MOST: MODELING OF SPACEWIRE TRAFFIC

SpaceWire test and verification, Short Paper

Brice Dellandrea
R&D Avionics Engineering
Thales Alenia Space
France
Brice.Dellandrea@thalesaleniaspace.com

David Jameux
On-Board Data Systems Division (TEC-ED)
European Space Agency
Netherlands
David.Jameux@esa.int

***Abstract*—MOST (Modeling of SpaceWire Traffic) is a representative and powerful SpaceWire traffic simulator designed to support conception, development and validation of SpaceWire networks. Its recent improvements have targeted simplification and performance enhancement while still being used for sizing the SpaceWire networks of multiple TAS missions. This presentation will focus on its current capabilities and how they were employed on real use-cases then will present the new improvements brought to MOST.**

With the increasing complexity of SpaceWire networks embedded on board satellites and the development of SpaceWire standards and components, this simulator tool proves itself more and more useful.

***Index Terms*—SpaceWire Networks, Simulation, OPNET, MOST, Design, Traffic analysis, Performance assessment, Failure injection, FDIR, Protocol testing**

I. INTRODUCTION

MOST offers the possibility to build SpW network models, selecting and configuring SpW components, simulating high-level applications (FDIR for instance) and to test designs without waiting for HW testing on Avionics benches:

- It allows keeping control on traffic load and identifying weak parts of the network topology,
- It gives load margins and traffic performances (end-to-end delays, buffers sizing),
- It simulates many SpaceWire failure cases and gives the possibility to run various FDIR scenarios,
- It allows decreasing design risks and securing planning thanks to early verification,
- It allows testing the impact of change of Node or Switch behavior to help assessing the criticality of a supplier's non-compliance which can occur during any satellite development phases.

In its current version, MOST is a SpaceWire library of OPNET ® (Open NETwork modeler 17.5). This object-oriented software allows SpW devices configuration thanks to a set of attached attributes. Its graphical editor provides a full set of possibilities to display and analyze simulation outputs.

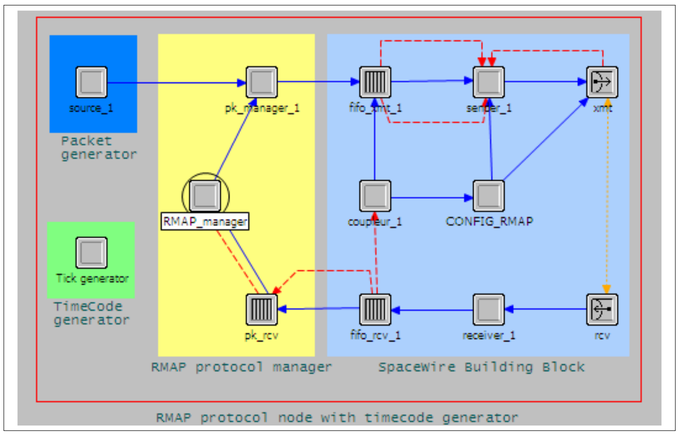
Two versions of MOST currently exist: one early version which has been intensively used by Thales Alenia Space to make its internal simulations and including a wide SpaceWire components library: MOST v1.4. Another version is currently under development and brings many enhancements: MOST v2.2. Both versions will be presented in this document with a focus on the latter.

II. FROM MOST v1.4 TO MOST v2.2

MOST is the result of continuous developments efforts performed since 2006: first as internal Thales Alenia Space development, then with support of ESA to bring MOST to an operable stage through SpaceWire library development, validation with representative test cases (scientific mission and robotic mission), and finally cross-validated with real hardware.

