OPNET Modeling of Plug and Play Spacecraft Networks

This presentation does not contain any ITAR restricted material.

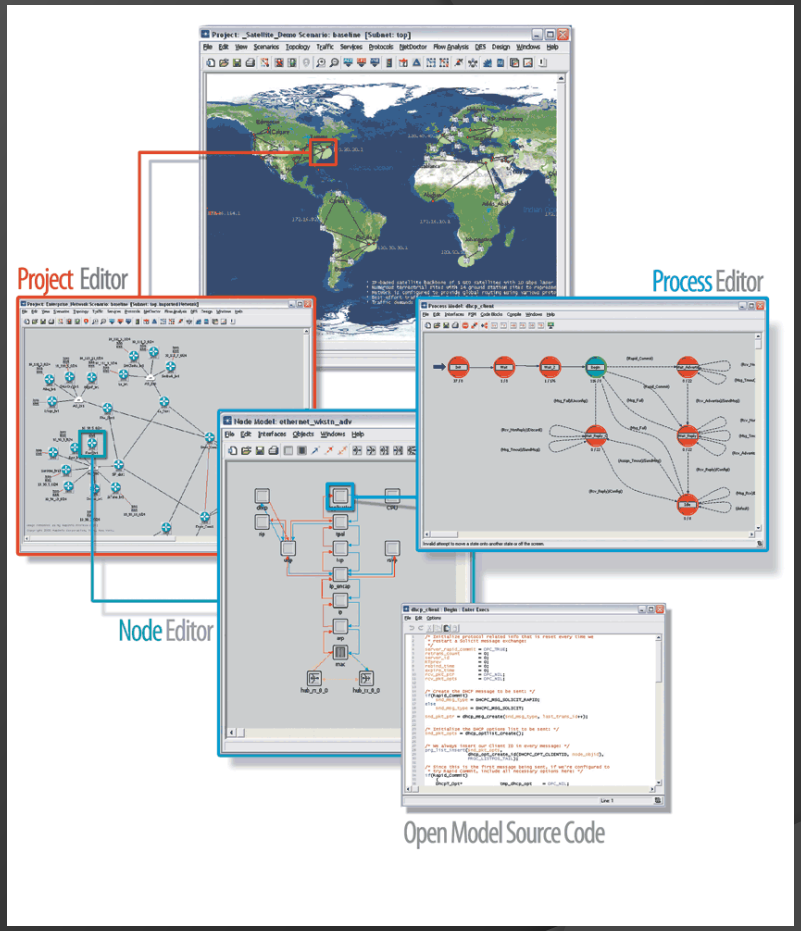
Carlos Quiroz
Southwest Research Institute
carlos.quiroz@swri.org

Agenda

- Brief OPNET Modeler overview
- SwRI test bed configuration
- SPA/OPNET implementation
  - SSM SPA modifications
  - OPNET Modeler integration
  - External Co-Simulation Controller
  - SSM SPA with OPNET recap
- Advantages SPA/OPNET integration
- Considered alternatives
- Conclusion
- Questions/Comments

