SpaceWire SystemC model

Yuriy Sheynin, Valentin Olenev,
Ilya Korobkov, Nikita Martynov, Arkady Shadursky

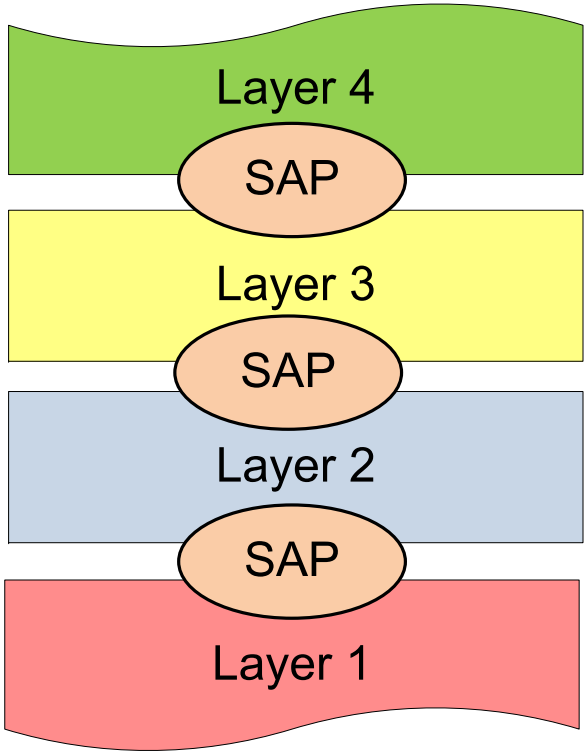
St. Petersburg State University of Aerospace Instrumentation
St.Petersburg, RUSSIA
sheynin@aanet.ru

SpaceWire Modeling

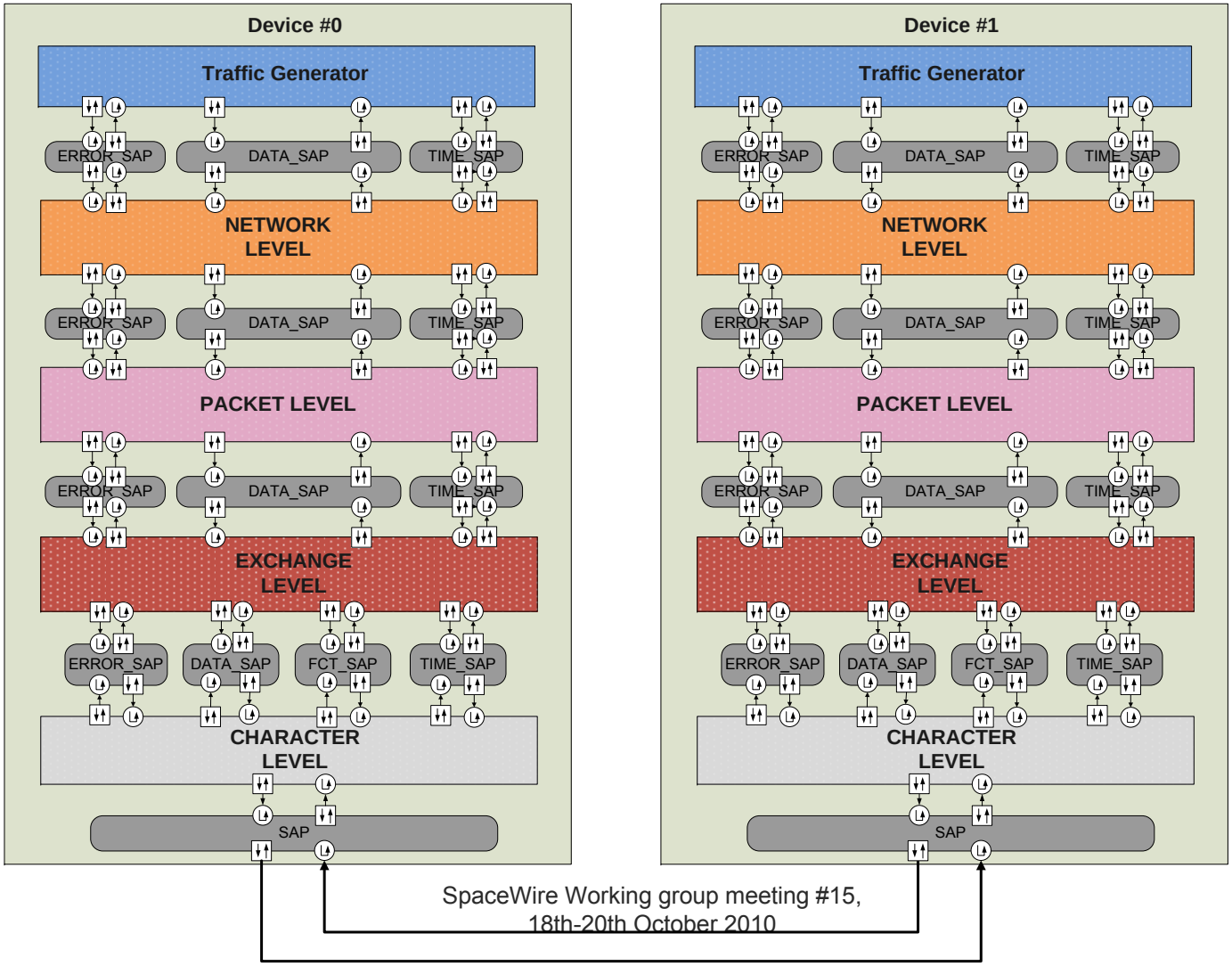
- To check the described in the specification protocol mechanisms;
- To get an executable specification model for testing;
- Modelling of the transport layer and application layer protocols on top of SpaceWire model.

