ones) and provides models of a terminal node, a routing switch and a channel (parameterized point-to-point link). Network models can be composed of these models of network elements. It also supports error imitation for channels and devices. With this tool, SpaceWire networks can be analyzed at the levels of bit flows, characters and packets. Therefore, one can analyze control codes and data packets propagation, channel workload and all errors occurred in channels. The simulator displays appropriate charts, statistics and information about every transferred code and packet.

We gathered the requirements for such kind of a simulation tool from industry. Analysis of these requirements showed that current version of the DCNSimulator also does not meet all industry requirements. Therefore, we started redesigning the existing tool in order to make it more useful and convenient in application.

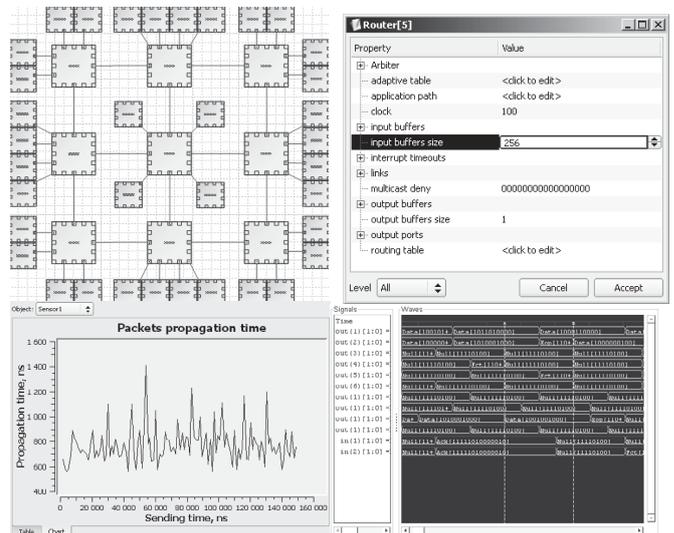


Fig.11. DCNSimulator user interface

#### IV. A NEW COMPUTER-AIDED DESIGN SYSTEM FOR THE SPACEWIRE NETWORKS SIMULATION

To be useful for the needs of the large industrial companies the following abilities should be provided by the network simulation tool:

- Simulation of network models that are implemented according to the SpaceWire and SpaceWire-RUS standards. That means, it has to have GigaSpaceWire extension;
- Simulation of the protocols of transport and application levels in nodes (at least RMAP [15] and STP-ISS [16]);
- Simulation of fault tolerance and redundancy;
- Simulation tool should be able to build a SpaceWire network consisting of network regions;

- Simulation tool should be able to save different designed objects (like nodes or switches) into a separate library that could be used for the other projects;
- Simulation tool should be able to build and model a network consisting of up to 1024 nodes.
- Simulation tool should give an ability to provide such simulation results like traffic between any two nodes, latencies, packet delivery time, transmission errors, channel bandwidth, etc.

Therefore, we made a decision to update the DCNSimulator with a number of new features that would give an ability to use the simulation tool during the whole process of the spacecraft onboard network design. This simulation tool would be a part of a complex computer-aided design system. This system will consist of four main components, and simulation tool would be only the one of them. Therefore, we propose a new computer-aided design system for SpaceWire on-board networks which will support full on-board network design and simulation flow,

which begins from the network topology automated generation and finishes with getting the simulation results, statistics and different diagrams. CAD will include all the improvements and extensions that are needed for the space industry. The proposed architecture of the CAD system is shown in Fig. 12. Network simulation tool is presented in the architecture diagram by the Component #4 and will be implemented in SystemC.

Each node of the simulated network will include an instance of the SpaceWire protocol and two transport layer protocols: RMAP and STP-ISS – two main transport protocols that are to be used in the Russian spacecraft. Switches in the network will operate according to the SpaceWire (ECSS-E-ST-50-12C [18]) and SpaceWire-RUS standards.

For some future implementations of the CAD system we plan to give a possibility to add other space protocols to the CAD system (e.g. SpaceFibre). For this purpose we will use particular library for each protocol implementation, so that in future we will be able to replace the SpaceWire for SpaceFibre.