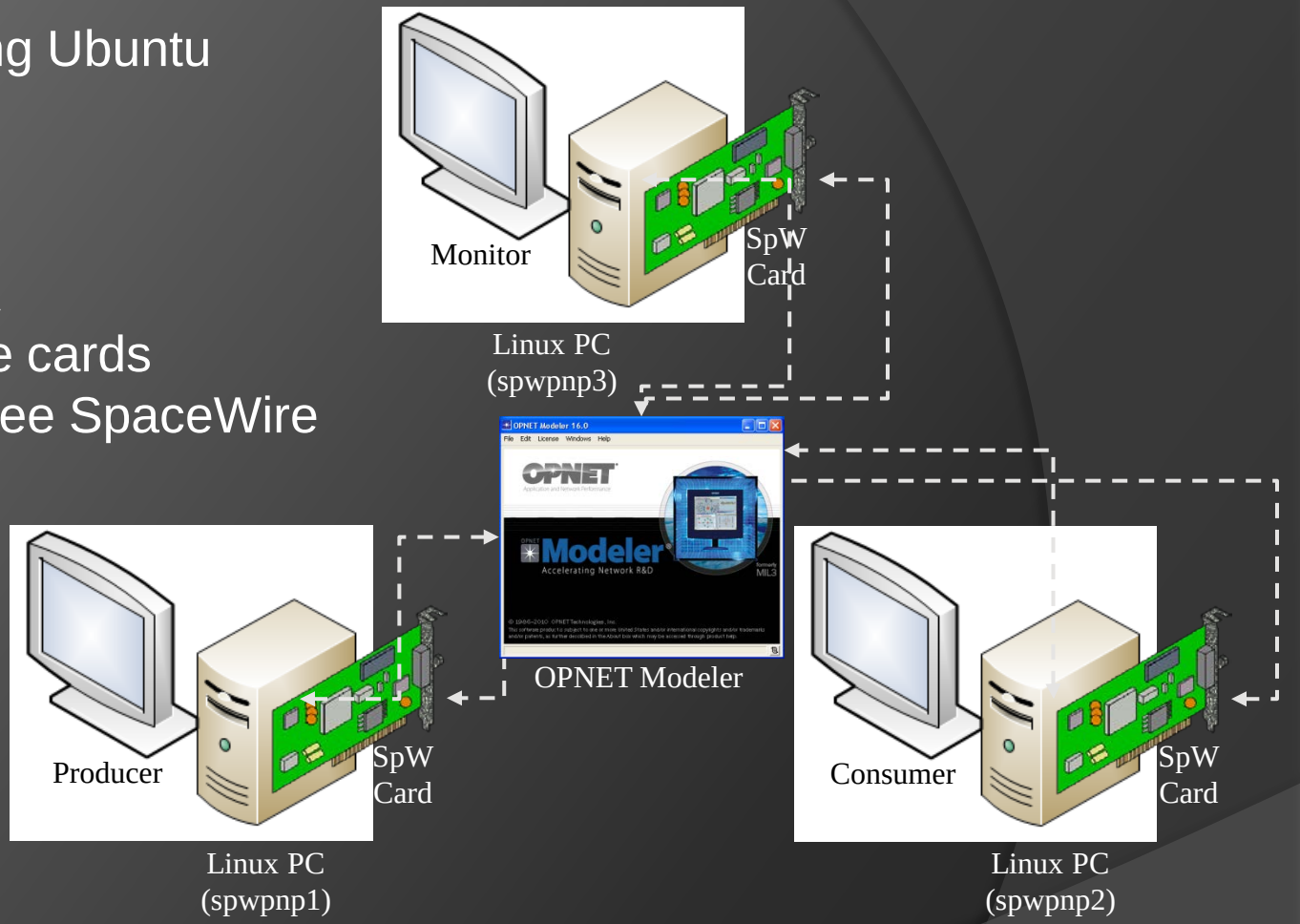
OPNET Modeler

- Network modeling and simulation tool
- Can be used to perform data collection and analysis of network traffic
- Co-Simulation allows OPNET Modeler to interact with external system during simulation
  - Example integration with SSM SPA (AFRL PnP middleware)



SwRI Test Bed Configuration

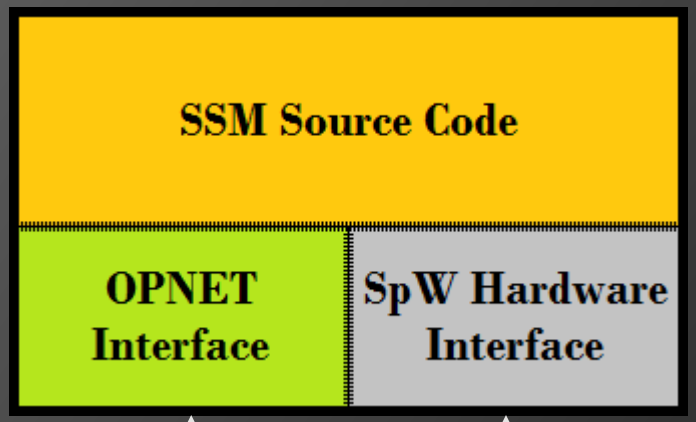
- 3 Linux PCs running Ubuntu
  - Producer
  - Consumer
  - Monitor
- Running SSM SPA
- Custom SpaceWire cards
- 8-Link STAR-Dundee SpaceWire Router