SystemC Model view

- The SpaceWire model is a layered structure.
- Layered approach :
  - gives an ability to test every layer independently
  - protocol data units of every layer could be seen
  - easier to make updates for internal model mechanisms
- Service Access Points (SAP) guarantees the correct data handshake



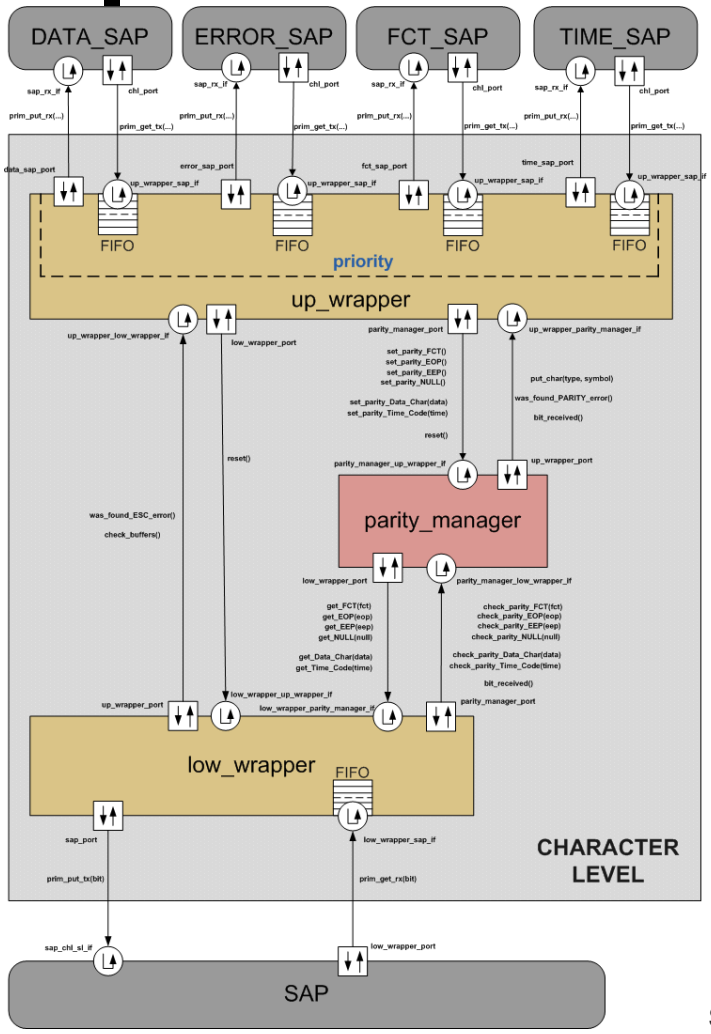
SpaceWire model structure