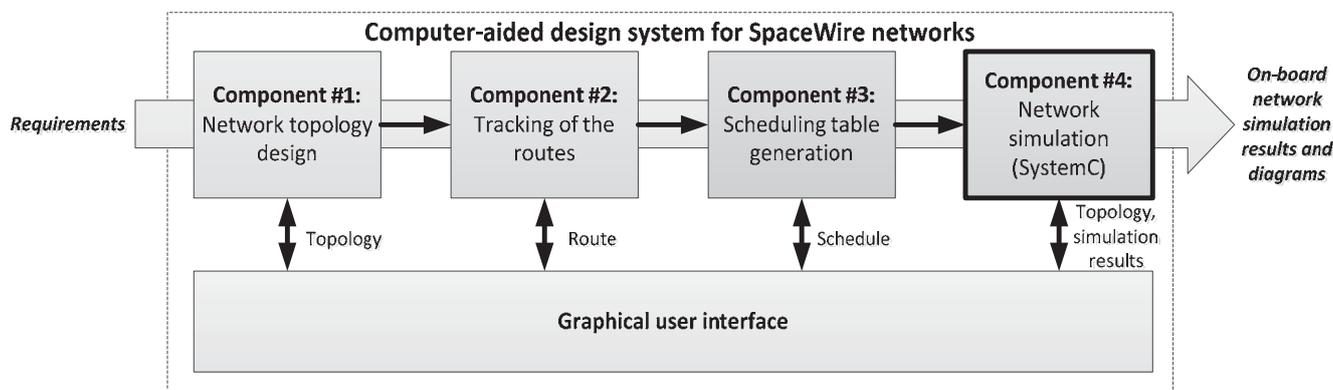


Fig. 12. A new computer-aided design system architecture

Visualization and graphical interface will be taken from the VIPE project [17]. Architecture of the proposed CAD system includes four main components:

- A component for on-board network topology design and estimation of its physical characteristics (based on Design Space Exploration technique [19], [20], [21]);
- A component for tracking of the routes for the data transmission in a network;
- A component for generation of the scheduling table for the STP-ISS transport protocol for the transmission of the data with Scheduled quality of service;
- A component for simulation of the network operation with all the data that component got from other 3 components and graphical user interface (majorly redesigned DCNSimulator).

Graphical user interface (GUI) will provide the visual network composition and management capabilities. It will allow designing SpaceWire network topology in visual interactive way from components. The component library is a

replenish set of network nodes and switches relevant to physical devices that are available for network building. It may also include flexible components if the developer wants to try various combinations of network equipment.

For all nodes, switches and channels the GUI will provide the configuration interface to set up their parameters, configure transport protocols and application-level traffic generators.

The designed network will be exported to the intermediate representation format to be used in other CAD tools for simulator, routes tracking, scheduling calculation and other tools. GUI will be also able to show simulation results both on the network structure and on charts and diagrams.

All these difficult tasks are planned to be solved in the new project. Implementation of such kind of a design and simulation toolset will give an ability for spacecraft designers to design the onboard SpaceWire network with all its technical characteristics and features, distribute the data flows and simulate it taking into account real latencies, channel errors, etc. In addition, the toolset will present simulation results in convenient and unambiguous form.

VII. CONCLUSION

In current paper, we overviewed existing simulation tools for on-board and local-area networks and proposed the solution of a new computer-aided design system for the SpaceWire onboard networks design and simulation. This software will solve important tasks, which spacecraft developers face with during implementation of satellites and other space vehicles. The CAD system will begin its workflow from the on-board network topology design and estimation of its physical characteristics. Then it will give an ability to track the routes for the data transmission in a network and generate the scheduling table for the STP-ISS transport protocol for data transmission with Guaranteed QoS. After the network design is finished – the last stage would be simulation of the network operation with real characteristics. Graphical user interface will provide possibilities to draw the network topology and set different parameters of nodes, switches and channels. The proposed software system would be a good assistant during the spacecraft design, implementation and testing.

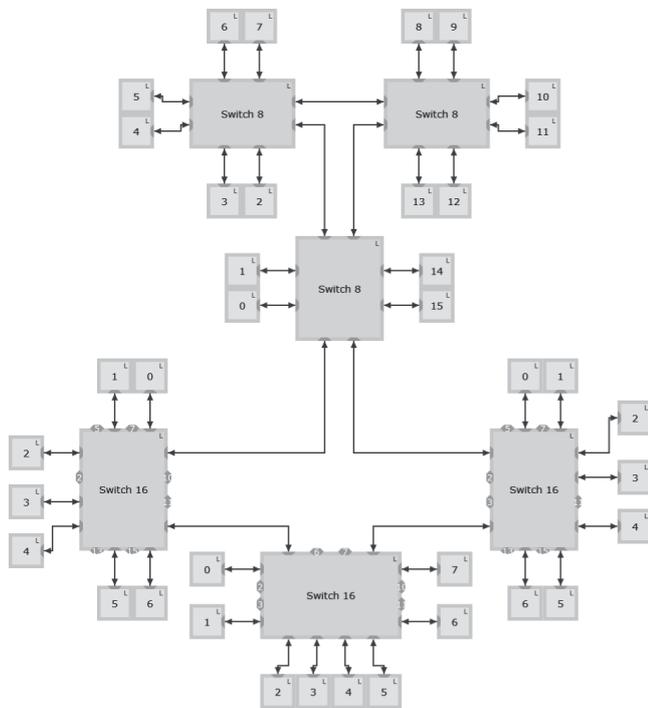


Fig. 13. SpaceWire network scheme in GUI

ACKNOWLEDGEMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under the contract RFMEFI57816X0214.

