VISUALSIM TRAINING

Agenda- Part 5: Software and Networking Modeling

Software Modeling 578-586

Networking Modeling 587-609

Audio Video Bridging 610-621

TSN, Ethernet, Gateway 622-634

Integration 635-657

Version control- 658-664

Using Eclipse Debugger- 665-672

Use Cases 673-690

MIRABILIS design

Software Modeling

Defining Software Functionality

- At a statistical-level, a delay value for each function is sufficient to trigger the traffic on the bus and the memory devices.
- At the hardware-level, an application-specific instruction allocation called instruction-mix table provides an extremely accurate representation of a software task.
- Annotate performance-intensive portions of the code and generate instruction trace during execution. This last technique is good to test the architecture behavior for a benchmark or set of benchmark. This is also good to evaluate how a piece of code will behave in a multi-core environment.

Mapping Behavior to Architecture

- SystemResources
  - ✔ Mappers have cycles/time being fed to SystemResources
  - ✔ Build a hierarchical SystemResource for emulating RTOS + Processor
  - ✔ Extend SystemResource_Extend using the External_Port
- Computed time used as service time in Timed/Shared Queue
  - ✔ Queue + Server to emulate any processing resource
- Architecture Library
  - ✔ Use SoftwareMapper, Script or Input Port to trigger processing in hardware
  - ✔ Create hardware platform using Hardware blocks
- Using Script
  - ✔ Script has a Timed_Queue and wait for delays, Queue for action and Scheduler call

Modeling Abstraction- Software-Level

- Instruction Set Simulator provides the user the ability to load the Operating System and execute the compiled code. This is a good solution for early software debugging. But it is not a good solution while experimenting new architectures such as a new bus topology, different memory hierarchy, or processor clock speed sizing.
- At the hardware-level, an application-specific instruction allocation called instruction-mix table provides an extremely accurate representation of a software task.
- The application-specific instruction allocation technique is the most versatile and can be used for software testing, hardware verification and architecture optimization.
- Using instruction-mix table method of software emulation, the designer can view the depth of the pipeline identify the cause of a stall, power management algorithm impact, memory hierarchy operation, performance slowdown of load/store requests, and cache coherency algorithm quality. The simulation reports provide significant visibility into the architecture operation and allow for great optimization of the system throughput.

Instruction Mix Table

- Each software task or thread has a number of instructions and percentage of different types of instructions.
- In the case of My_Task_1, we have 10% of integer, 48% floating point, 10% logical, 7% load-store, and 25% brand instructions.
- This table is fed into a software generator block that generates the instruction sequence based on an intelligent algorithm.
- This sequence is used for the hardware testing, thus providing a more realistic test of the platform architecture.
- One can modify the task instruction mix and study the impact on your architecture by simply modifying the percentage table.

```
A_Task_Name      Num_Instr Type Pct   Type Pct   Type Pct   Type Pct   Type Pct   */

My_Task_1        500       INT  10    FP   48    LOG  10    IO   7     BRCH   25  ;
My_Task_2        500       INT  10    FP   28    LOG  10    IO   7     BRCH   45  ;
My_Task_3        500       INT  10    FP   48    LOG  10    IO   7     BRCH   25  ;
```

Instruction Mix Table for a Software Task

Modeling Software Blocks

Delays through the hardware platform

UML or Flow Chart model of the software with profiles

Generate instruction sequence

- Synthetic or profile-driven

Link code execution with hardware model execution

Modeling Results

- Software Tasks per Second (Min, Mean, StDev, Max)
- Software Deadlines Exceeded per Second (Min, Mean, StDev, Max)
- System Response Time vs. (Simulation Time, Histogram)
- System Throughput vs. (Simulation Time, Histogram)
- Hardware Efficiency (Utilization Summary)

- Software-based engineering discipline which involves
  - Modeling a system
  - Simulating and visualizing its behavior under real-world operating conditions
  - Refining its design through an iterative process
- To be truly effective, it must include
  - Task graph analysis
  - Mapping of behavior to architecture
  - Extremely accurate representation of a software task
  - Generating timing, power and behavior correctness

Solutions

Current solution

- Software is defined as a task graph, traffic, trace file and profile-based task generator
- Execute software on a device, FPGA or emulator

New solution

- Using GEM5 to create a architectural prototype

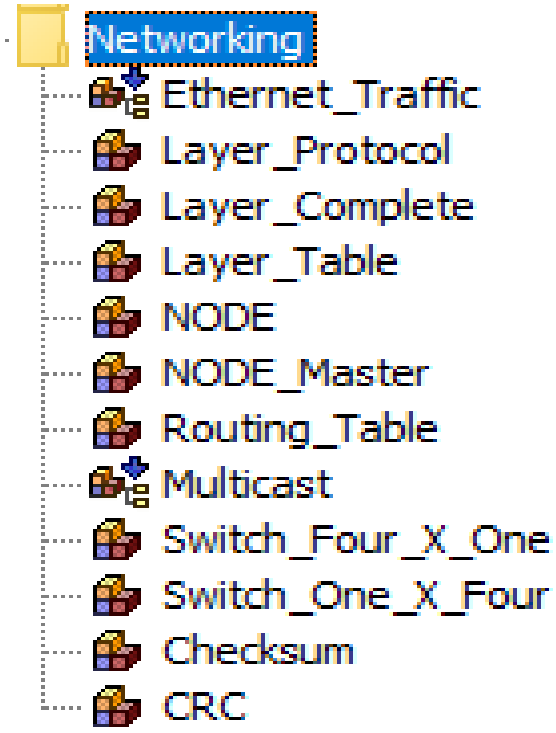
MIRABILIS design

Network Modeling

Overview of the Network Block Library

- Used to tune the parameters of a computer network, design the topology, develop new protocols and evaluate the application of a protocol for an application
- Library provides the infrastructure to handle the routing, Ethernet layer, fragmentation, retransmission, protocol delays and network delays
- Library also offers the user the ability construct custom protocols of a particular layer of the protocol and use the infrastructure to emulate the others
- Links can be connected or connection-less
- Multiple routing tables can exist in a single model

Network Library Location



Interfaces and Buses -> Networking

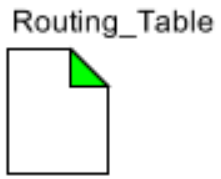
Fields Necessary for Network blocks

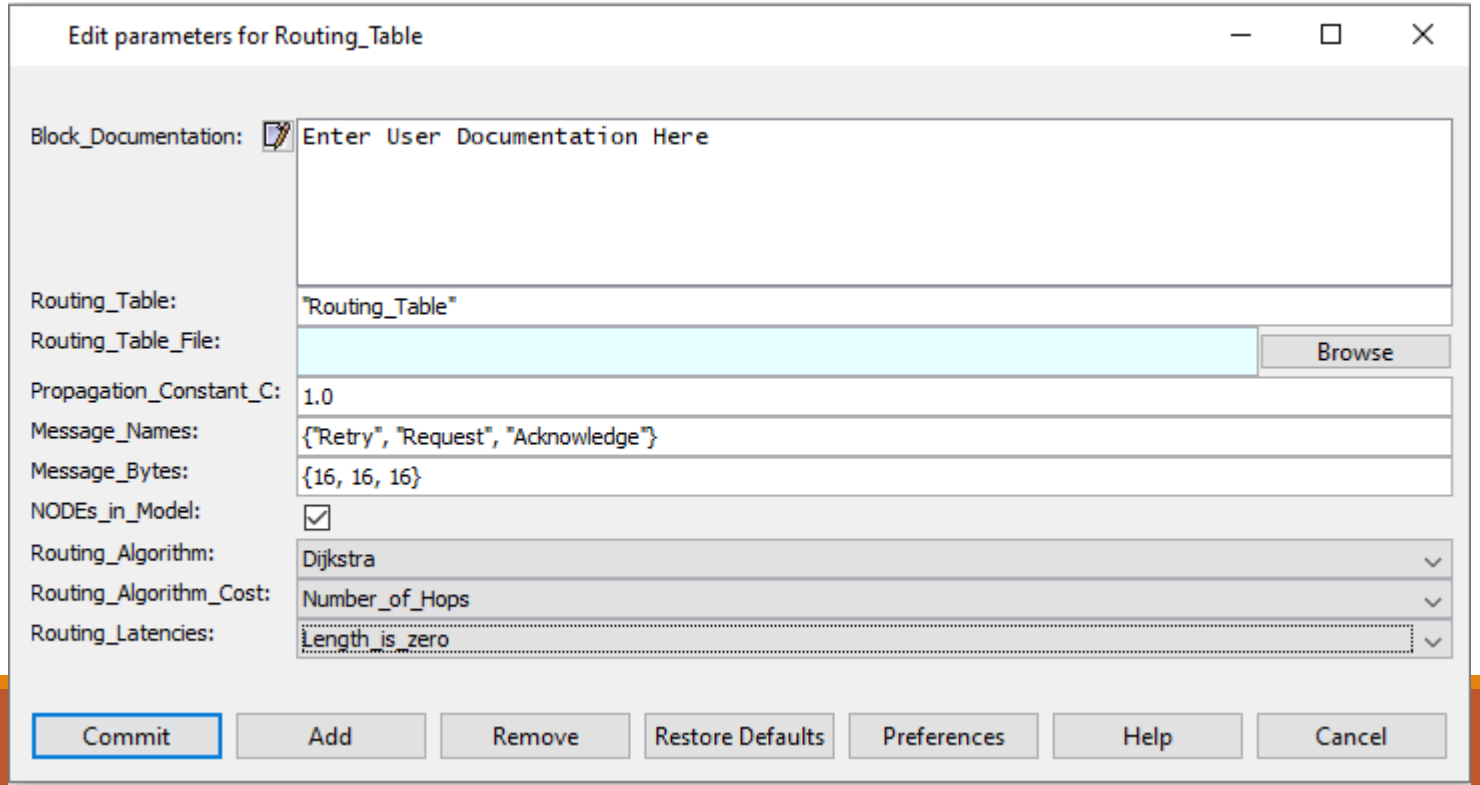
- Task_Source : Source
- Task_Destination : Destination
- Task_Size : Data Size
- Task_Layer : Overhead Size
- Task_Hop : Next Node or if going up/down, it lists this as Up.
- Task_Number : Unique number over the whole model
- Task_Trace : Array of all the Nodes that this transaction goes through