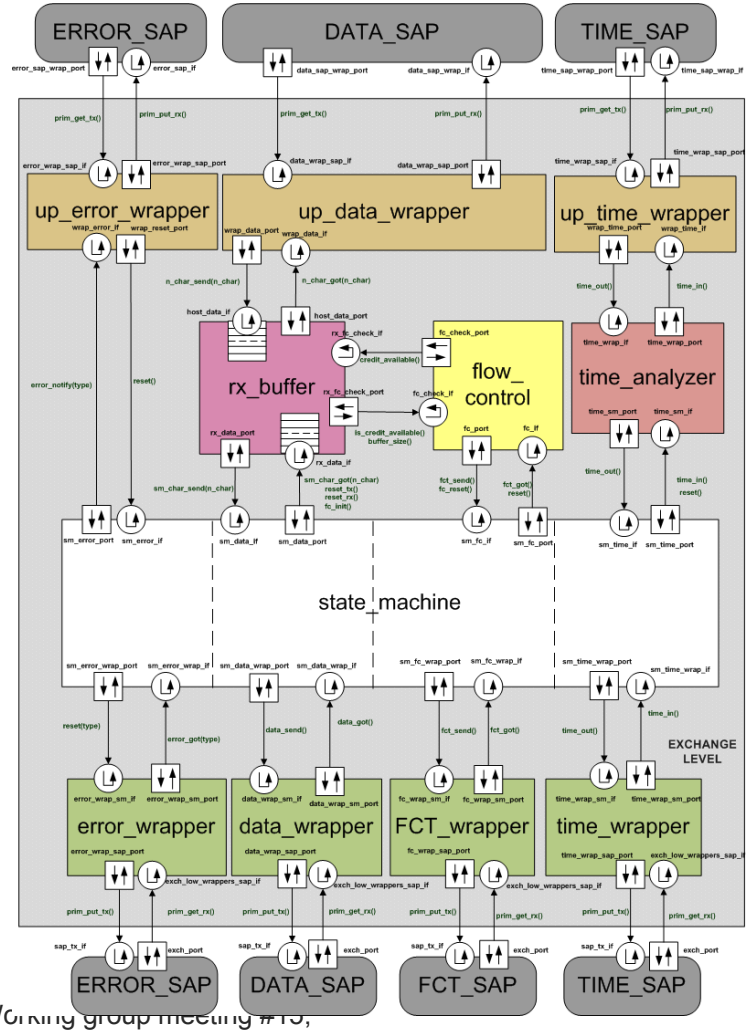


SpaceWire model features

- Every layer is a number of modules written in SystemC language;
- Every module performs a specific function of a layer;
- Logging system is implemented;
- An example of SpaceWire model communication implemented for the point-to-point case.

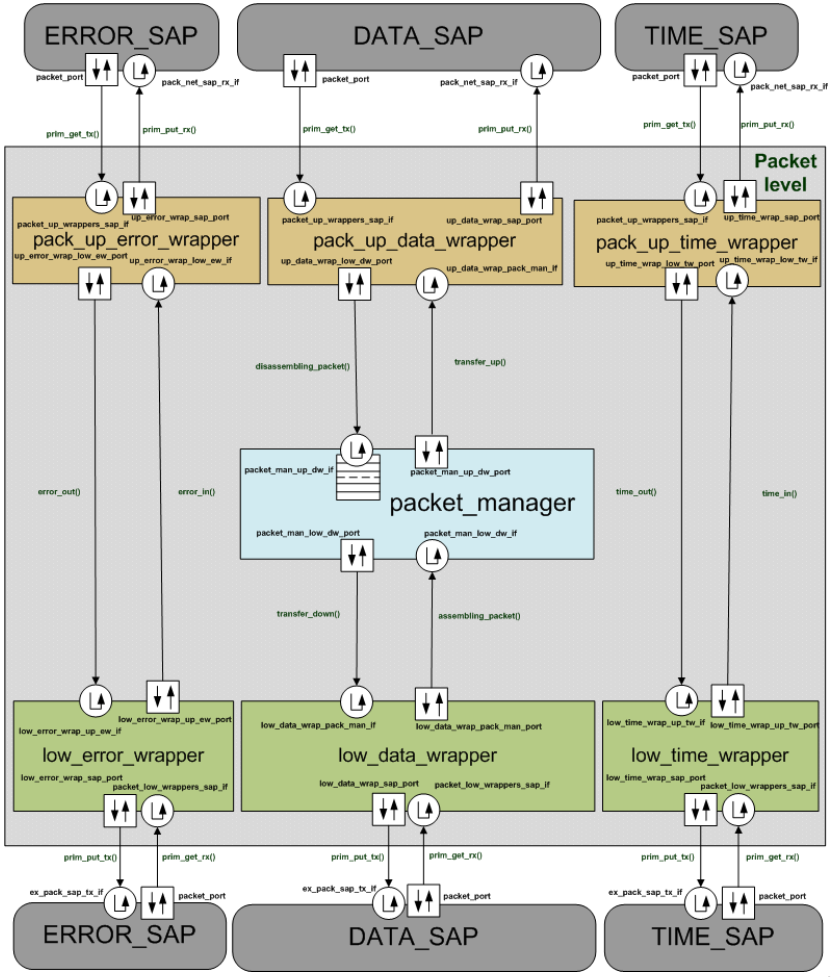
Architectural diagrams: Character and Exchange levels