The progressive stages of development of MOST v1.4 followed the protocol stack of the SpaceWire standards: Physical level, Character level, Packet/Network level then User layer for SpW (PID, RMAP, PTP,…) and was internally developed accordingly as depicted in the following figure:

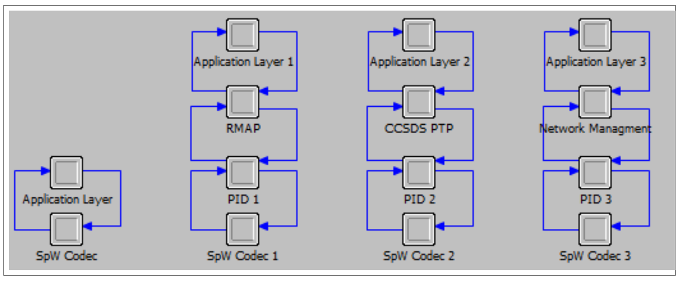
This brings a highly modular structure managed through



finite-state automates which was very convenient during the initial SpaceWire CODEC development phase at the cost of additional OPNET processing time for automates processing. This first development generated MOST v1.4 which was delivered to ESA by end of 2011 and was used, for instance, by ESA to analyze the Bepi-Colombo SpaceWire-based payload Command & Control network and by TAS for the MTG SpaceWire network along with other TAS missions.

MOST v1.4 includes components developed according to some specific SpW component datasheets: the SpW-10X switch with GAR mechanism and round robin for priority management, the SMCS116SpW, the SMCS332SpW, the RTC. This MOST library has also been enriched with generic nodes fully representative of the SpW standard and including protocols building blocks such as RMAP, STUP and CPTP.

In parallel to the TAS simulation activities and continuous MOST v1.4 improvements with the addition of many FDIR functionalities (including dynamic reconfiguration of switching table through RMAP messages), ESA started the development of an experimental branch of MOST (v2.1). This development aimed at improving the usability of the tool by merging the physical, character and packet/network layers in a single SpW CODEC layer and at allowing the construction of protocol stacks as depicted in the following figure:



This new architecture had the advantage of providing a clearer view on the atomicity of the SpaceWire elements: one SpaceWire CODEC including the Physical, Character and Packet/Network levels developed in C-code instead of automates, then PID, RMAP or CPTP elements connected together through OPNET Modeler® exchange links. MOST v2.2 currently under development by TAS took as input this new architecture with the aim to, keep the advantages of the ESA solution while adding all features from MOST v1.4.

III. MOST v2.2 DESCRIPTION

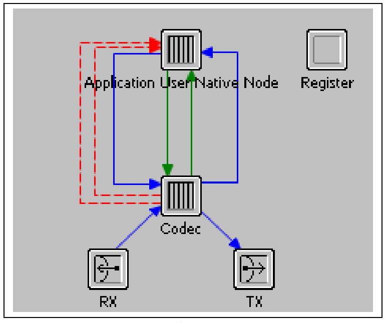
MOST v2.2 targets a release to the SpaceWire community in 2013. In that respect, the library has to be robust and easy enough to be used not only by its developers. This has been at the heart of the development undergone since end of 2012.

First of all, this new MOST library includes some new highly generic components to allow designing networks with very generic behaviors. Each of these generic components is made of multiple building blocks and shares a common building block with all the others: the SpW CODEC which fully complies with the ECSS-E-ST-50-12C SpaceWire. This CODEC building block can be tuned by the MOST user changing a set of parameters available in the OPNET Modeler® user interface:

- Link Enabled to enable/disable a port,
- Autostart is used to enter in Ready mode, node does not send Null characters but waits for Null characters to switch to Started mode,
- Link Start is used to enter in Started mode (sends Null characters). This attributes should be set to disabled if Autostart is set to enabled but MOST accepts both,
- TX Data Rate: this value corresponds to the physical transmission rate of the SpaceWire link,
- RX Buffer Size, this value represents the size of the CODEC reception buffer used to send the right number of FCT,
- Show NULL Messages allows to display NULL messages on SpaceWire link,
- Timer Disconnect is the time at which the CODEC disconnects its SpaceWire link (FDIR),
- Timer Parity Error is the time at which the CODEC simulates a parity error on the next characater received. This Error causes a Disconnect of the SpaceWire link (FDIR),
- Delay For Disconnection After Parity Error is the time between the detection of a Parity Error and the disconnection of the SpaceWire link (FDIR),
- Debug Level with 4 different values corresponding to the level of details on the internal CODEC behavior to print in the Console