REFERENCES

- [1] Mrs. Saba Siraj, Mr. Ajay Kumar Gupta, Mrs Rinku-Badgujar. "Network Simulation Tools Survey", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 1, Issue 4, Wagholi, 2012, pp. 201-210
- [2] J. Nuevo, "A Comprehensive GloMoSim Tutorial", INRS - Universite du Quebec, 2004, 34 p.
- [3] T. Doerffel, "Simulation of wireless ad-hoc sensor networks with QualNet", Advanced Seminar on Embedded Systems, Technische Universitat Chemnitz, 2009, 16 p.
- [4] SCALABLE Network Technologies, "Make Networks Work. Network modeling software for Development and Analysis", QualNet Datasheet, 2014, 4 p.
- [5] Hou Jianru, Chen Xiaomin, Sun Huixian, "An OPNET Model of SpaceWire and Validation", Proceedings of the 2012 International Conference on Electronics, Communications and Control, Zhoushan, 2012. pp. 792-795
- [6] T. Issariyakul, E. Hossain, "Introduction to Network Simulator NS2", Springer Science+Business Media, 2012, 512 p.
- [7] NS-3 Manual, "NS-3 Network Simulator", 2017, 165 p.
- [8] A. Varga, R. Hornig "An overview of the OMNeT++ simulation environment", Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, Marseille, France, 2008.
- [9] S. Keshav, "REAL: A Network Simulator", University of California, Berkeley, 1988, 16 p.
- [10] B. Dellandrea, B. Gouin, S. Parkes, D. Jameux, "MOST: Modeling of SpaceWire & SpaceFiber Traffic-Applications and Operations: On-Board Segment", Proceedings of the DASIA 2014 conference, Warsaw, 2014.
- [11] Thales Alenia Space, "Modeling Of SpaceWire Traffic", Project Executive Summary & Final Report, 2011, 25 p.
- [12] B. van Leeuwen, J. Eldridge, J. Leemaster, "SpaceWire Model Development Technology for Satellite Architecture", Sandia Report, Sandia National Laboratories 2011, 30 p.
- [13] Mirabilis Design, "Mirabilis VisualSim data sheet", 2003. 4 p.
- [14] A. Eganyan, E. Suvorova, Y. Sheynin, A. Khakhulin, I. Orlovsky, "DCNSimulator – Software Tool for SpaceWire Networks Simulation", Proceedings of International SpaceWire Conference 2013, 2013, pp. 216-221.
- [15] ESA. Standard ECSS-E-ST-50-52C, SpaceWire – Remote memory access protocol. Noordwijk : Publications Division ESTEC, February 5, 2010.
- [16] Y. Sheynin, V. Olenov, I. Lavrovskaya, I. Korobkov, D. Dymov "STP-ISS Transport Protocol for Spacecraft On-board Networks", Proceedings of 6th International SpaceWire Networks Conference 2014 Program; Greece, Athens, 2014. pp. 26-31.
- [17] Syschikov, A., Sheynin, Y., Sedov, B., Ivanova, V. "Domain-specific programming environment for heterogeneous multicore embedded systems", International Journal of Embedded and Real-Time Communication Systems, Volume 5, Issue 4. 2014, pp. 1-23.
- [18] ESA (European Space Agency), "Standard ECSS-E-50-12C, Space engineering. SpaceWire – Links, nodes, routers and networks", European cooperation for space standardization / ESA. Noordwijk: ESA Publications Division ESTEC, 2008.
- [19] K. Lahiri, A. Raghunathan, S. Dey "Design Space Exploration for Optimizing On-Chip Communication Architectures", IEEE transactions on computer-aided design of integrated circuits and systems, Vol. 23, No. 6, 2004.
- [20] F.B. Abdallah, C. Trabelsi, R.B. Atitallah, M. Abed. "Model-Driven Approach for Early Power-Aware Design Space Exploration of Embedded Systems", Journal of Signal Processing Systems, 2016, pp. 1-16
- [21] Sheynin, Y., Suvorova, E., Syschikov, A., Sedov, B., Matveeva, N., & Raszhivin, D. Toolset for onboard networks design and configuration. Proceedings of the 29-th Congress of the International Council of the Aeronautical Sciences, ICAS 2014, 2014.