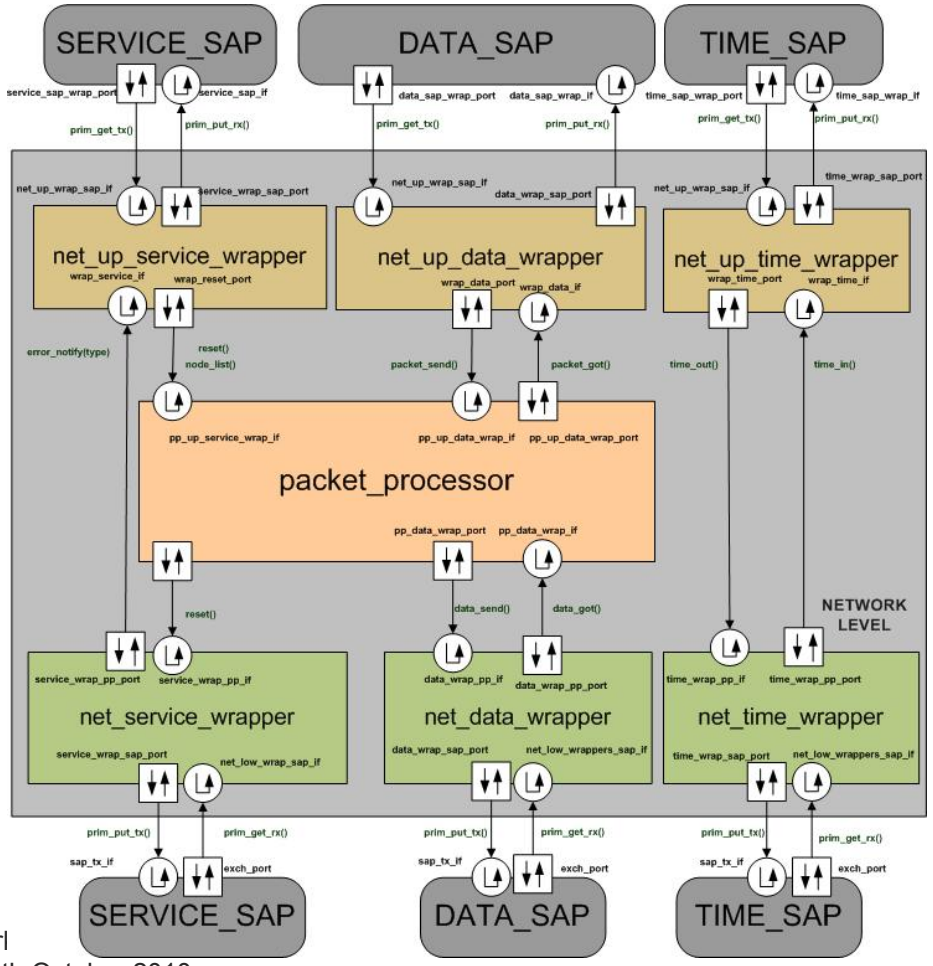




Exchange level

Architectural diagrams: Packet and Network levels





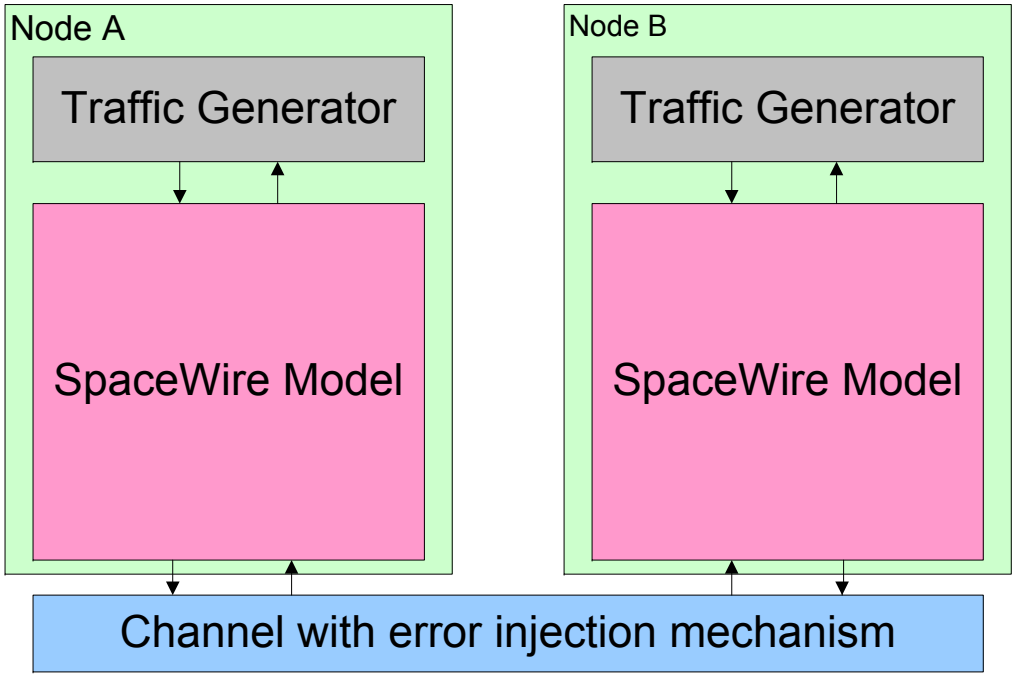
Protocol mechanisms checking

- Could be done along with the specification development;
- Gives an ability to find bugs and vulnerabilities in the specification algorithms;
- Protocol mechanisms checking before the physical implementation stage.

Model testing

- Error situation tests:
  - Error injection mechanisms for the channel between the nodes.;
  - Generation of a traffic with errors.
- Conformance testing;
- Hardware Tester implementation

Model testing



- The picture shows the point-to-point communication testing
- Traffic generators generate data, process and log the incoming data. Erroneous data generation is available.
- Channel is a full duplex channel model. Erroneous data generation is also available.