Four components are currently implemented in MOST v2.2: a Native Node, a Generic CPTP & RMAP Node, a Generic Switch and a comprehensive simulation of the 10X router with GAR mechanism, round robin for priority management, and the implementation of its RMAP manager for component dynamic tuning (switching table, port enabling/disabling, link speed selection, etc…).

The Native Node is a very simple component, implementing a SpW CODEC and a generic User Application.

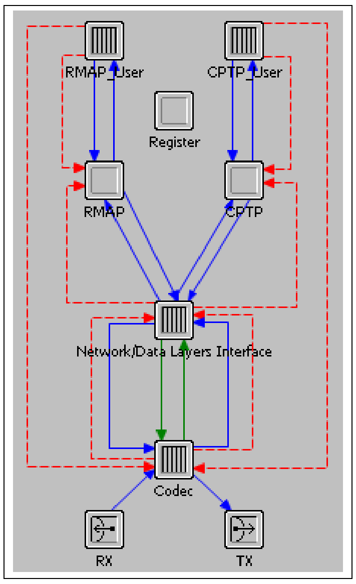


The User Application of the Native Node manages the SpW CODEC as a packet handling level which can be seen as a higher level from the ECSS-E-ST-50-12C point of view. It handles the TICK_IN & TICK_OUT interfaces and the exchange of bytes with the CODEC. It provides packet management: packet generation and packet reception. To support this feature, it implements an input buffer to assemble the bytes received from the CODEC before using it, and an output buffer to send byte per byte their content to the CODEC. It is configurable through the following parameters:

- Timecode Master (Enabled/Disabled) defines if the Node can issue Time-Codes,
- Timecode Interarrival Time is the period of Time-Codes emission,
- Time Code Start / Stop Time is a time defining when the Time-Code service is enabled (respectively beginning and end),

- Debug Level with 4 different values corresponding to the level of details on the internal Native Node User Application behavior to print in the Console,
- Packet Type is an integer between 0 and 98, this parameter allows to identify the packets sent by the Node so that OPNET can compute its specific End-To-End Delay,
- Cargo Size is the size of the packets sent by the application to the CODEC,
- SpW Packet Interarrival Time is a waiting time between each packet sent by the Node,
- SpW Destination Address defined the destination of the packet. MOST accepts logical and physical addressing,
- Packet Generator Start / Stop Time is a time defining when the Packet generation service is enabled (respectively beginning and end),
- SpW Packet Deadline is the maximum time a packet can take to cross the network from its source to destination. The destination Node computes the real end-to-end delay and in case it is higher than the specified value, generates an error in the OPNET console.

The figure here-below provides an overview of the Native Node MOST architecture with a Register for Time-Count storage:



As it can be seen, the Generic CPTP & RMAP Node is more complex; it includes the implementation of PID protocol (ECSS-E-ST-50-51C), RMAP (ECSS-E-ST-50-52C), CPTP (ECSS-E-ST-50-53C) and a generic User Application on-top of each of the CPTP and RMAP protocol layers:

The Network/Data Layers Interface is a layer introduced for direct management of the SpW CODEC interfaces including time-codes handling and exchange of bytes with the upper-layers, it also performs the PID check on the received packets to route the packet to the relevant upper-layer protocol between RMAP and CPTP. To do so, this layer implements on the top-down direction an output buffer to assemble the packets from the data received from the CPTP and the RMAP blocks before sending their bytes one by one with a FIFO process and, on the bottom-up direction, transfers the bytes from the CODEC to one of the protocol layers depending on the PID information. CPTP and RMAP modules implement the protocol part of each standard: packet formatting according to each protocols. Their respective User Applications implement a reception buffer and manage the actions related to the packet content analysis.