" Task_Class" DataStructure consists of the fields necessary for networking

Routing Table

- Provides information for the network
  - ✔ Routing_Algorithm, Routing_Algorithm_Cost Routing_Latencies, or Routing_Configuration
- Must be instantiated with Database



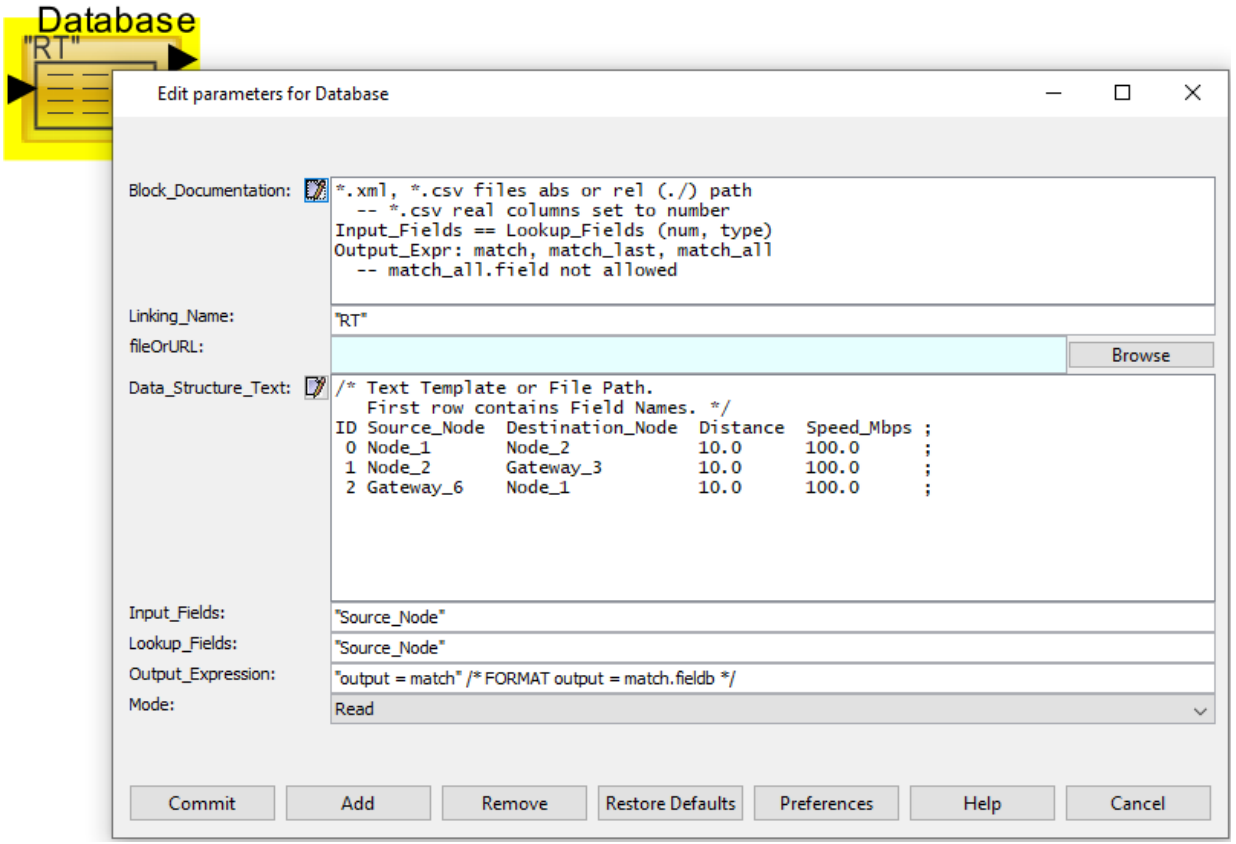


Routing_Table Block Parameters

- Routing_Table_Name: Routing table name must be unique
- Propagation_Constant_C: This is a multiple of C where C is the speed of light. This is used for computing the link delay based on Distance/(Propagation_Constant_C*C)
- Routing_Algorithm: The default routing algorithm is the Dykstra algorithm. User defined (i.e., custom) algorithms can be used as well.
- Routing_Algorithm_Cost: The type of cost function used in the determination of path
- Routing_Latencies: Must be renamed to Distance Units

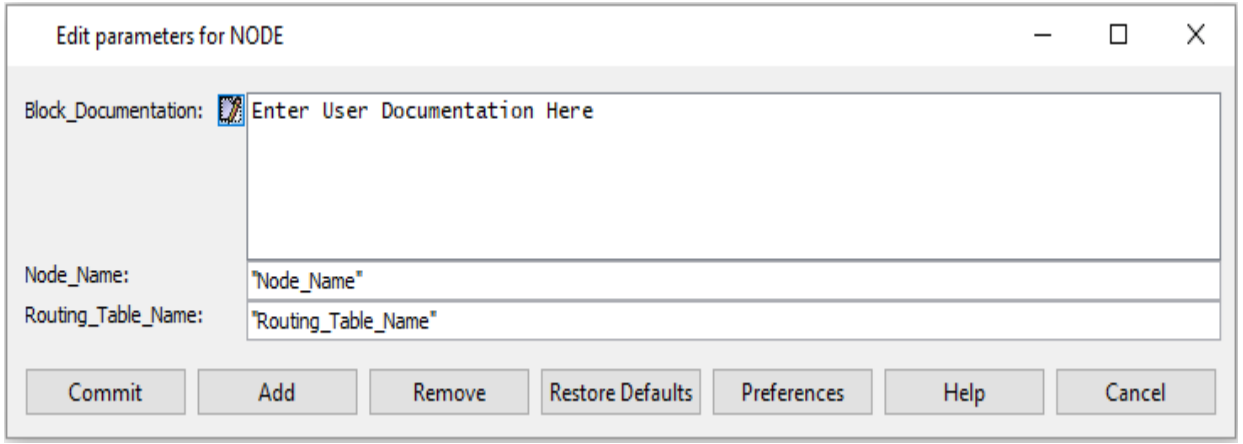
Database

- Routing Table is defined
- Must have the same name as that of the Routing Table Block



NODE Block

- Defines a basic Node within a large network
- Finds the next hop in the network using Routing Table
- Can operate in Two modes
  - ✔ Connected Routing Table Mode
  - ✔ Connectionless Routing Table Mode
- Used *Network Message* field to identify Retry and Drop
- Two delays-
  - ✔ Data Transfer on the link (Task_Size/Bandwidth)
  - ✔ Propagation Delay (Distance/(Speed of Light * Propagation Constant)



Operation

- The data can arrive at the Node from the Layer or from another node.
- When it arrives from another node, it checks the Network_Message == Retry or Drop_Packet. In that case, it checks whether the current node is the Source. If so, it sends it directly to the Layers. If it is not the Source, it sends out to the next Hop. It does not send it to the Layers.
- If the Node cannot find the Next Hop to the Destination, it sends the packet back to the Source Node.
- If it comes from the Layers and a path exists, it updates Task_Hop and then sends it to the next Node.
- If it came from another Node, it sends it to the Layer.
- If this is the Destination, it immediately sends it to the Layer.

Node Block Parameters

- Node Name: Name of this block. Required field and must be unique
- Routing Table Name: Name of the associated routing table

NODE- Statistics

- Generated using
  - ✔ RegEX
  - ✔ NODE master
- getBlockStatus("RT", "stats", 0) -> Returns subnet statistics for routing table domain

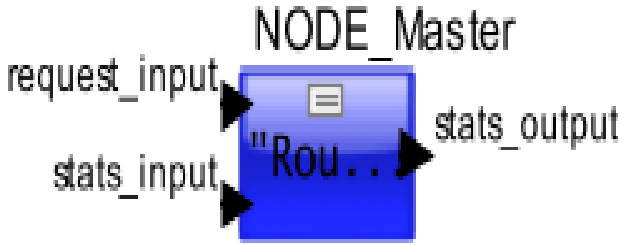
  getBlockStatus("RT", "stats", 1) -> Returns routing table for routing table domain

  getBlockStatus("RT", "stats", -1) ->Resets the routing table statistics

  getRoutingTableHop("RT","Node_1","Node_2") -> Returns the next node hop, if there is no hop, then this RegEx will return "none"

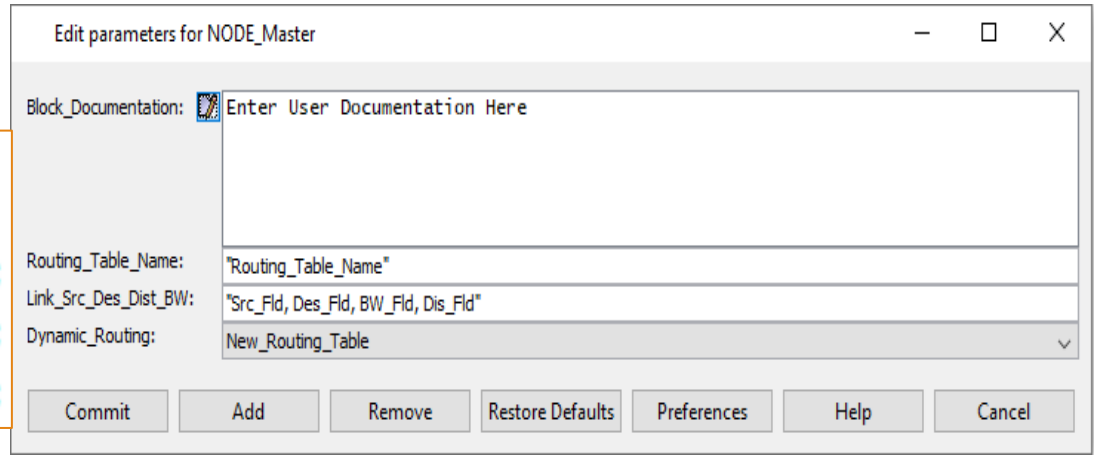
NODE Master

- Used to manipulate the operation of a network from a central location
  - ✔ Add Link
  - ✔ Remove Link
  - ✔ Recompute the Routing table
- Generates Statistics and current Routing table