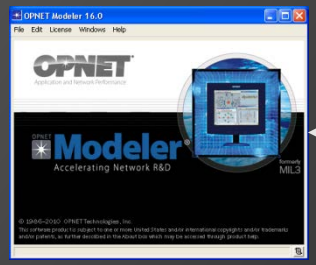


Lab Configuration with SpaceWire Hardware

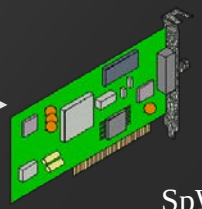
SSM SPA Modifications

- Original interface to custom SpaceWire card driver
- Added custom OPNET interface to SSM SPA code
- Custom interface to OPNET PC using sockets
- Acts as client





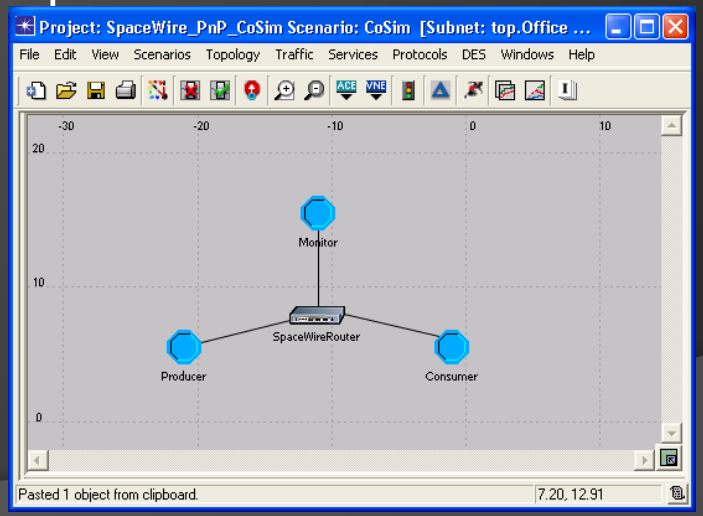
OPNET Modeler



SpW Card

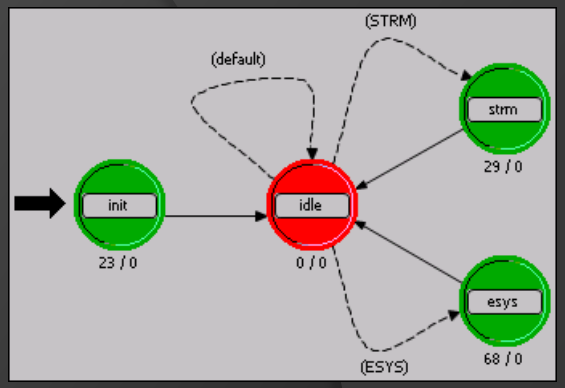
OPNET Modeler Implementation

- Recreated lab SpaceWire configuration in OPNET Modeler
  - 3 Nodes:Producer, Consumer, Monitor
  - SpaceWire router
- Created one model for 3 nodes
  - Receives incoming messages from external Co-Simulation Controller
  - Forwards incoming message into simulated network
  - Sends incoming messages from simulated network to external Co-Simulation Controller
- Created simple model for SpaceWire router
  - Routes messages according to SPA-S message route (path addressing)
  - Removes current hop from route

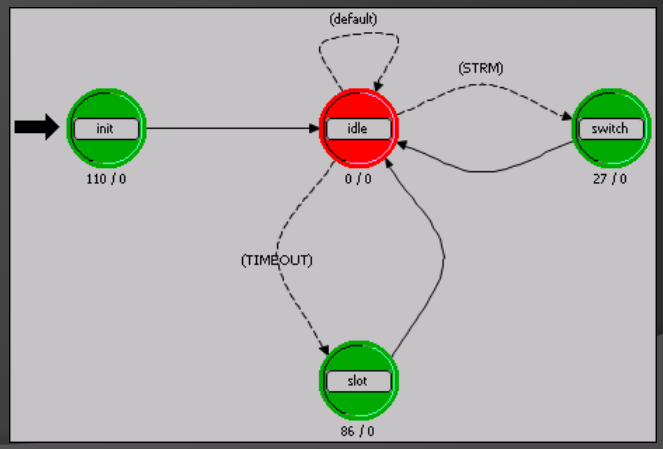


OPNET Modeler Implementation

- OPNET SPA node
  - Not designed to implement SPA standard
  - A means to interact with an external source
  - Translates messages to/from OPNET packets