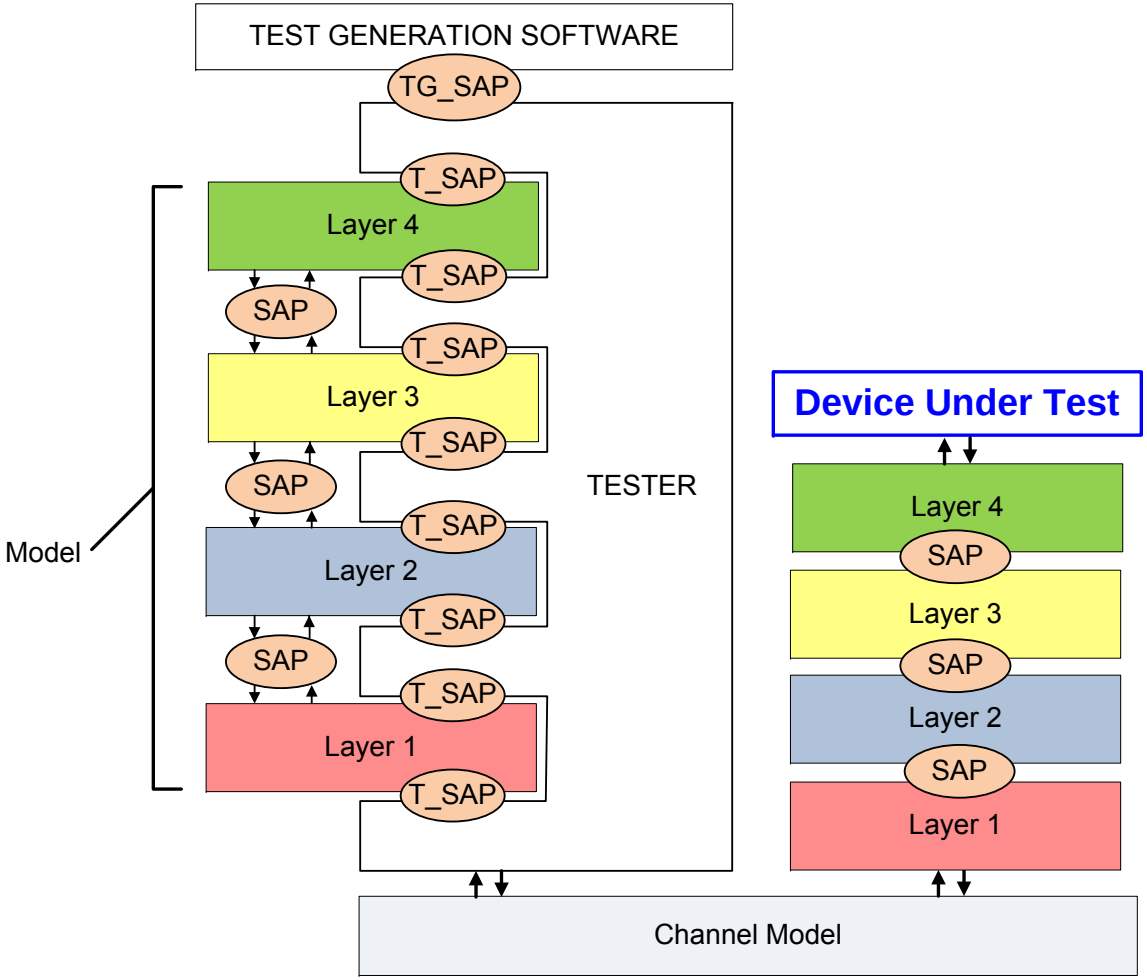
Implementation testing

- Testing software implementation
- Driver implementation
- Implementation of the point-to-point communication of the two instances: model and DUT (Device under test).

