EXPLORING NEW SYNERGIES IN SIMULATION AND EGSE

Michiel Haye[(1)], Chris Plummer[(2)]

Dutch Space B.V.[(1)]
P.O. Box 32070, 2303 DB Leiden, The Netherlands
E-mail: m.haye@dutchspace.nl

OHB System AG[(2)]
Universitätsallee 27-29, 28359 Bremen, Germany
E-mail: c.plummer@ohb-system.de

INTRODUCTION

In 2011 OHB, Europe's newest space prime contractor, approached Dutch Space to discuss some new concepts for EGSE systems and simulators for their current and future projects. It was soon found that most of these concepts could be better addressed via an actual trial set-up than via a paper exercise. Therefore a demonstrator Simulator-EGSE system was set up using existing and readily available components, which was then successfully used to explore some key new concepts.

The demonstrator system was realistic enough to be representative of real aspects of AIV usage, while at the same time avoided over-complexity so that the real issues at stake could be fully evaluated. Furthermore the use of the system was very effective in increasing awareness of simulation and EGSE concepts within OHB.

This paper will describe the reference system and some of the concepts explored. The system consists of CMDVS as a lightweight CCS (Central Check-out System), EuroSim as simulator kernel, an EDEN protocol connection between CMDVS and EuroSim, a simple OBC (on-board computer) model implementing PUS services connected to a sensor model, a simulated and a real hardware digital I/O interface between the OBC and sensor model. This system is representative of both an EGSE with hardware in the loop simulation and of a "software only" SVF (Software Verification Facility).

Using this system the paper further explores an object oriented infrastructure for dynamic simulator configuration for hardware in the loop simulation, and synchronisation of an EGSE via PTP (Precision Time Protocol).

THE DEMONSTRATOR

The Wishlist

The project started off with a wishlist of to be included features. This list was further fine-tuned to arrive at those features that were deemed essential for a representative simulation system for AIV purposes. The features are:

- An interface from the simulator to a CCS (Central Check-out System) including monitoring and control of the simulation as well as an OBC (On-Board Computer) TM/TC link based on PUS (Packet Utilisation Standard).
- The ability to run both as a software only and as a HIL (Hardware In the Loop) configuration.
- The inclusion of simple hardware I/O to demonstrate hard real-time simulation.

Apart from these functional requirements, OHB has a strong preference for a modular architecture where as much as possible off-the-shelf products are used, not necessarily restricted to the space domain.

Architecture

Fig. 1 shows the selected building blocks for the demonstrator system. For the CCS the Terma CMDVS product is selected. This is already in use at OHB for several projects and is a lightweight application that can be used for both EGSE and software only environments. EuroSim is selected as simulation environment covering both the hard, soft, and non real-time use cases. For the hardware I/O a NI (National Instruments) card is selected as a typical COTS manufacturer which provides a wide range of I/O for industrial interfacing.

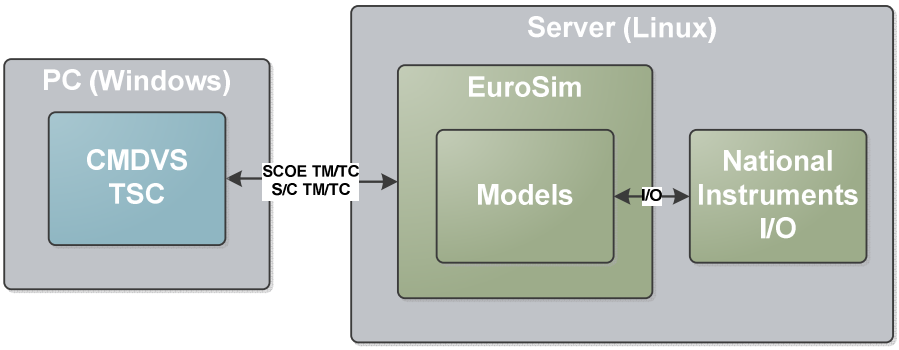

Fig. 1. Top level overview of the demonstrator system.

The design of the interface between CMDVS and EuroSim is shown in Fig. 2. A “de-facto” standard interface is used between the CCS and the simulator: the EDEN protocol. The protocol uses a single TCP/IP link to multiplex simulator monitoring and control and spacecraft TM/TC. On the simulator host, a dedicated EDEN I/F application translates between EDEN and native EuroSim interfaces. For the monitoring and control of EuroSim, the evClient library is used, for the OBC TM/TC link the esimLink library provides direct access of the OBC model to the TM/TC source packets.