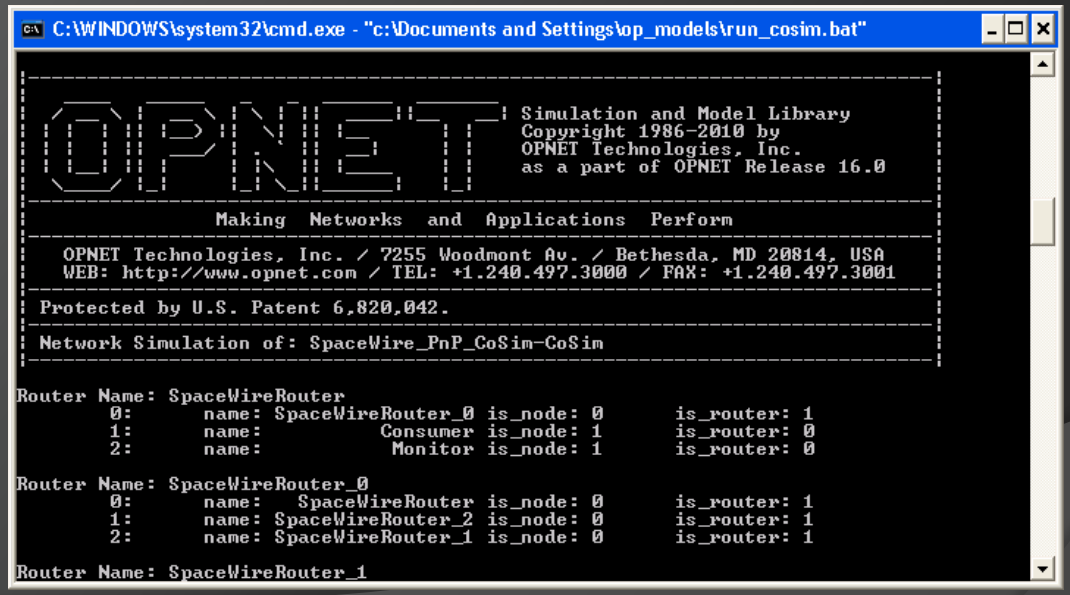


- OPNET SPA router
  - Simple model, no timing or flow control
  - Incoming messages use “path routing” addressing scheme
  - Uses path route to determine output port
  - Removes first hop of path route from the message



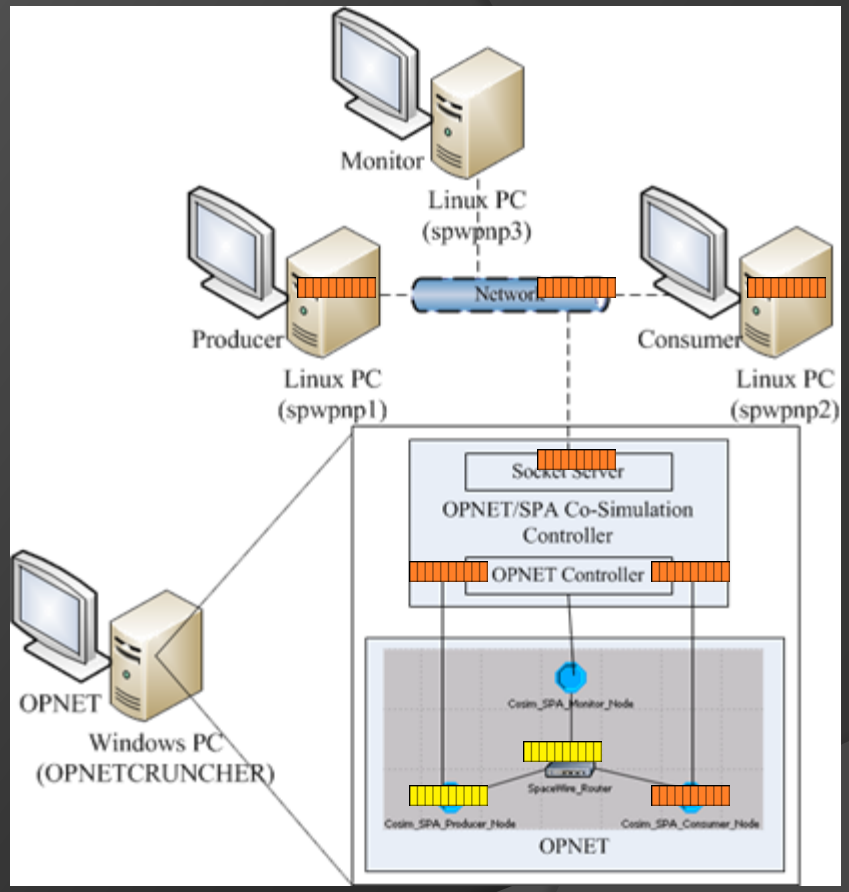
External Co-Simulation Controller Program