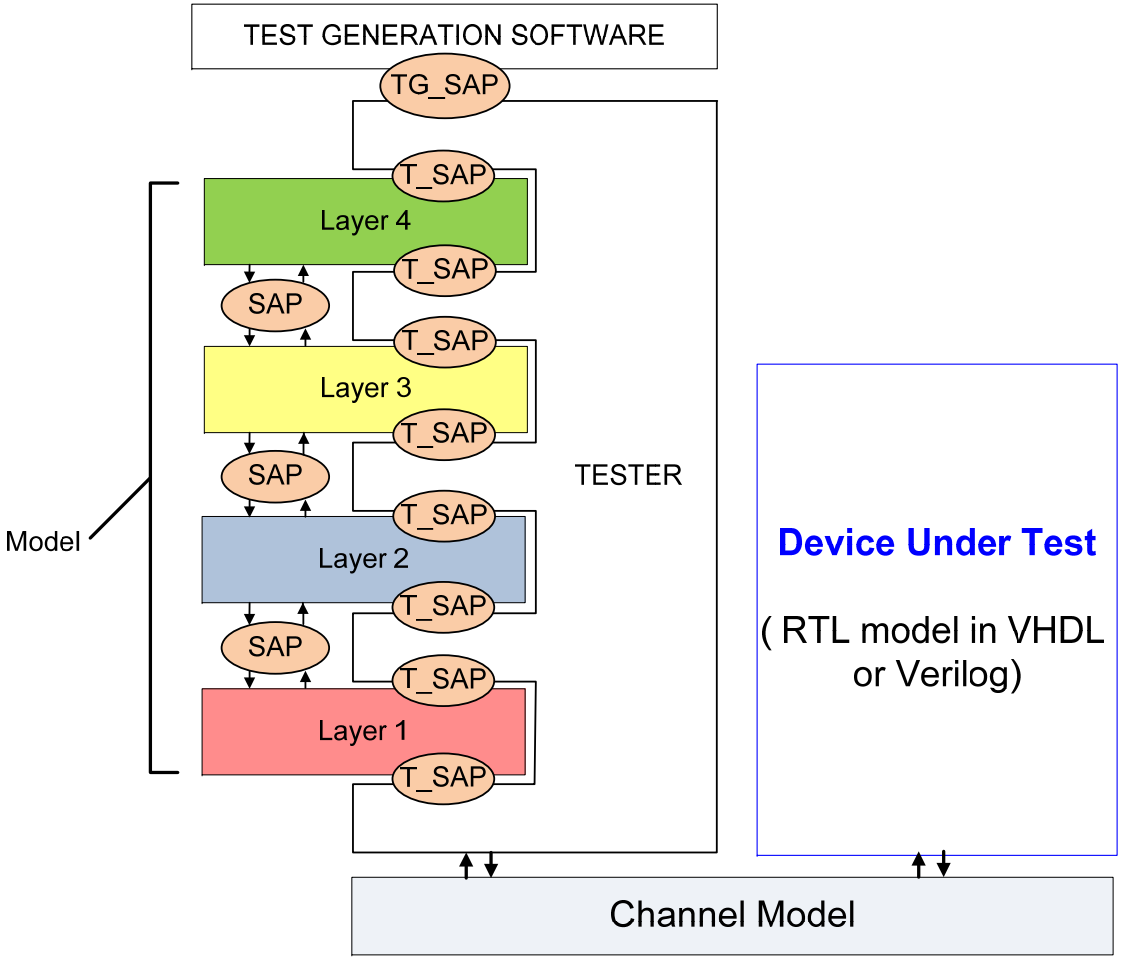
Device Model Testing

- Four main parts:
  - The Tester with the protocol stack included
  - The channel for communication
  - The Test generation software
  - A Device Under Test
- Device Under Test is an implementation of a real device in one of the modeling/HDL languages.
- Test Generation Software and Device Under Test communicate via the two protocol stacks plus the Channel