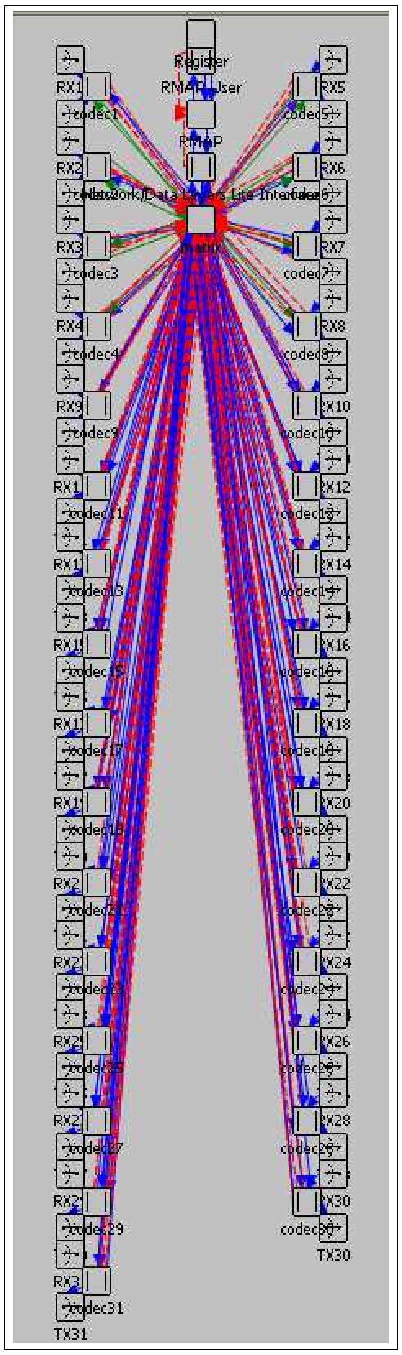
For the Native Node as well as for the Generic CPTP & RMAP Node, the embedded User Application basically perform packet sending and consumption whatever their actual content. The RMAP User Application is more advanced; it allows sending a request which is pre-configured by the MOST user (allowing for instance to configure a switch configuration table). The format of this request is checked at receiver level to determine its effect (for example: READ, READ-WRITE, READ-MODIFY-WRITE), or simply discard in case of invalid request.

The Generic CPTP & RMAP node can be configured through a similar set of parameters than the Native Node with some additional features:

- Network/Data Layers Interface specificities:
  - NDLI Emission Buffer Size is the size of the packet emission buffer,
  - NDLI Local address is the logical address of a Node, this value shall be comprised between 32 and 254. It is optionally used to check the received packets address and discard invalid packets,
  - NDLI Local address Check, this value enables/disables the address check,
- CPTP & CPTP User Layers specificities:
  - CPTP packet EEP Status allows to end a packet with EEP (value = 1), by default all packets are ended with EOP,
  - CPTP Elephant Message Size: this parameter defined the size of an elephant packet,
  - CPTP Elephant Message Destination Address,
  - CPTP Elephant Message Start Time is a time defining when the elephant message is sent,
  - CPTP Reception Buffer Size: this size shall be greater than the biggest packet received,
  - CPTP Service Rate sets the rate at which packets are destroyed by the application
- RMAP & RMAP User Layers specificities:
  - RMAP Command Value is the content of a RMAP command to be transmitted,
  - RMAP Service Rate sets the rate at which packets are destroyed by the RMAP application,
  - RMAP Key is the value of local key for compare with RMAP request,
  - RMAP Reception Buffer Size: this size shall be greater than the biggest packet received,
  - RMAP Reply Delay is the delay between the reception of a request and the creation of its reply,

- RMAP Local Address is the local address used to send a reply and shall be set between 32 and 254,
- RMAP Reply Packet Type is the packet type use for reply packet, this value shall be an integer between 0 and 98

The Generic Switch is a 32-port (31 external ports, plus 1 connected to a configuration port) Switch configurable either in Static Mode (no reconfiguration on the initial table) or in Dynamic Mode (taking into account RMAP messages to change the routing table). It is able to perform Group Adaptative Routing and message priority management.