The simulator models are intentionally simple, to not distract from architectural considerations. In terms of model design, also the simplest approach is chosen where the interface between model and simulation infrastructure is based on the standard EuroSim interface for C models. There are three models (see Fig. 3).

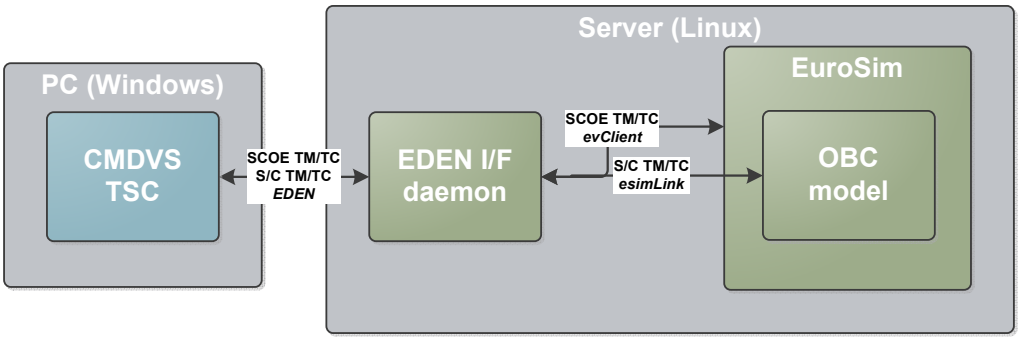

Fig. 2. Interface between CMDVS and EuroSim.

The OBC model is a simple model of an on-board computer that implements PUS service 1 to acknowledge commands, service 3 to enable and disable periodic sending of housekeeping telemetry, and service 5 to report events related to temperature limits. The model has HPC outputs that can be used e.g., to switch equipment. In the simulator these are connected to a sensor model. It also has pulse inputs that are used to detect an "alive" signal from the sensor model.

The sensor model represents an equipment that can be switch on or off via command pulses. When on it sends a periodic pulse train as output (to the OBC model).

The purpose of the "electrical interface unit" (EIU) model is for the demonstration of hardware in the loop. The model simply passes signals from its inputs to its outputs, and the real hardware would thus be a straight wire. Hence, for demonstration of EIU hardware in the loop we can use a wire.

For the hardware in the loop demonstration, the simulator is equipped with a National Instruments digital input/output PCIe card (or, alternatively, connected to a National Instruments PXI chassis with equivalent PXI card). The models could directly get and set the I/O channels via an off-the-shelf kernel level driver.

In the simulation only configuration (Fig. 3 left) the EIU model routes pulse on/off commands from OBC to sensor model and an alive pulse train in the reverse direction. If a real EIU hardware unit needs to be included in the closed loop testing, the EIU model needs to be replaced by an EIU stimulus model that ensures correct I/O to the real EIU. For this particular case, where the real EIU is just a wire, the stimulus model must send on/off pulses to one end of the wire and read back the pulses from the other end of the wire. The reverse holds for the alive signal.

Results

The system is available for demonstration at the EuroSim exhibition boot at the SESP2012 conference. An impression is shown in Fig. 4. Via CMDVS both the simulator and OBC model are controlled and monitored. The system can run in SVF and hard real-time mode. In the latter case, the available I/O can be used to demonstrate the hard real-time performance of the complete system.