DISPLAY AT TIME ------ 0.10 ns ------

Source,	Destination,	Hop,	Cost,	Meters,	Mbps
"Gateway_6",	"Node_1",	"Node_1",	1.0E-10,	10.0,	1.0E8
"Node_1",	"Node_2",	"Node_2",	1.0E-10,	10.0,	1.0E8
"Node_2",	"Gateway_3",	"Gateway_3",	1.0E-10,	10.0,	1.0E8

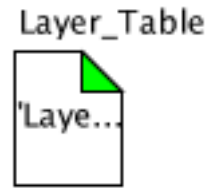


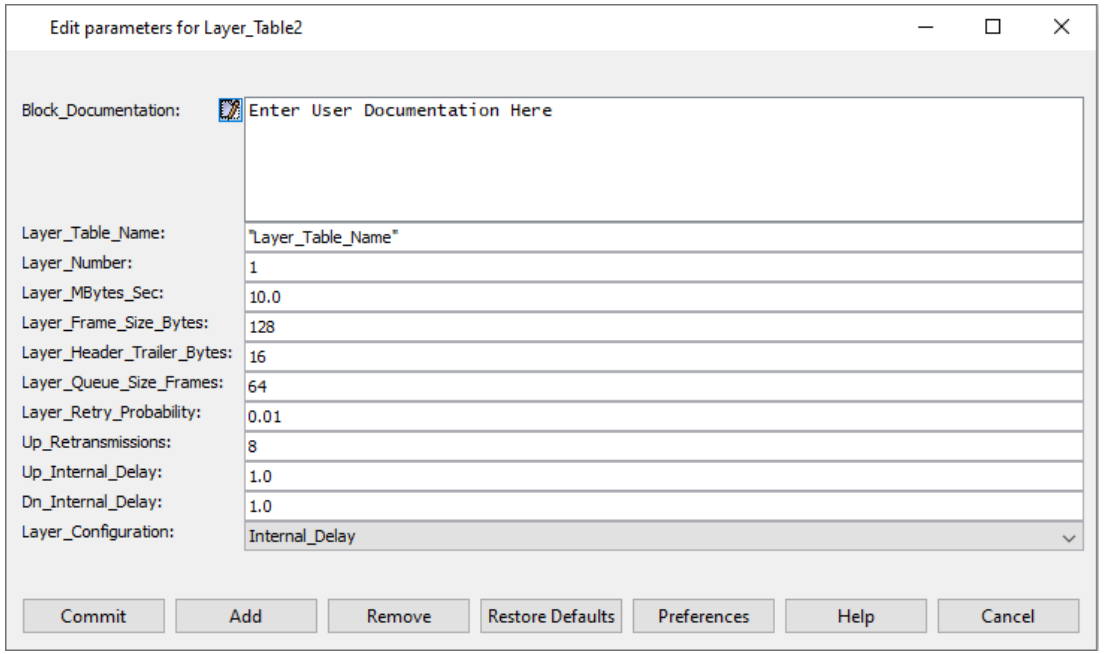
Node Master Block Parameters

- Routing Table Name: Name of the associated routing table
- Link_Src_dest_Dist_BW: Specifies where the block will get the link information
- Dynamic Routing: Specifies whether to add or remove a network link, or create a new routing table

Layer Table

- Defines the characteristics of Protocol Layer
  - ✔ Fragmentation
  - ✔ Latency
  - ✔ Queueing



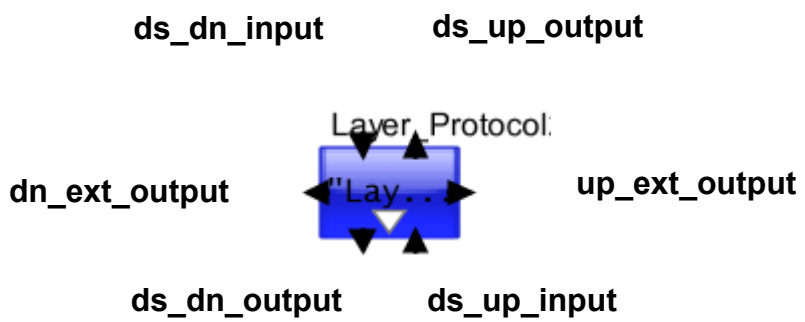


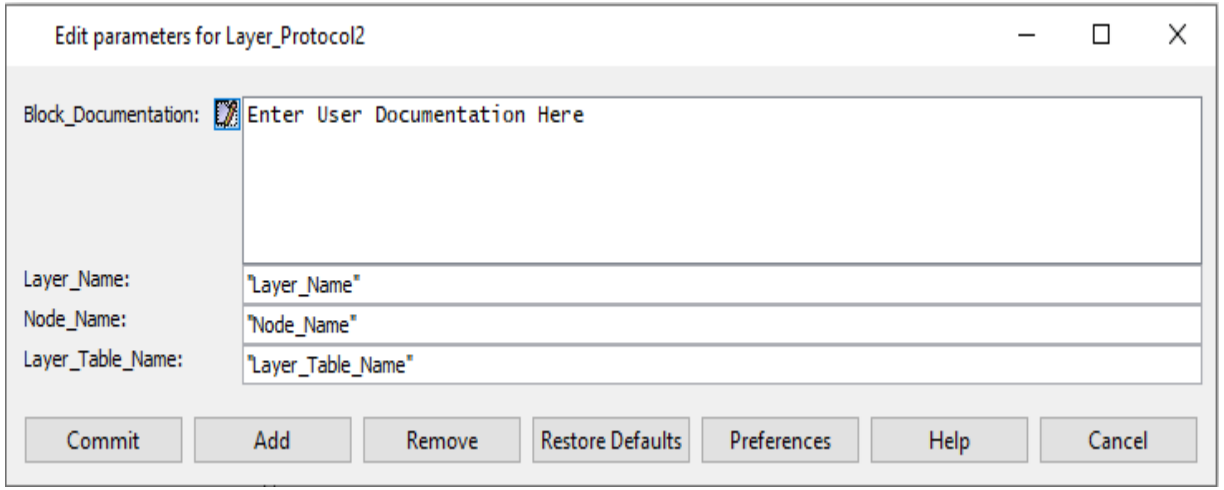
Layer Table Parameters

- Layer Table Name: Name of Layer, such as MAC (name must be unique)
- Layer Number: Number corresponding to layer, 1 through 7 valid entries. Used internally be retry mechanism.
- Layer MBytes_Sec: Speed. This is the layer throughput in the upward or downward direction in MBytes per second
- Layer_Frame_Size_Bytes: This is maximum frame size that can be transmitted in the upward or downward direction
- Layer_Header_Trailer_Bytes: Header/Trailer Bytes for Layer_Frame_Size_Bytes
- Layer_Queue_Size_Frames: Queue length of upward or downward flow. This length equates to sessions.

Layer Protocol

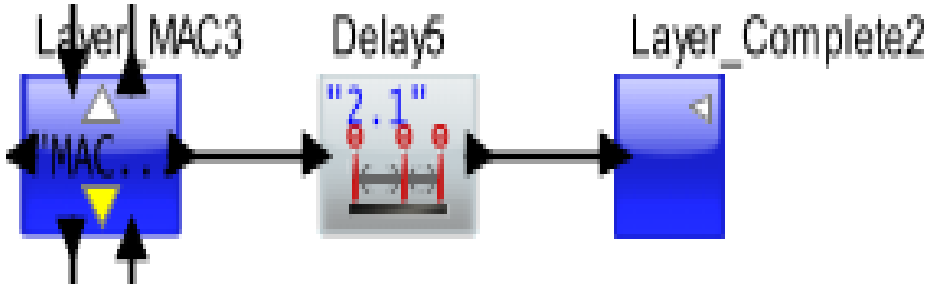
- Used to define each layer of Network Protocol stack
- Each Layer_Protocol must reference one Layer_Table block
- A layer block can add/remove the necessary overhead bytes for header and trailers, delay the block for the processing time, queue, force a retry, and add custom logic and timing
- Two delays
  - ✔ Data Transfer delay
  - ✔ Processing Delay (internal or external)





External processing

- Layer Configuration parameter must be set to External Delay in Layer Table
- Sends the data structure to the ports called 'up_ext_output' (going up the stack) and 'dn_ext_output' (going down the stack) to implement the external delay
- External process must be terminated with a Layer_Complete block, which returns the packet to the Layer_Protocol block to resume either up or down the layer stack
- Internal Delays are ignored



Statistics for Layer Protocol

- Generate statistics using the RegEX function
  - ✔ getBlockStatus("MAC_1","Any Value", "stats", 1,"Any Integer") - stats
  - ✔ getBlockStatus("MAC_1","Any Value", "stats", -1,"Any Integer") - reset stats
  - ✔ getBlockStatus("MAC_1","Any Value", "length", 1,Any Integer) – up queue length
  - ✔ getBlockStatus("MAC_1","Any Value", "length", 2,Any Integer) –down queue length

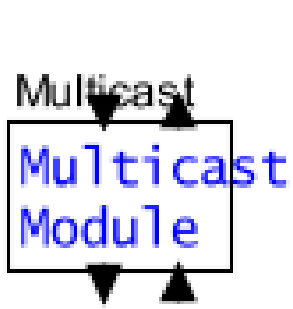
```
DISPLAY AT TIME                ------ 4999.99999999990 sec --
{A_Layer                     = "IP2",
A_Layer_Table                = "LT2",
BLOCK                        = "Layer_IP2",
DELTA                        = 0.0,
DS_NAME                      = "Layer_Stats",
Dn_MBps                      = 0.5200006032,
Dn_Max_Delay                 = 1.00000116,
Dn_Max_Occupancy             = 1.0,
Dn_Mean_Delay                = 1.00000116,
Dn_Mean_Occupancy            = 0.5,
Dn_Min_Delay                 = 1.00000116,
Dn_Min_Occupancy             = 0.0,
Dn_Number_Entered            = 26,
Dn_Number_Exited             = 26,
Dn_StDev_Delay               = 0.0,
Dn_StDev_Occupancy           = 0.5,
Dn_Utilization               = 0.5200006032,
ID                           = 1,
INDEX                        = 0,
TIME                         = 4999.9999999999,
Up_MBps                      = 0.52000052,
Up_Max_Delay                 = 1.000001,
Up_Max_Occupancy             = 1.0,
Up_Mean_Delay                = 1.000001,
Up_Mean_Occupancy            = 0.5,
Up_Min_Delay                 = 1.000001,
Up_Min_Occupancy             = 0.0,
Up_Number_Entered            = 26,
Up_Number_Exited             = 26,
Up_StDev_Delay               = 0.0,
Up_StDev_Occupancy           = 0.5,
Up_Utilization               = 0.52000052}
```