As it can be seen on this figure, the Generic Switch includes 31 SpW CODEC connected through a switching matrix that implements the SpW Network level. This matrix can switch packets from a port to another including configuration packets to be handled by a local RMAP User able to receive and interpret the requests and to reconfigure the matrix dynamically.

The routing switch building blocks can be configured through the following additional set of parameters:

- Watchdog Timer (Enabled/Disabled): protects the network of elephant messages. If the time taken to transmit a message is higher than the timeout value, the packet will be destroyed automatically in the switch
- Timeout of watchdog timer,
- Switching Table to configure the Switch (including header deletion capability, priority and one or more output ports per logical address for GAR).

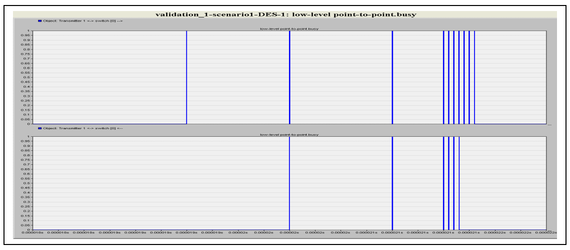
Apart from the RMAP reconfiguration requests, the RMAP & CPTP User Applications implement currently very generic behaviors based on data generation (with selection of size, address, periodicity, emission buffer sizing) and data consumption (reception buffer sizing, application service rate). These basic settings can be refined through the use of parameter files ("gdf" files) which allows configuring for instance a non-periodic data generation sequence or a packets sequence of different lengths, with different destination addresses, ended with EEP/EOP, etc... However, no special action is taken pending on the packet content. This kind of behavior is related to upper-level application (PUS for instance) and is currently not implemented in MOST v2.2. However, a MOST user can implement such functions in C-code in the RMAP User or CPTP User applications. This has already been done successfully by TAS in the frame of multiple simulations.

IV. SOME OF THE MOST v2.2 FEATURES

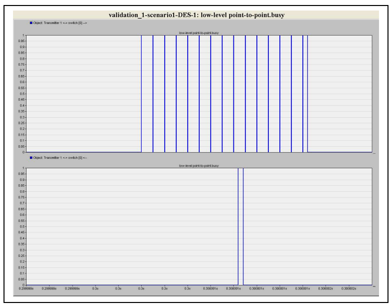
MOST v2.2 aims at providing full visibility on SpaceWire networks behaviour and provides many ways to configure them. Addressing can be either physical or logical, priority can be provided as per ECSS-E-ST-50-12C standard as well as header deletion. Address check can be performed optionaly at receiver level. Moreover GAR and dynamically configurable switching tables are implemented in the Generic 32-port switch.

MOST v2.2 allows configuring the links data rates; the CODECs reception buffer sizes, the packet emission and reception buffers. This has proved very useful to test the effect of SpaceWire network congestions over applications and the possible loss of packets due to emission buffers saturation. As the application does not necessarily consumes incoming packets at the speed of its underlying SpW CODEC, a service rate has been put in place to simulate the actual data consumption capability and provide better representativity for network sizing. For instance a Payload Data Hanling Unit receiving science packets from a high speed SpaceWire network and sending it over a lower speed Radio-Frequency Unit or Mass Memory with an intermediate buffering might block the network in case of reception buffer saturation.