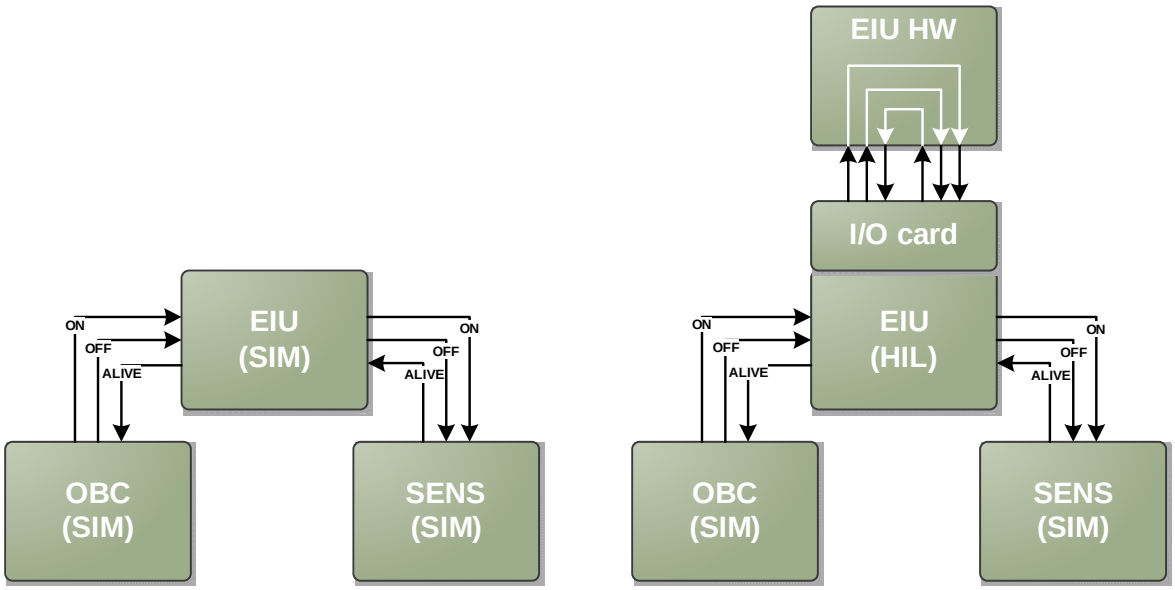

Fig. 3. Model overview of the demonstrator system.

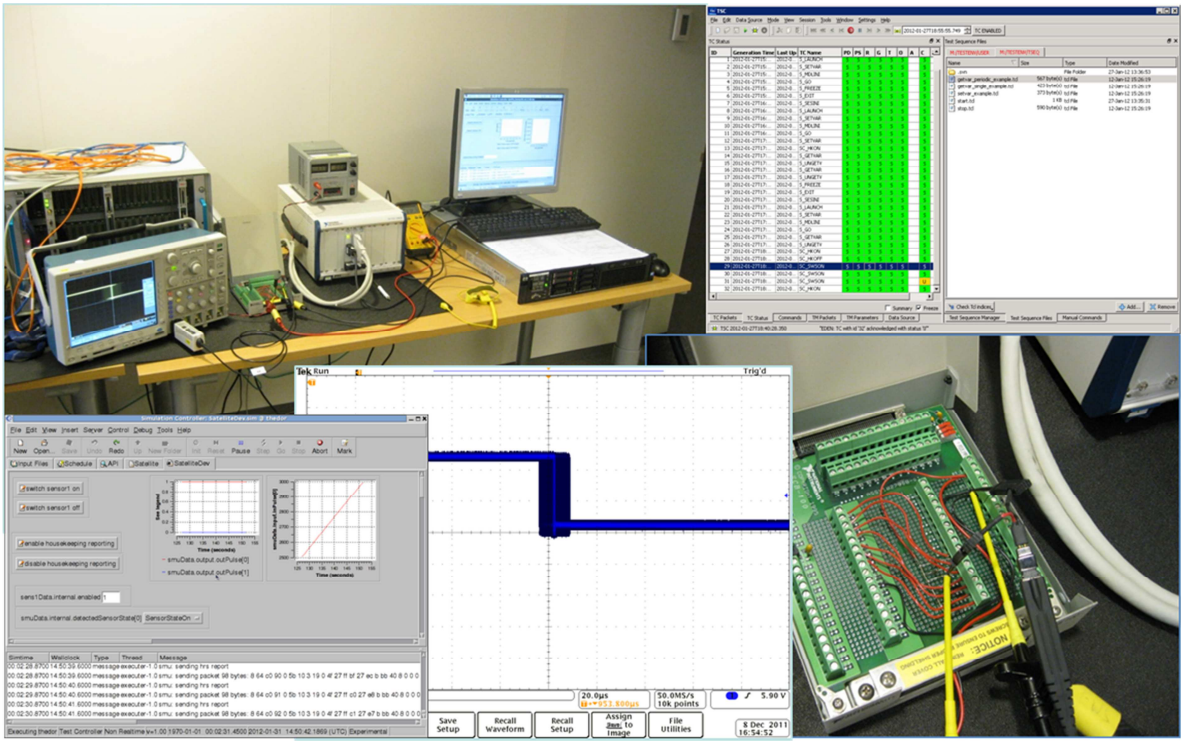

Fig. 4. Snapshots of the demonstrator system from left to right and top to bottom: Overview; CMDVS; Eurosim; Jitter measurement on pulse generation; “Real EIU hardware”.