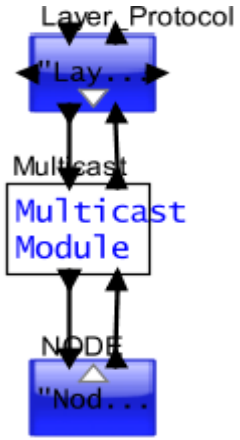
Multicast

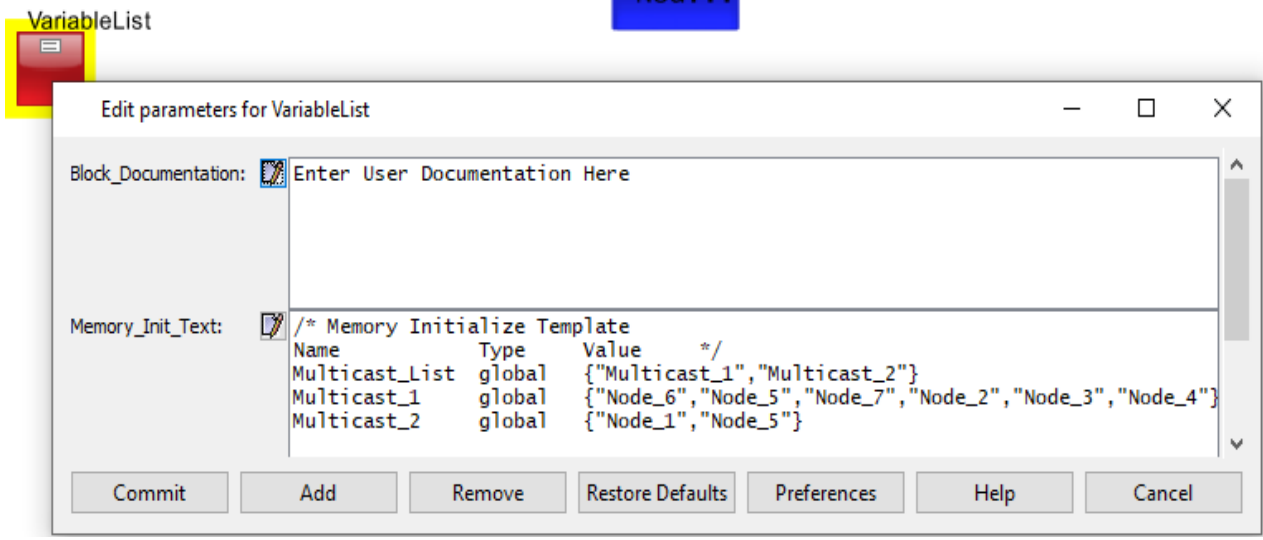
- Simulates Internet Multicast Protocol. Performs Multicast and Broadcast
- Spanning Tree algorithm for routing of packets
- The Signal that has to be multicasted must have the Network_Message field with the name of the multicast

  Network_Message = "Multicast_1"

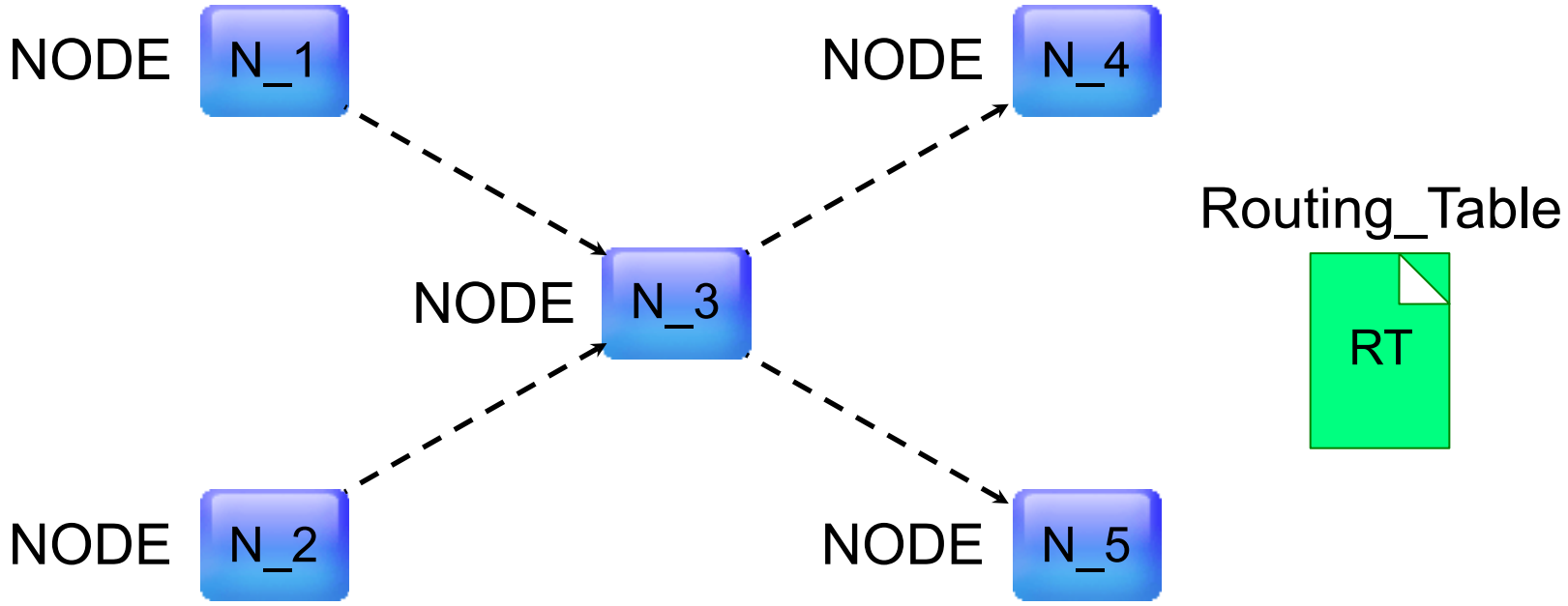
- Configurations for Multicast must be done in Variable List Block