- Co-Simulation used to bridge SSM SPA nodes with OPNET Modeler simulation nodes
- External program running on the OPNET Modeler PC
- Launches the OPNET Simulation
  - "Normal" simulations are started from OPNET Modeler GUI
- Acts as a server for SSM SPA nodes
  - Uses sockets – select(), recv(), send()
- Uses OPNET's External Simulation Access (ESA) library to transmit messages to/from OPNET simulation



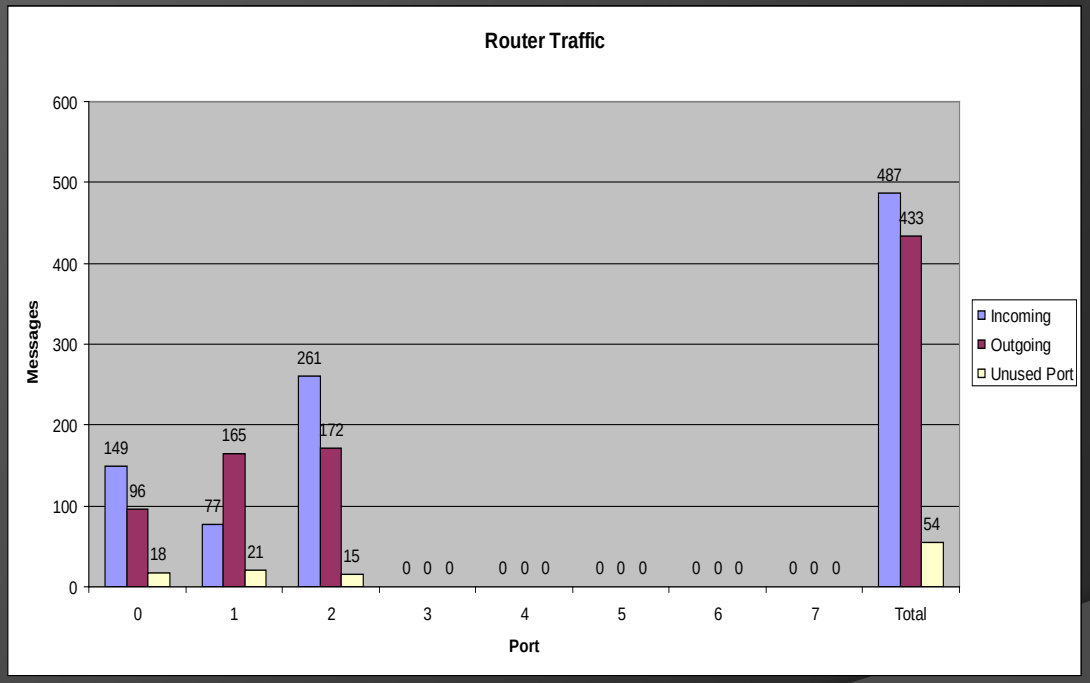
SSM SPA with OPNET Recap

- Trace message from SSM Producer to SSM Consumer
  - SSM SPA Producer node
  - Co-Simulation Controller
    - Socket server
    - ESA API
  - OPNET simulation
    - OPNET Producer node
    - OPNET SpaceWire route
    - OPNET Consumer node
  - Co-Simulation Controller
    - ESA API
    - socket server
  - SSM SPA Consumer node



Advantages of SPA/OPNET Integration

- No SpaceWire hardware required
- Uses actual SSM software
- Ability to test different network configurations
- Test for correctness of implementation
- Collection of statistics



Considered Alternatives

Alternative	Pros	Cons
OPNET Modeler: Build the “bridge” within the OPNET Simulated Network.	• No need for external program.	• No socket server. • Each node requires a direct socket connection and code modification. • High maintenance effort.
OPNET Modeler with SITL: Use the OPNET SITL add-on to communicate directly with the physical SpaceWire Cards.	• Process can be started within OPNET Modeler GUI. • In theory, approach can be combined with Co-Simulation.	• Requires a SpaceWire card per node in the network. • As per OPNET Technical Support, SITL is just Co-Simulation re-packaged for the purpose of connecting external hardware via Ethernet or Wireless.
OPNET Modeler with Shared Code Module: Use add-on to include SSM SPA Code within OPNET.	• SSM SPA would run within OPNET Modeler.	• Add-on not released, still in development. • No set release date. • Some code divergence. • Complexity issue.