EXPLORING MODEL INTEGRATION INFRASTRUCTURE

The simple example models of Fig. 3 already show a number of features that are important for the user of an AIV simulator:

- The ability to easily configure the simulator for different AIV hardware in-the-loop configurations without detailed knowledge of the underlying simulator architecture.
- The ability to inject errors on model variables, especially the interface variables of models.

A solution for this has been implemented in the AIV simulators for the Herschel-Planck and Gaia programs, but we now want to explore the possibility to extend this solution with additional functionality for the simulator developer that simplifies implementation issues that commonly arise in simulators used for AIV. These functions are:

- Support for an object oriented modelling approach, including the model integration.
- Support for scheduling against different timelines.

All this requires good concepts and a lot of "model glue" that we want to simplify and make to a large extent independent of a particular project.

For the user level AIV configuration of the simulator we defined the so-called model mode: Each model instance corresponding to an equipment has an associated variable that selects the AIV configuration of the model. In its simplest form it can have e.g., the values "disabled", "simulate equipment", "stimulate hardware in-the-loop equipment". Depending on the value of the model mode, the correct equipment submodels and corresponding data exchanges need to be enabled. To support this, an infrastructure is developed where specific model entrypoints can be registered to specific values of a model mode. Data exchanges (also entrypoints) can be registered to specific values of two model modes, so that they can be disabled/enabled depending on the configuration of either model. An example is shown in Fig. 5.

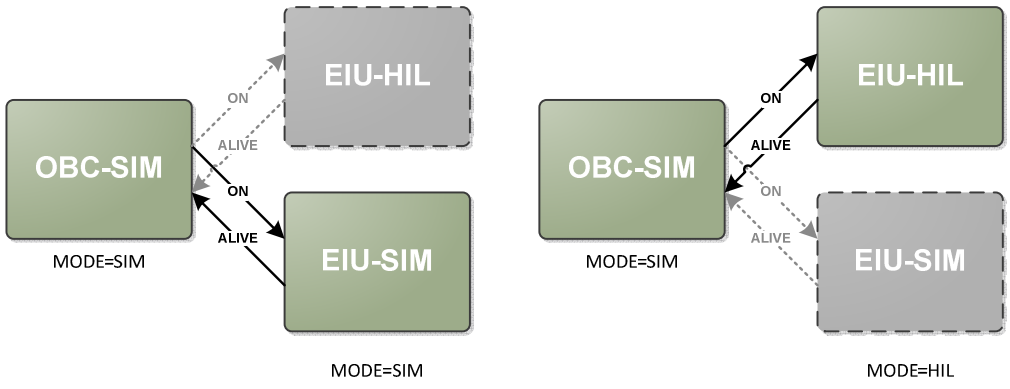

Fig. 5. Model mode settings determine which (sub)models and data exchanges are activated.

For the ability to inject errors and to formalize the model interfaces, an additional model interface layer is defined where each model publishes its interfaces to be used for integration to the infrastructure. This

results in specific interface variables (input and output ports) to be created by the infrastructure, which are connected to local variables of a model. Data exchanges between models are always via model ports. The copy between local and port variable can triggered by the execution of a data exchange (active port) or explicitly scheduled by the simulator developer (passive port). When creating a port, an error injection function can be registered, fully programmable by the developer, which triggers when data is copied between port and local variable. The error injection function is accessible to a simulator user via error injection variables that are generated as part of the port object. The approach is shown in Fig. 6. This port approach is the object oriented evolution of the EuroSim datapool.

Finally, support for scheduling against different timelines has been added. This is often encountered when some models need to be scheduled synchronised to physical time, e.g., spacecraft dynamics, and other models to the on-board time cycle, e.g., equipment command handling. Nevertheless, some synchronisation between the different timelines is desirable to avoid parallel execution of models that share data and to be able to force a deterministic order of model execution when needed. The solution is to schedule all models at the highest needed execution frequency against simulation time. The simulation infrastructure then dynamically enables individual models according to their scheduling definition and the evolution of the timelines. The evolution of the timelines needs to be implemented by the simulator developer (e.g., start of cycle interrupts from a 1553 front end can be used for the on-board time line). An example is shown in Fig. 7. Note that these principles can be applied not only to hardware in-the-loop simulators, but also to SVFs, thus allowing to use the same schedule for these two configurations.