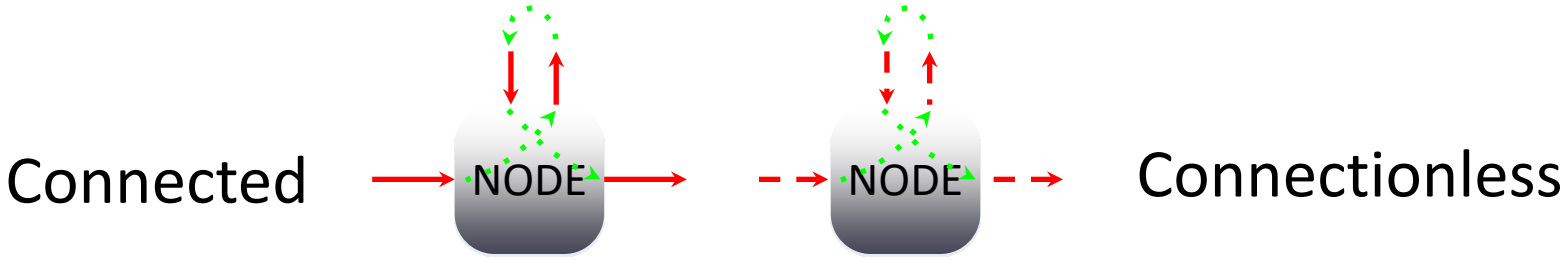




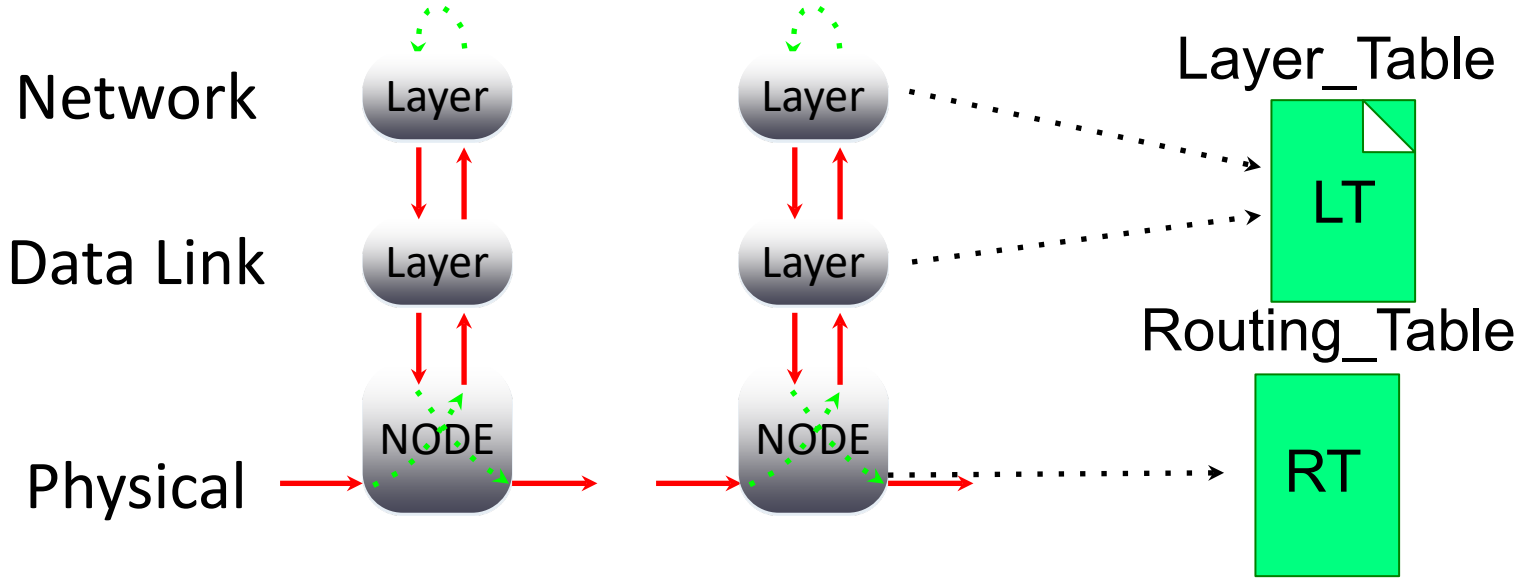
Networking Nodes



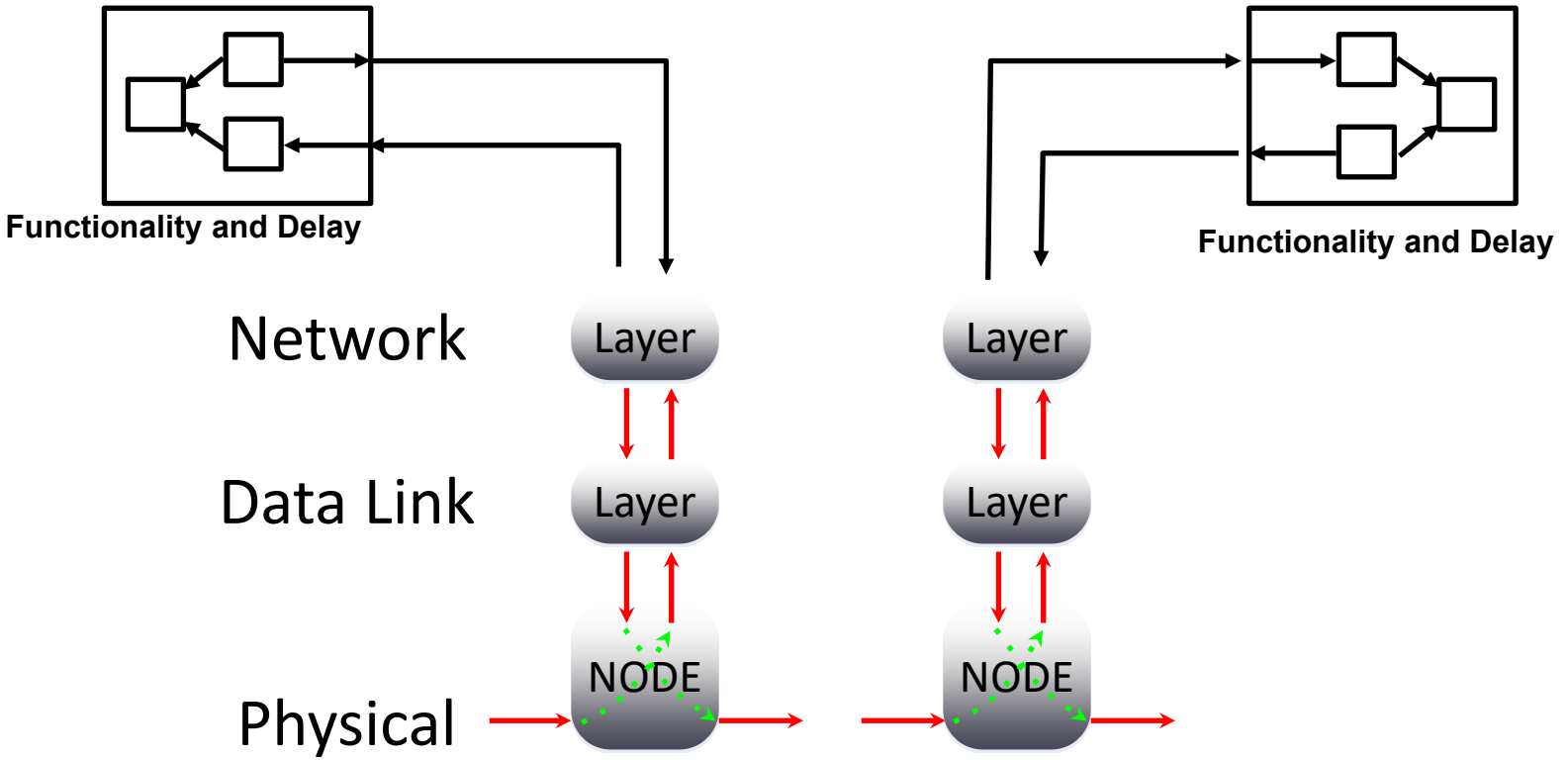
Connected and Connectionless Nodes



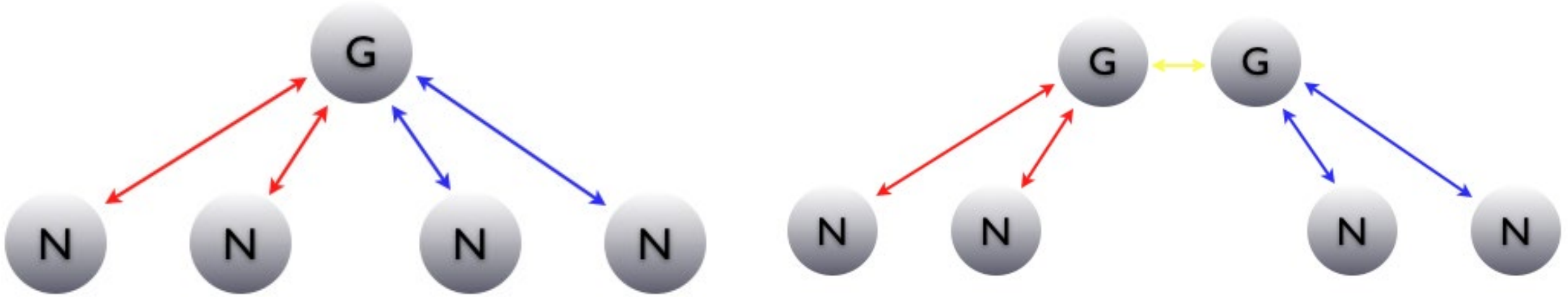
Network Node Layers



Network Node Layers (continued)



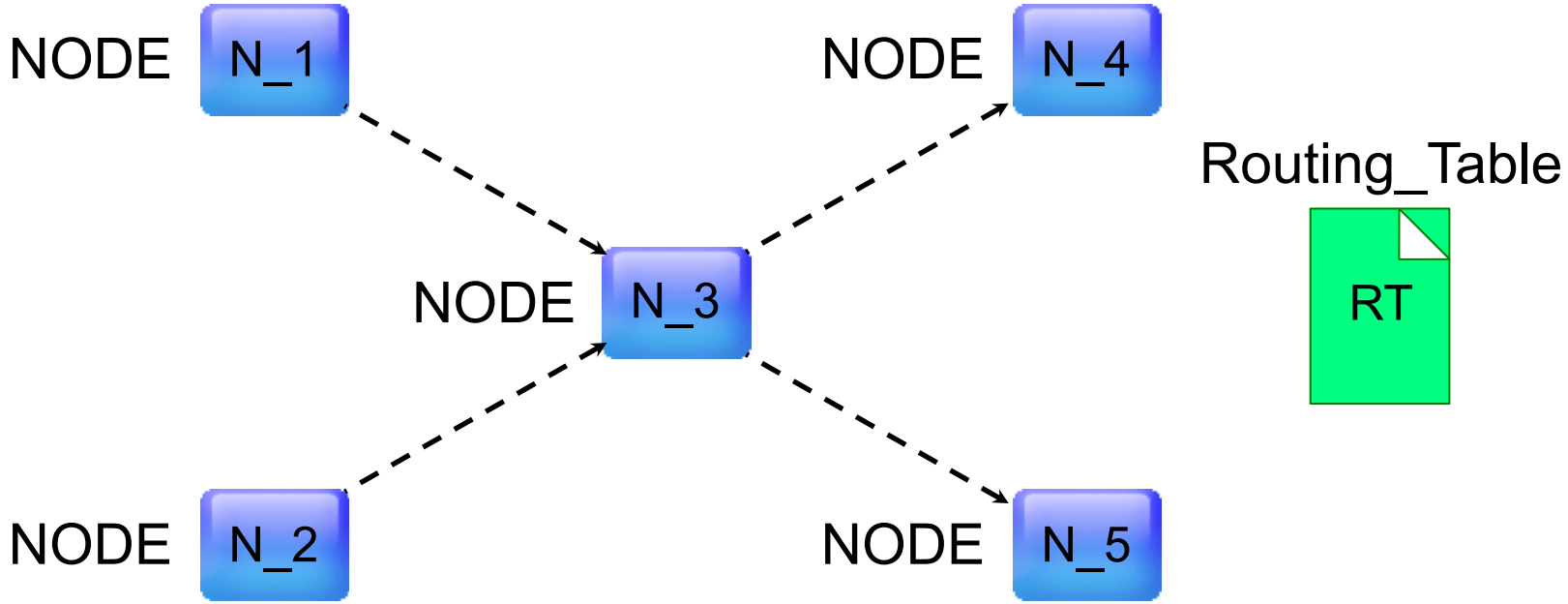
Network Node Layers (continued)



Routing Algorithms

Shortest Path First -- Dijkstra Algorithm

- Static Routing



MIRABILIS design

Networking Library
Audio Video Bridging

Audio Video Bridging Library

- Library of components that emulates the AVB operation at the Talker, Bridge and Listener locations
- Works in conjunction with the existing Networking library
- Provides traffic generator, protocol additions, and statistics reporters
- Tested to meet the specification and experimental data
- Easily extendable for future enhancements