Implemented Approach	Pros	Cons
OPNET Modeler with Co-Simulation:	• Easily configurable. • Low maintenance effort. • Uses unmodified SSM SPA code.	• Requires third application: Co-Simulation Controller.

Conclusion

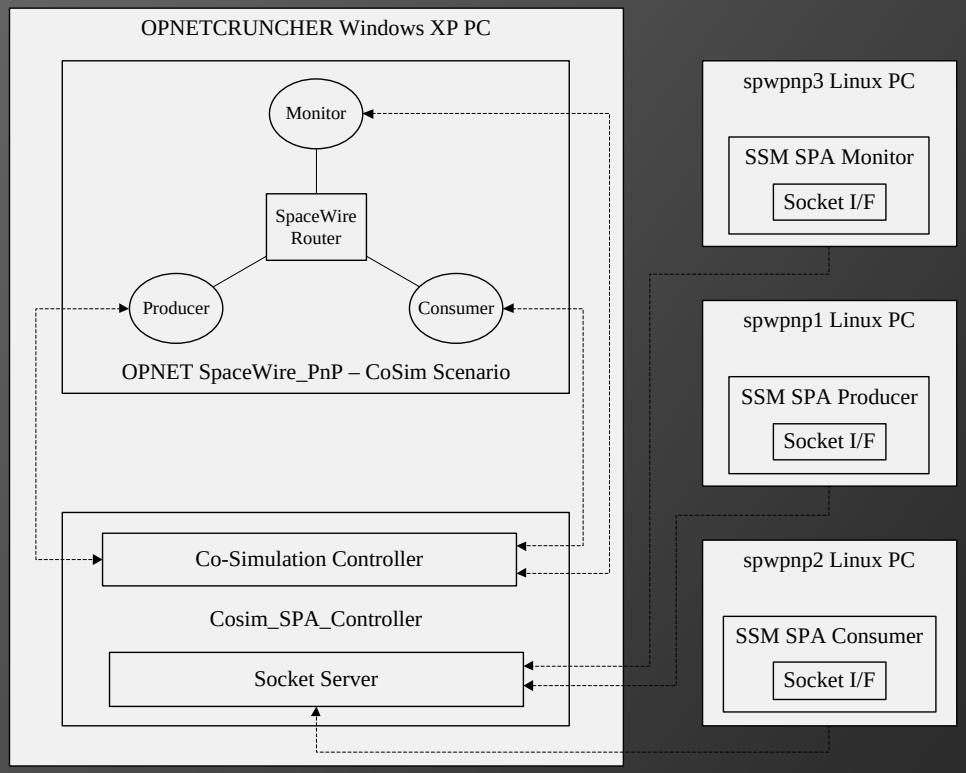
- Produced working configuration using Co-Simulation
- Ability to test unmodified SSM SPA software
- Does not require SpaceWire hardware
- Easily configurable
- Ability to acquire network traffic statistics
- Proof of concept
  - Funded by SwRI IR&D Project # 10.R8216