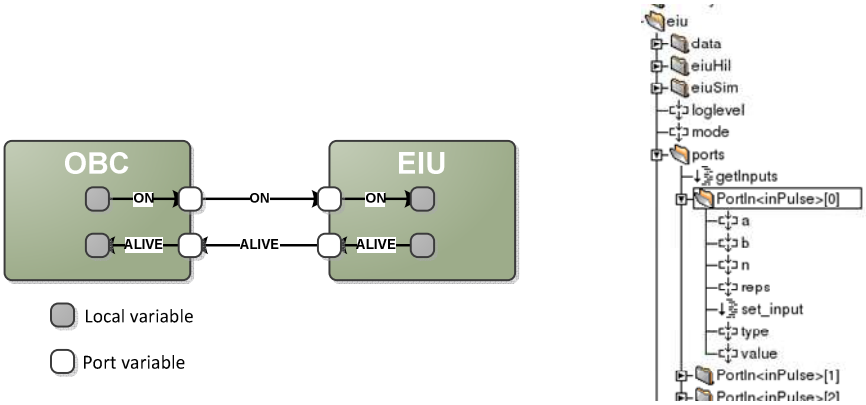

Fig. 6. Definition of model ports. To the right a snapshot of the corresponding definition in the EuroSim data dictionary (EIU on pulse corresponds to inPulse[0]).

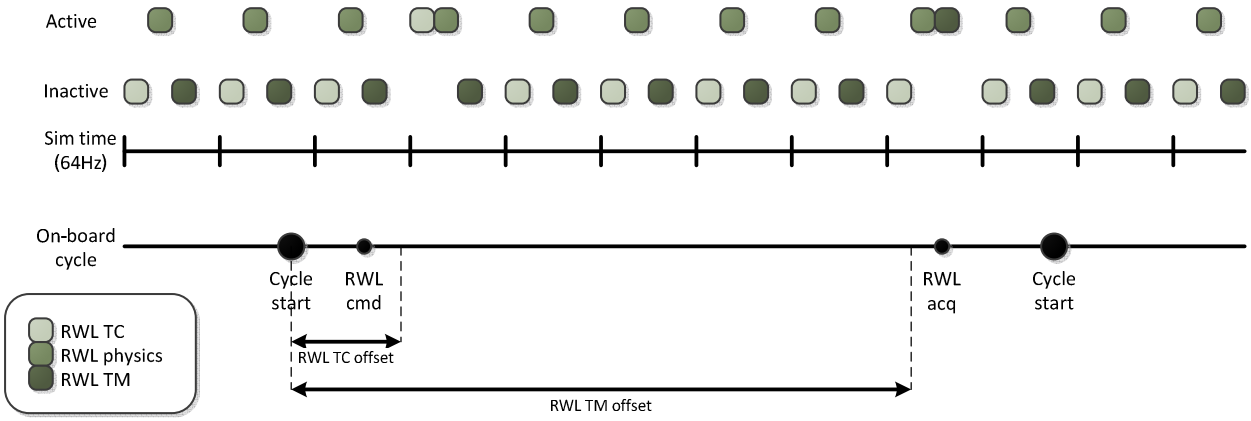

Fig. 7. Example of "dynamic" scheduling using two timelines for a reaction wheel (RWL) model. All (sub)models are scheduled at the highest frequency but inactive. They are activated following their scheduling criteria. Physics is scheduled at 64Hz. RWL TC and TM at fixed offsets relative to the on-board cycle.

EXPLORING PRECISION TIME PROTOCOL

Time synchronisation within EGSE is conventionally done via two parallel mechanisms: Hardware based synchronisation for stringent timing requirements and software based synchronisation for "standard" computers not requiring too stringent constraints. The de-facto standards within EGSE are IRIG-B for hardware synchronisation and NTP for software. Recently a new network time synchronisation protocol, PTP, has matured outside the space domain. It promises very accurate synchronisation (microseconds and below) via an Ethernet based protocol. This could simplify and reduce the cost for time synchronisation within EGSE.

As a first step to explore the feasibility we extended the demonstrator system with a PTP grandmaster clock as "EGSE time generator" and a PTP slave card inside the simulator host. With this set-up we measured the accuracy of the synchronisation by comparing the pulse per second (PPS) outputs. Two network layouts were studied. The first has a single network connection from the SCOE to the EGSE LAN, the second has a separate network for PTP synchronisation. Results are shown in Fig. 8.