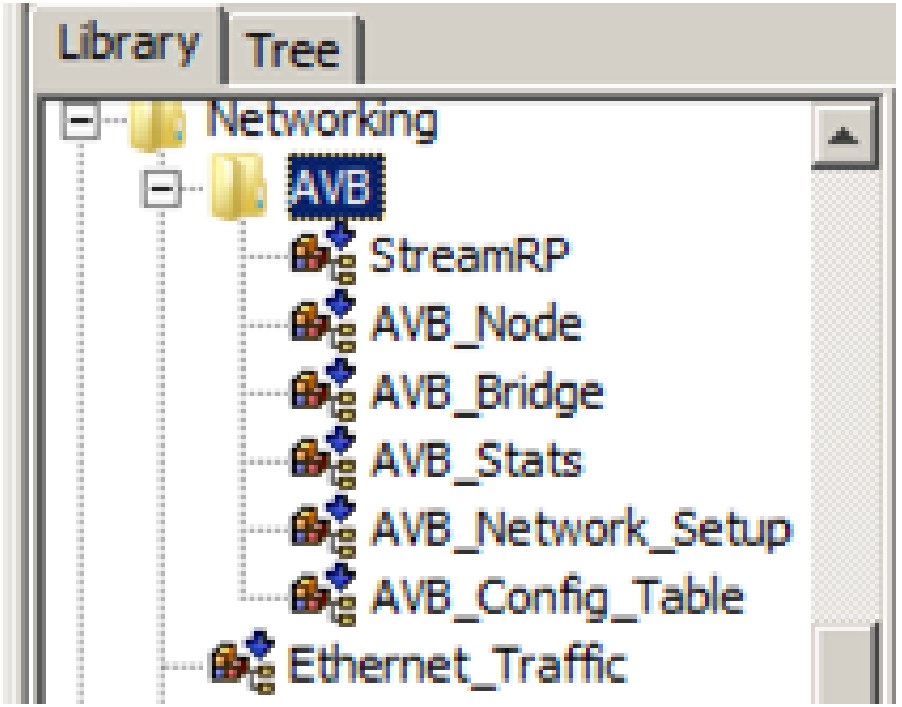
Audio-Video Bridging- Standards Supported

- IEEE 802.1AS: Timing and Synchronization for Time-Sensitive Applications (gPTP),
- IEEE 802.1Qat: Stream Reservation Protocol (SRP),
- IEEE 802.1Qav: Forwarding and Queuing for Time-Sensitive Streams (FQTSS), and
- IEEE 802.1BA: Audio Video Bridging Systems

AVB Library Usage

- Assemble a complete end-to-end automotive applications with multiple sub-systems, ECU hardware, cameras and other devices connected via AVB over Ethernet
  - ✔ Determine the network and the hardware configurations required to meet the latency, throughput and power requirements
- Assemble a network of recording equipment, displays, projectors and other audio/video equipment in a professional studio or concert hall.
  - ✔ Configure the network architecture to ensure low-latency and synchronized streaming operation

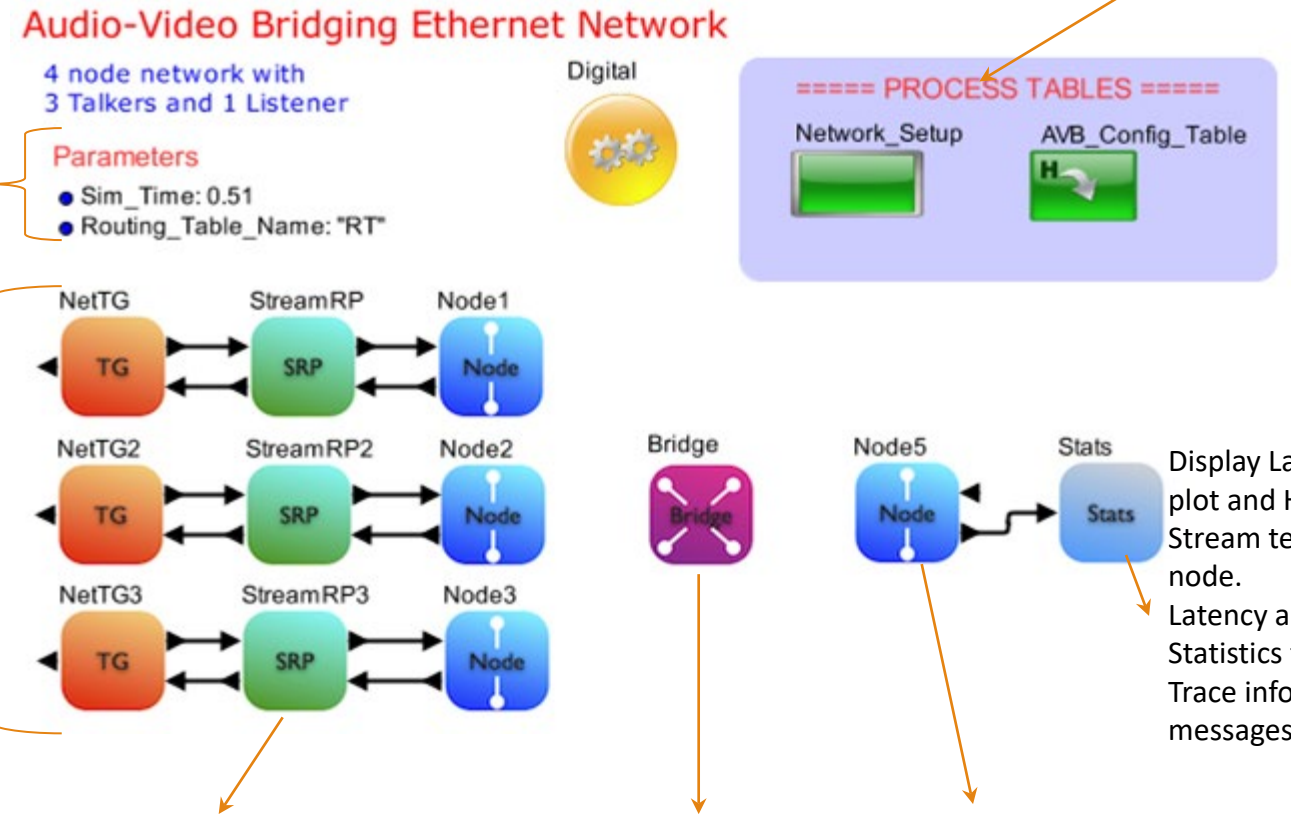
AVB Library



Using AVB Blocks- Rules to be Followed

- All rules of the Network Node blocks apply here
- AVB_Config_Tables and AVB_Setup are required blocks for all AVB Models
- AVB_Config_Table contains the Routing_Table block and the Link_Setup blocks. One set is sufficient
- If using Ethernet_Traffic block to generate Ethernet traffic, then the Traffic Table is sufficient. The Stream block is not required.
- Each Ethernet_Traffic block must have a unique Traffic table
- Each AVB stream must have a unique ID in the stream table.
- If using AVB streams, then all the blocks in the AVB_Config_Table are required.
- Bandwidth assigned to all Type classes on a link should not exceed link bandwidth
- All links at a bridge have the same bandwidth assignment for the Type classes

AVB Library Example



Required Blocks to configure the network, bandwidth allocation. AVB attributes and traffic

Top-level parameters to ensure all blocks Use same names and Simulation end time

Generates both AVB and Ethernet Traffic

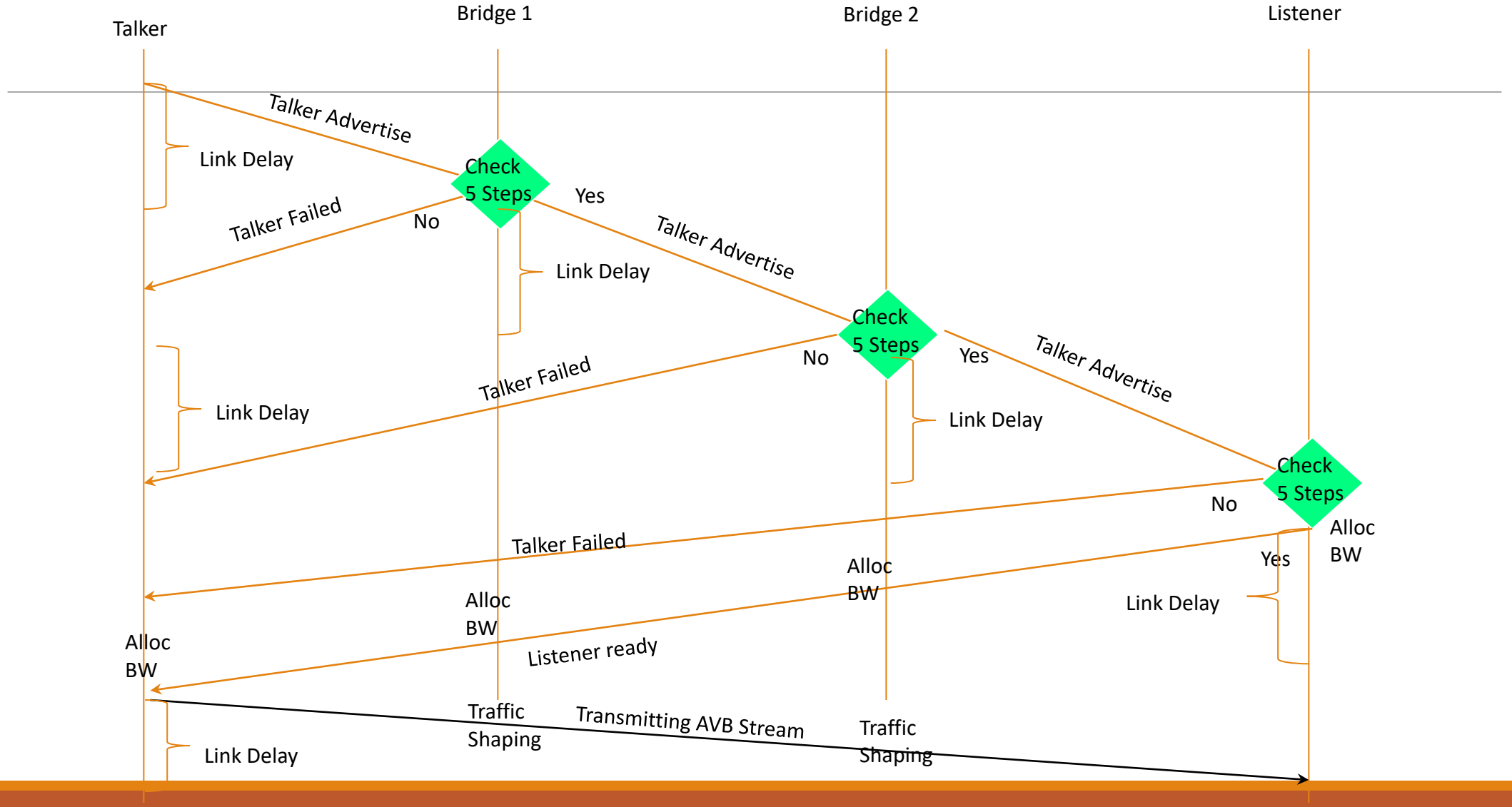
Display Latency as a Timed plot and Histogram per Stream terminating at this node.
Latency and Throughput Statistics for all the Streams
Trace information for the messages and transfers