Questions/Comments

Acronyms

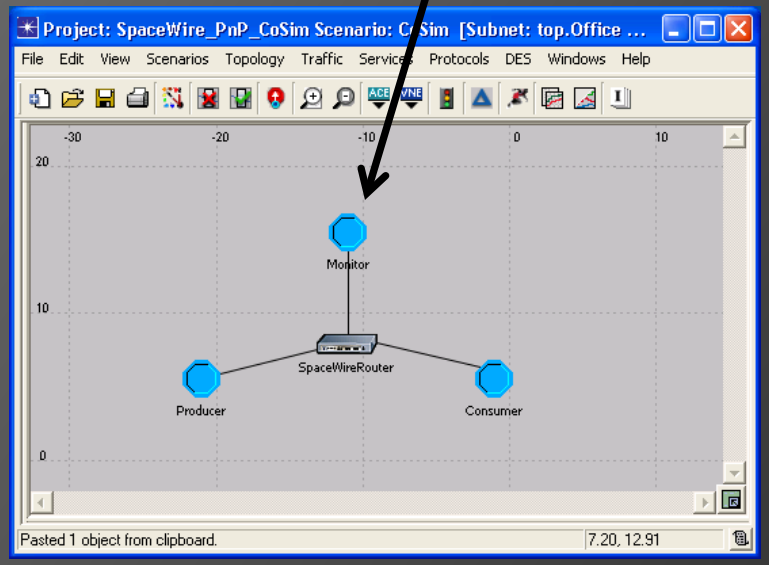
AFRL	Air Force Research Laboratory
ESA	External Simulation Access
PnP	Plug and Play
SITL	System-in-the-loop
SPA	Space PnP Avionics
SpW	SpaceWire
SSM	SPA Services Manager

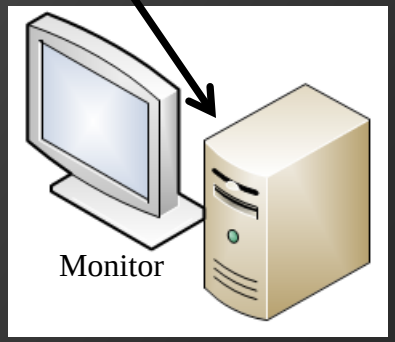
References


OPNETCRUNCHER Windows XP PC
Monitor
SpaceWire Router
Producer
Consumer
OPNET SpaceWire_PnP – CoSim Scenario
Co-Simulation Controller
Cosim_SPA_Controller
Socket Server
spwpnp3 Linux PC
SSM SPA Monitor
Socket I/F
spwpnp1 Linux PC
SSM SPA Producer
Socket I/F
spwpnp2 Linux PC
SSM SPA Consumer
Socket I/F

References

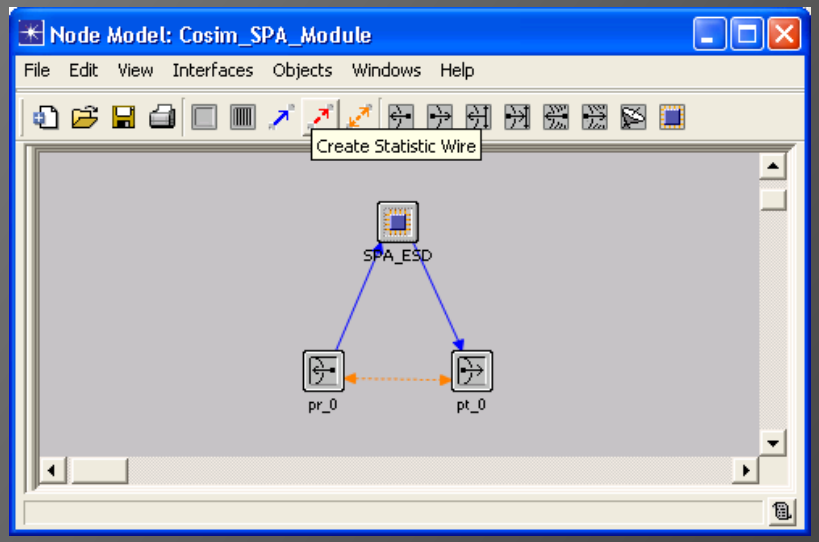
OPNET_Node_Name	Client_IP_Address
Consumer	129.162.108.133
Producer	129.162.108.50
Monitor	129.162.108.130