The single connection under normal network load is in general accurate well within 0.1ms. However, during longer measurements (above an hour) occasionally higher peak values are observed. To estimate the upper boundaries we generated high bandwidth network traffic between CCS and SCOE hosts (>10MB/s). Under this load the synchronisation accuracy was within ±2ms (Fig. 8 top).

When the normal network traffic and PTP are separated, much higher accuracies can be obtained. This is shown at the bottom of Fig. 8. In this configuration the network load between CCS and SCOE host obviously does not affect the PTP synchronisation, which is accurate within ±0.2µs.

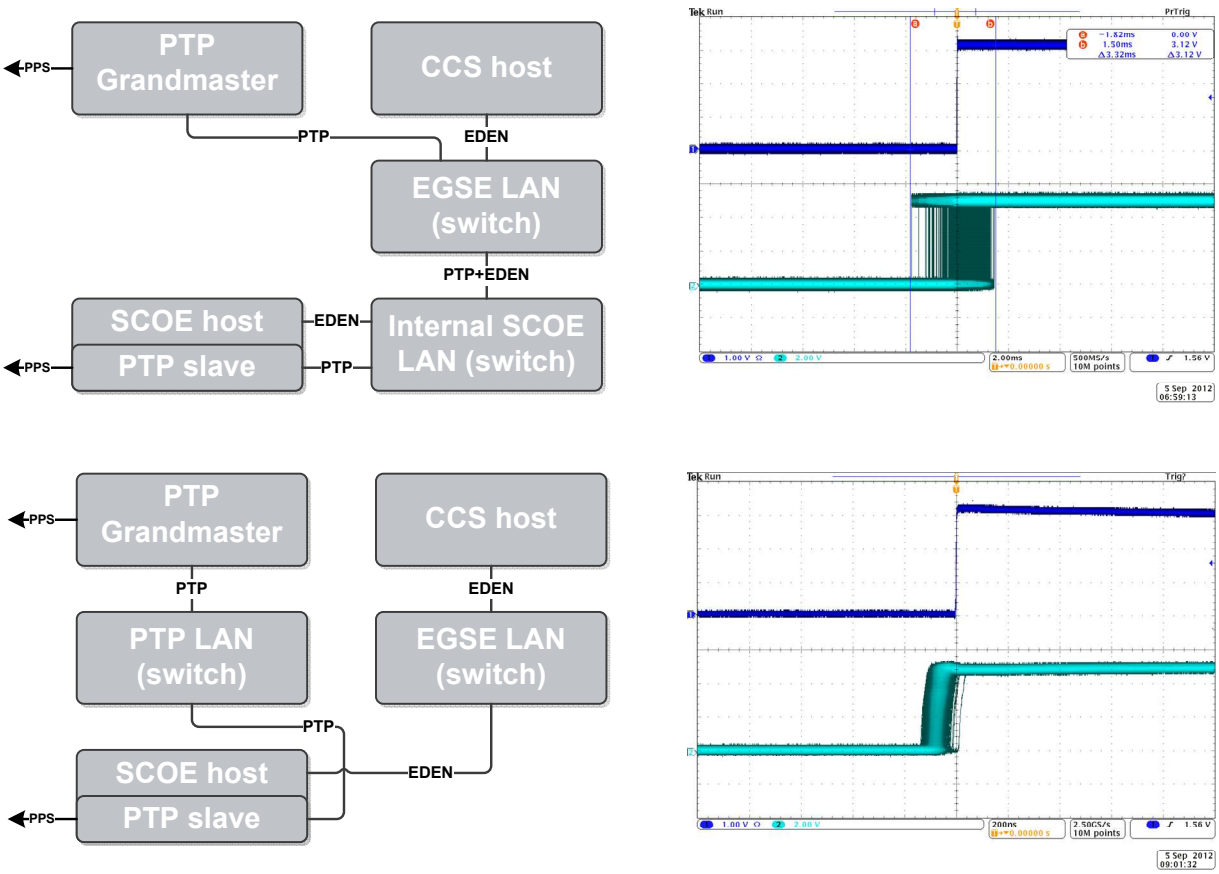

Fig. 8. Testing of PTP synchronisation under network load with two network topologies. PPS signals generated by master and slave are synchronised within ±2ms (top) and ±0.2µs (bottom).