Manages the
- Generation of Talker Advertise
- Handles algorithm to respond to Talker Failure
- Generate Deregister messages

Connects the Nodes in a network and the routing between the Nodes

Ethernet, Traffic Shaping and Clock Sync Messages

AVB Flow Diagram- Stream Reservation Procedure



Ethernet Traffic Shaping Algorithm

- Uses Leaky Bucket for AVB streams
- Bandwidth is for a fixed time period of 100 frames of 1500 bytes each
- Bandwidth credit assigned to each Type as a percentage of this period
- Each successful AVB stream is assigned bandwidth as a percentage of the Type bandwidth
- After the end of this period, bandwidth credit reset for all the types

Ethernet and AVB- Traffic Shaping

- Requires the Stream, Type_to_BW and Class_to_Type tables
- Priority is higher for the higher number
- Queue for each Type
- Unassigned bandwidth kept in Type 8
- Period duration for ensuring bandwidth is 100 frames of 1500 bytes or 150,000 bytes transfer time
  - For a 100 Mbps, this is 12ms and for 1Gbps it is 1.2 ms
- Bandwidth allocated is reset at the end of the period.

AVB and Ethernet- Traffic Shaping Algorithm

- Starts with the highest Type with assigned bandwidth
- If bandwidth is available, a packet will be transmitted, even if the credit goes to negative
- If Packet Available, packet selection
  - ✔ If Class A or B, then one of the AVB streams in the queue is sent out first
  - ✔ If there is no AVB or it is not a Class A or B, then the head of the queue for that type is sent out
- If packet not available
  - ✔ The scheduler does a best effort
  - ✔ First it searches for a Class A and then a Class B AVB packet
  - ✔ If no AVB is available, it goes through from 7 to 0
  - ✔ To ensure fairness, the next time, the sequence will start from 6-0,7 and so on
  - ✔ Credit is not decremented in this case
- When packet is sent out, the scheduler moves to the next lower Type
- When Type 0 is complete, the scheduler goes to Type 8. This goes Type 7 to 0. The next time, the Scheduler starts from next lower one.
- When all credit has expired, the credit are reset for all the Types

Stream Reservation checks and Failure Codes

- AVB allocated bandwidth exceeds the threshold for the Class = 1
- Worst case Execution Time (WCET) is greater than 2ms for Class A and 50 ms for Class B = 2
- Next Bridge has a different type for the Class A or B = 3
- Optional check where the listener has not buffer capacity = 4 (Currently not used)
- Maximum number of Hops Exceeds 7 = 5

AVB_Config_Tables

- Link Setup- Associated with the Node and Routing activities
- Routing Table- Required for Routing between Network Nodes
- Traffic Table- Requires one per Traffic block in the model
- Stream- Required if AVB stream exists in the model
- Type_to_BW- Bandwidth allocation by type for Nodes and Bridges
- Class_to_Type- Class A and B assignment to a Type for Nodes and Bridges

MIRABILIS design

Networking Library
TSN, Gateway, Ethernet
Semiconductor Device

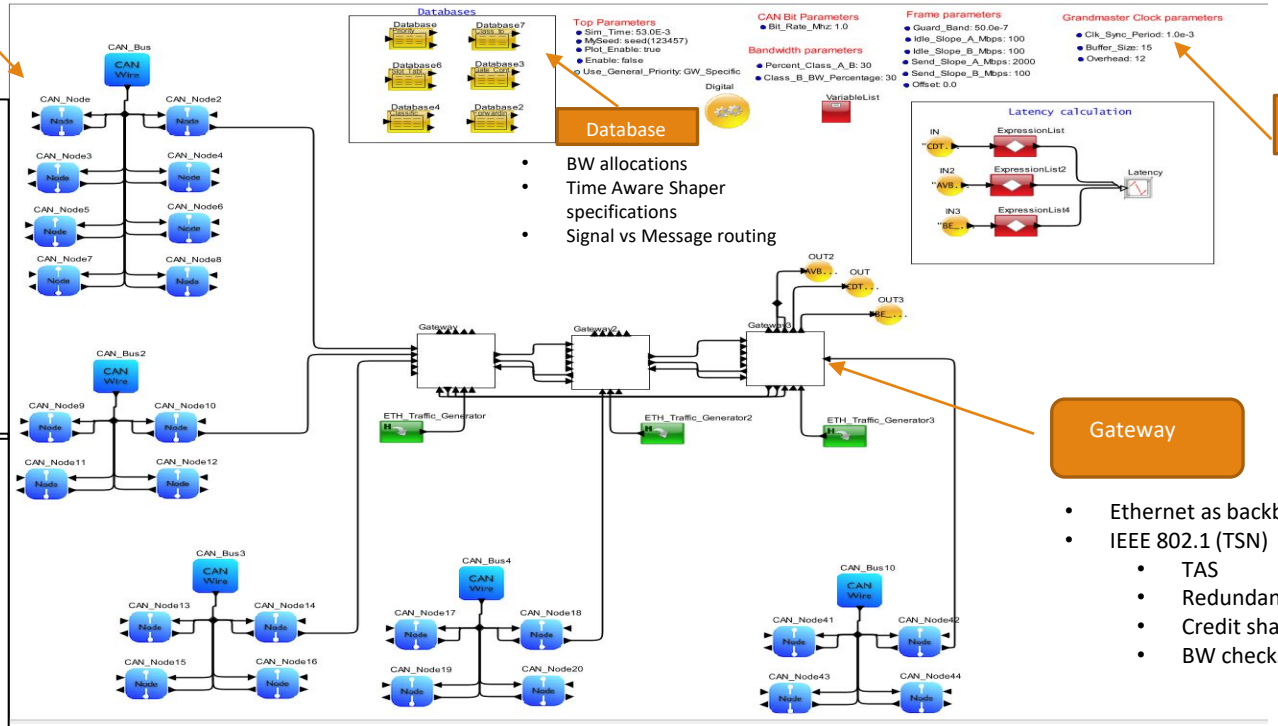
Automotive Network containing TSN Switch, Gateway and CAN Buses

CAN Bus

- 11 bit and 29 bit
- Power calc
- Bus off
- Data Frame, Remote frame, Overload Frame , Error Frame

Entire end to end system can be assembled in a go

- Provides the support for TCP/IP or UDP
- User can configure each Layer as per their requirement



Database

- BW allocations
- Time Aware Shaper specifications
- Signal vs Message routing

Parameters

- Each Linked to a feature
- Ease of evaluation

Gateway

- Ethernet as backbone
- IEEE 802.1 (TSN)
  - TAS
  - Redundancy
  - Credit shaper
  - BW checks

Library provides various types of schedulers

- IEEE 802.1Q compliant
- WFQ
- WRR
- DRR
- Strict Priority
- FCFS
- RR
- RR Priority

Standards supported Automotive library

TSN

IEEE 802.1Qbv

IEEE 802.1Qbu

IEEE 802.3br

IEEE 802.1Qca

IEEE 802.1Qcc

IEEE 802.1Qci

IEEE 802.1QCB

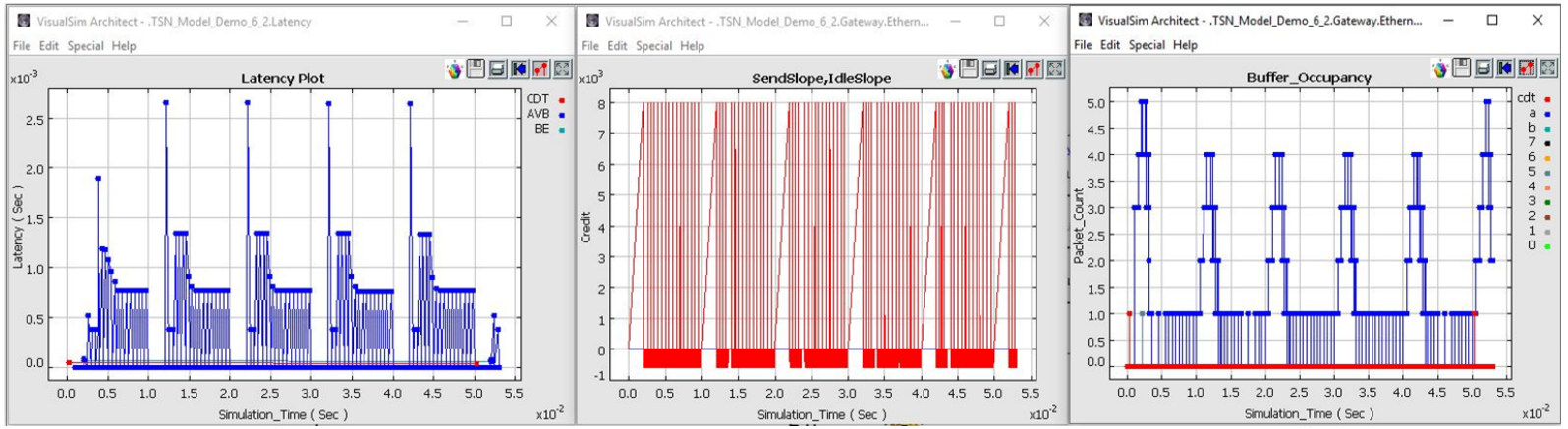
IEEE 802.1Qch

IEEE 802.1AS

Bus standards

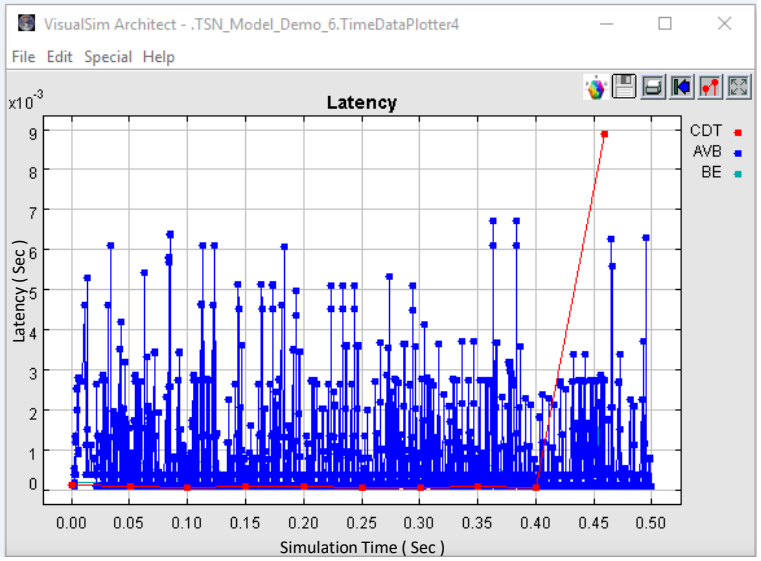
- CAN A and CAN B
  - Data Frame
  - Remote Frame
  - Overload Frame
  - Error Frame
  - BusOff
  - Manual/Automatic Restart
  - Power Calculation
  - Filtering
  - Fast Data rate