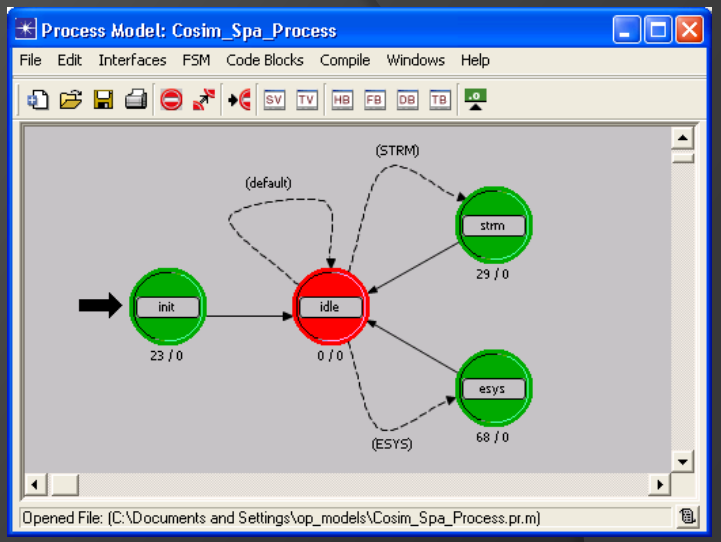


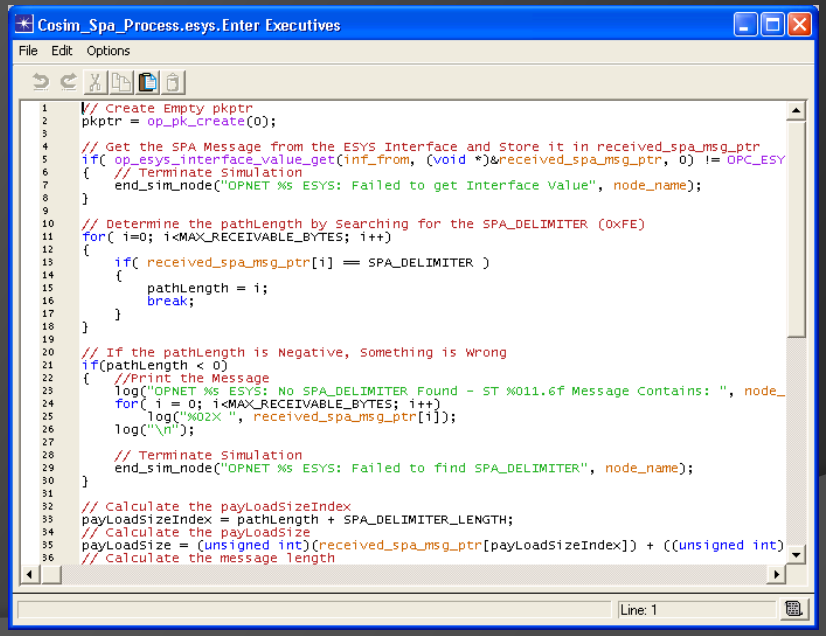


Linux PC
(spwpnp3)

References


Node Model: Cosim_SPA_Module
File Edit View Interfaces Objects Windows Help
Create Statistic Wire
SPA_ESD
pr_0
pt_0


Process Model: Cosim_Spa_Process
File Edit Interfaces FSM Code Blocks Compile Windows Help
(STRM)
(default)
strm
29 / 0
init
idle
23 / 0
0 / 0
esys
68 / 0
(ESYS)
Opened File: (C:\Documents and Settings\op_models\Cosim_Spa_Process.pr.m)


Cosim_Spa_Process.esys.Enter Executives
File Edit Options
// Create Empty pkptr
pkptr = op_pk_create(0);
// Get the SPA Message from the ESYS Interface and Store it in received_spa_msg_ptr
if(op_esys_interface_value_get(inf_from, (void *)&received_spa_msg_ptr, 0) != OPC_ESY
{ // Terminate Simulation
end_sim_node("OPNET %s ESYS: Failed to get Interface Value", node_name);
}
// Determine the pathLength by Searching for the SPA_DELIMITER (0xFE)
for(i=0; i<MAX_RECEIVABLE_BYTES; i++)
{
if(received_spa_msg_ptr[i] == SPA_DELIMITER)
{
pathLength = i;
break;
}
}
// If the pathLength is Negative, Something is Wrong
if(pathLength < 0)
{ //Print the Message
log("OPNET %s ESYS: No SPA_DELIMITER Found - ST %011.6f Message Contains: ", node_
for(i = 0; i<MAX_RECEIVABLE_BYTES; i++)
log("%02x ", received_spa_msg_ptr[i]);
log("\n");
// Terminate Simulation
end_sim_node("OPNET %s ESYS: Failed to find SPA_DELIMITER", node_name);
}
// Calculate the payLoadSizeIndex
payLoadSizeIndex = pathLength + SPA_DELIMITER_LENGTH;
// Calculate the payLoadSize
payLoadSize = (unsigned int)(received_spa_msg_ptr[payLoadSizeIndex]) + ((unsigned int)
// Calculate the message length
Line: 1