- The Tester is responsible for error injection, logging, configuring and data transformation.

Device Model Testing: Example 1



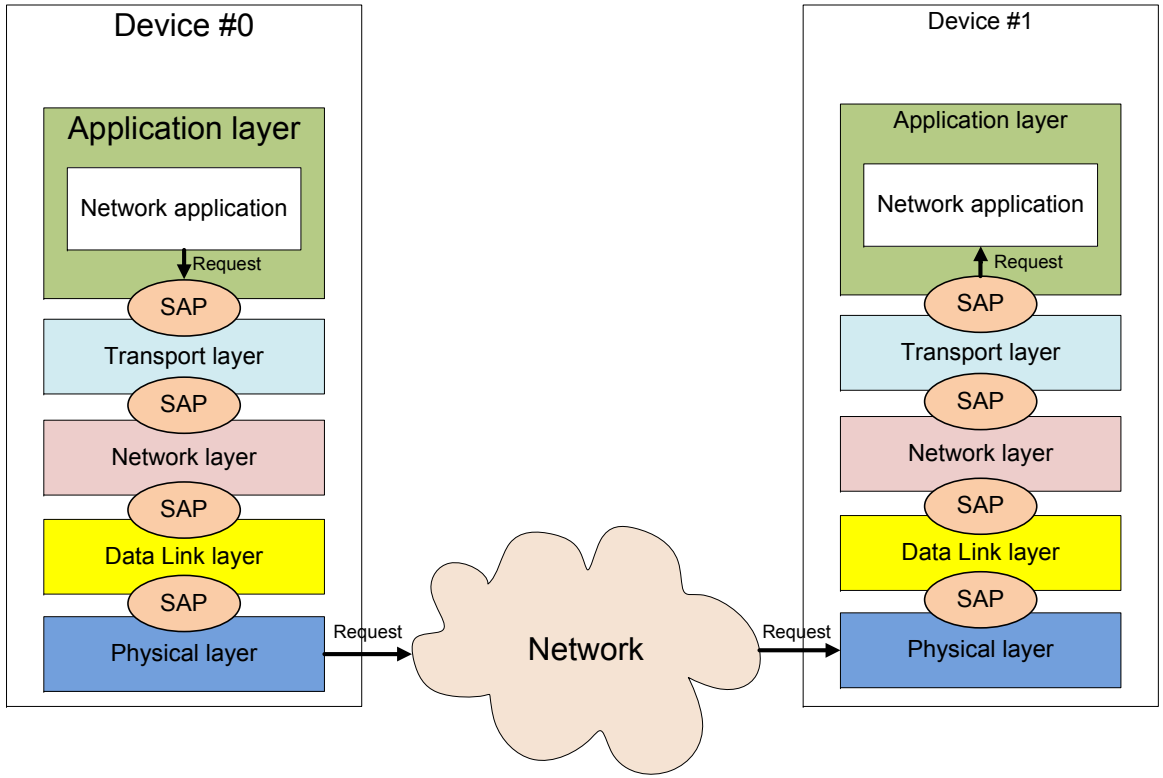
Device Model Testing: Example 2



Transport protocols modeling

- Development of protocols to work over SpaceWire.
- Gives an ability to see the basic SpaceWire protocol stack with transport protocols joint work;
- Gives an ability to model the work of applications through the SpaceWire network;