Initialisation of the SpaceWire network is also taken into account with the simulation of exchange of Null messages, and implementation of LinkStart, AutoStart and LinkEnable flags, triggering the corresponding intialisation sequence with the final sending of the Flow Control Tokens:

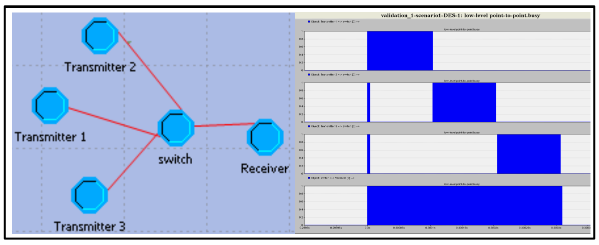


FCT are implemented and exchanged according to the received data characters flow. We can see here-below that on the sending direction of the link, a small packet is emitted with data characters and an EOP (the “small” ending character), while on the reception direction of the link, a FCT is emitted after 8 data characters have been sent:



NULL characters are also taken into account, providing realistic time-code propagation jitters and time spacing between symbols. These NULL characters can either been showed or masked in the simulation with the drawback to lengthen the simulation time and results processing as the links appears very busy (constant oscillations on the link). Their effect is anyway taken into account to compute the sending time of the emitted characters so masking them does not impair the representativity of the simulation.

Wormhole routing and the corresponding switching ports blocking until end of packet transmission is an important aspect of SpaceWire and can be analysed in details using MOST:



We can see on the previous figure the effect of congestion on three nodes willing to send packets to a single “Receiver”: some data characters are stored in the input buffers of the routing switch then, when saturated, the communication is blocked until the emission port is free.

FDIR is a major feature for avionics and data handling engineers. In that respect, many events can be triggered in MOST, from parity bit error to EEP insertion, elephant packet generation, or spontaneous disconnections.

At last, MOST v2.2 has a clear implementation of the SpaceWire protocol stack, providing high flexility to the design of SpaceWire networks through the delivery of generic nodes and switches including basic applications with possible insertion of user-made C-code in identified areas of the generic User Application code for more advance behavior implementation (PUS or instrument HKTM packets generation pending on the reception of a special TC).

It is also possible to design user-made SpaceWire components through the development of specific assemblies (for instance a single User Application with multiple underlying CODECs) through modifications of the generic components using the OPNET interface (for more advanced users). This is optimized through the possible re-use of the already developed building blocks and has been performed by TAS in multiple occasions (for Virtual Channel Multiplexing machines, simulation of Masss Memory behaviors, SpaceWire couplers with different buffering schemes, etc...).

This library is scheduled to be enriched in the future with other specific components existing on the market, for instance the RTC, SMCS116SpW and SMCS332SpW as it was the case in the former MOST v1.4 version. Other implementations could be foreseen on a case-cy-case basis.

V. CONCLUSION

MOST aims at enriching OPNET Modeler® with an operational SpaceWire library available to the SpaceWire community. The recent development performed in the frame of an ESA contract extension brings MOST a major step closer to this objective. Simplification and modularity enhancement to facilitate the design of networks and new protocols are the key features driving the current developments.

ESA own the MOST IPR and intend to release MOST 2.2 to the SpaceWire community in 2013. Some maintenance and further developments of MOST 2.2 and improvements will be performed for ESA by Thales Alenia Space until October 2014.

VI. REFERENCES

[1] “SpaceWire – Links, Nodes, Routers and Networks”, ECSS-E-ST-50-12C, 31st July 2008

[2] “SpaceWire protocol identification”, ECSS-E-ST-50-51C, 5th February 2010

[3] “SpaceWire – Remote memory access protocol”, ECSS-E-ST-50-52C, 5th February 2010

[4] “SpaceWire – CCSDS packet transfer protocol”, ECSS-E-ST-50-53C, 5th February 2010

[5] ISC 2010 – “Simulation of SpaceWire Network” – Thales Alenia Space - France

[6] “Modeling of SpaceWire Traffic” DASIA 2011 & 2012 – Thales Alenia Space - France

[7] “MOST User Manual” 100435074I, 8th March 2013 – Thales Alenia Space - France