Transport protocols modeling: Example

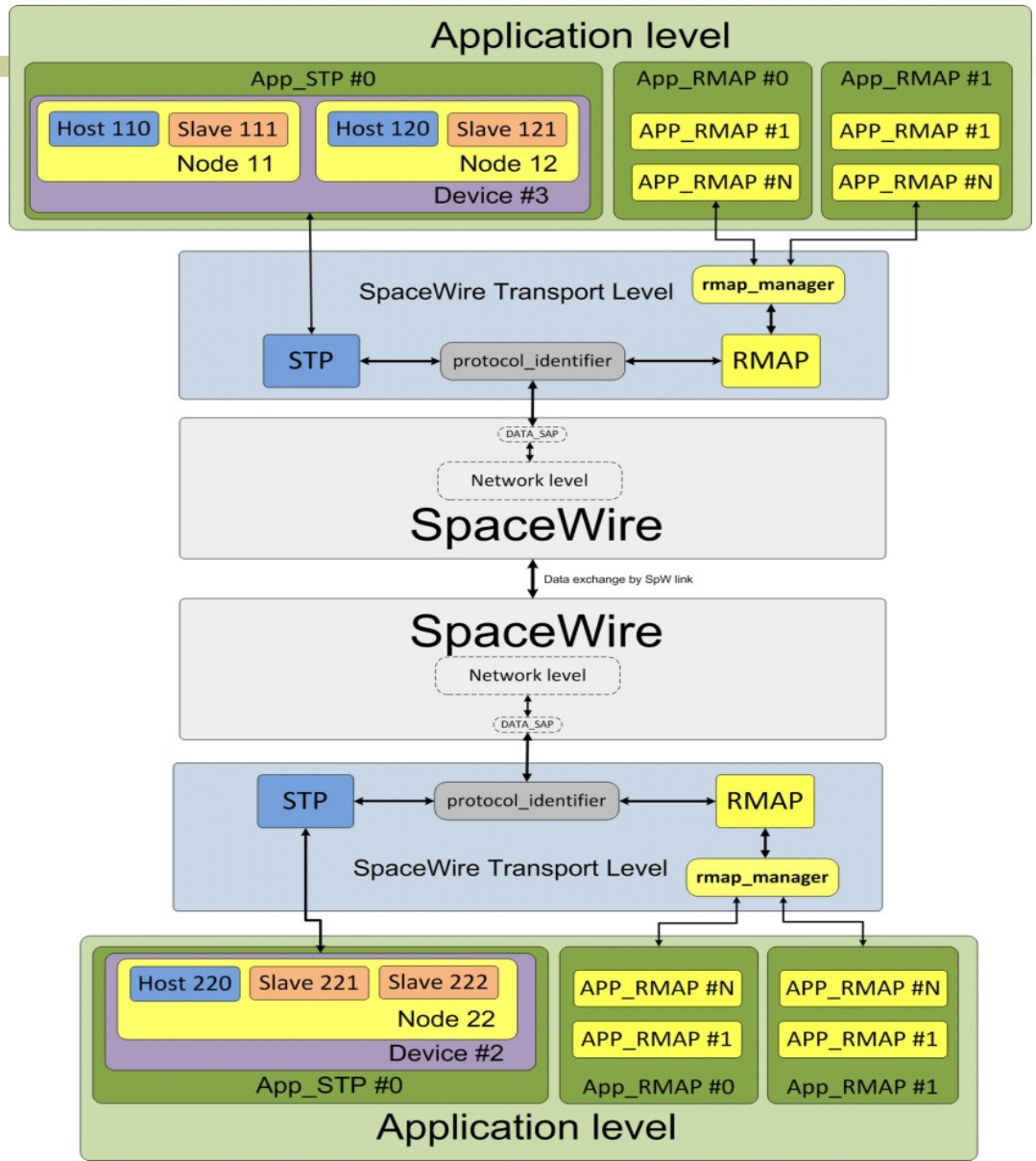


- An example of Application #0 sending request to Application #1 through the communication protocol model and a network model.

RMAP and STP modelling on top of the SpaceWire

- Modelling of transport protocols over the SpaceWire model
- STP and RMAP specifications checking
- Space Wire model joint work in the protocol stack
- Ability to model the applications work through the SpaceWire network

RMAP and STP modelling : Architectural diagram



Modelling results

- The SpaceWire model is implemented inline with the current version of the specification+;
- CRs for the SpaceWire specification found;
- STP and RMAP transport protocols models implemented;
- STP and RMAP models works on top of SpaceWire model;
- CRs for the STP specification found;
- CRs for the RMAP specification found;

Thank You!