TSN Stats Generated



Evaluation of an Error in the TSN Scheduler

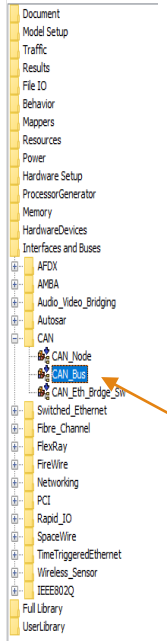
Evaluation on BW,MIF,TAS,CBS gives us idea on what could happen with a worst case scenario



Latency for CDT spiked

- CDT frame misses the time slot

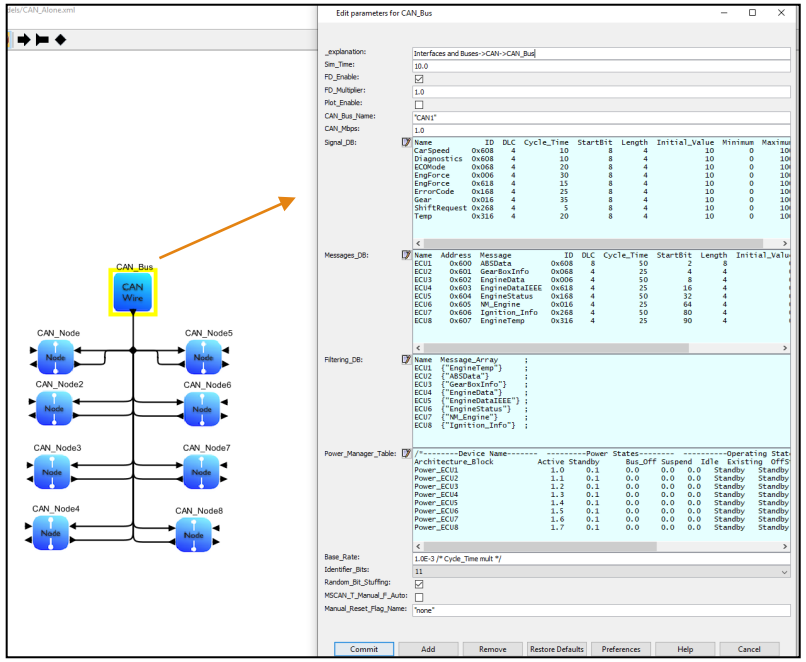
CAN Bus



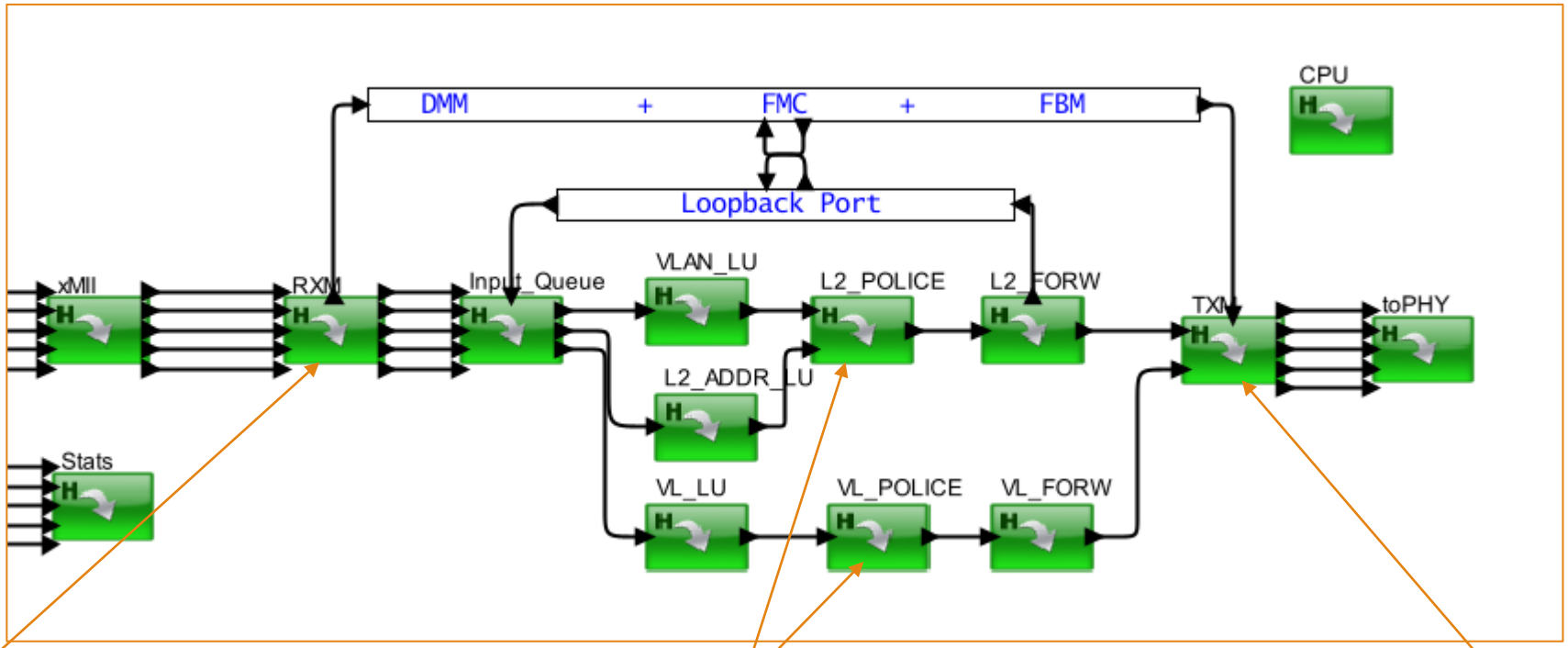
- Different parameters for toggling between functionalities
- Drop down menu for selecting 11 bit or 29 bit
- Filtering DB can be modified to select required messages
- Manual or Automatic restart can be selected just by checking the box

CAN Bus Modules can be accessed simply by going into the CAN folder

We just have to drag and drop them on the window



Ethernet Switch – Semi abstract



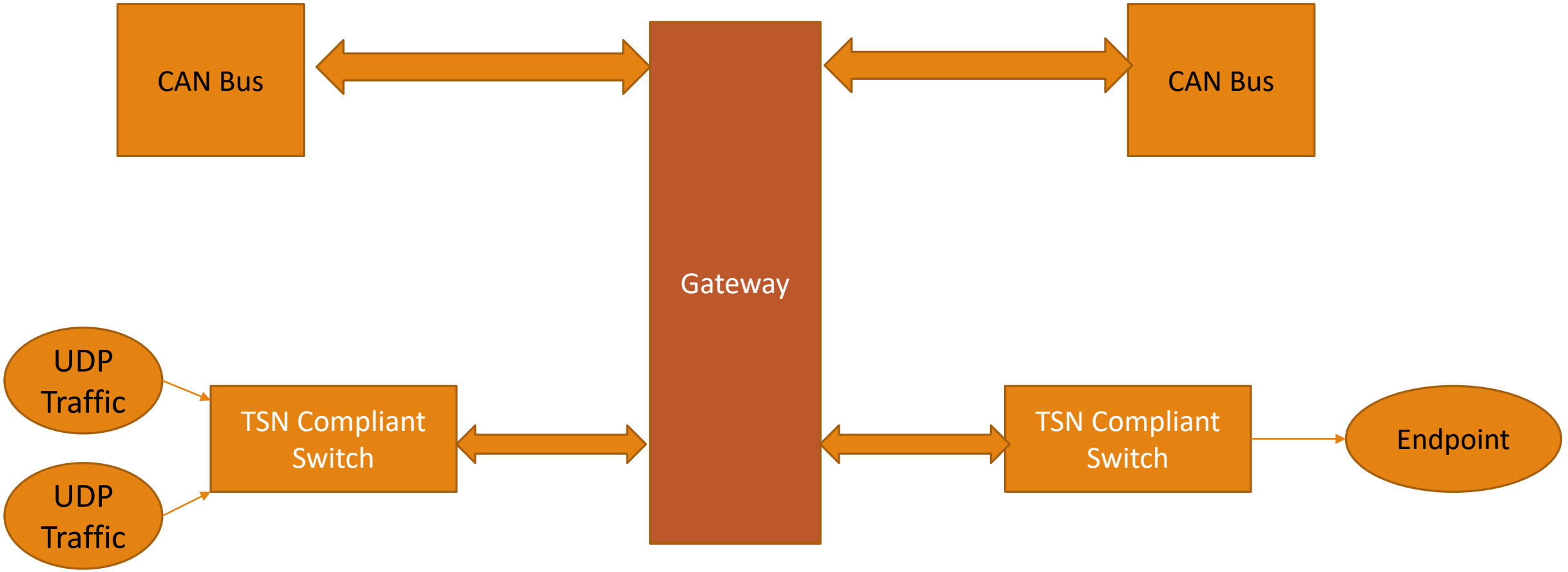
Store and Forward or Cut through can be implemented here

Different Policing schemes of interest can be implemented here

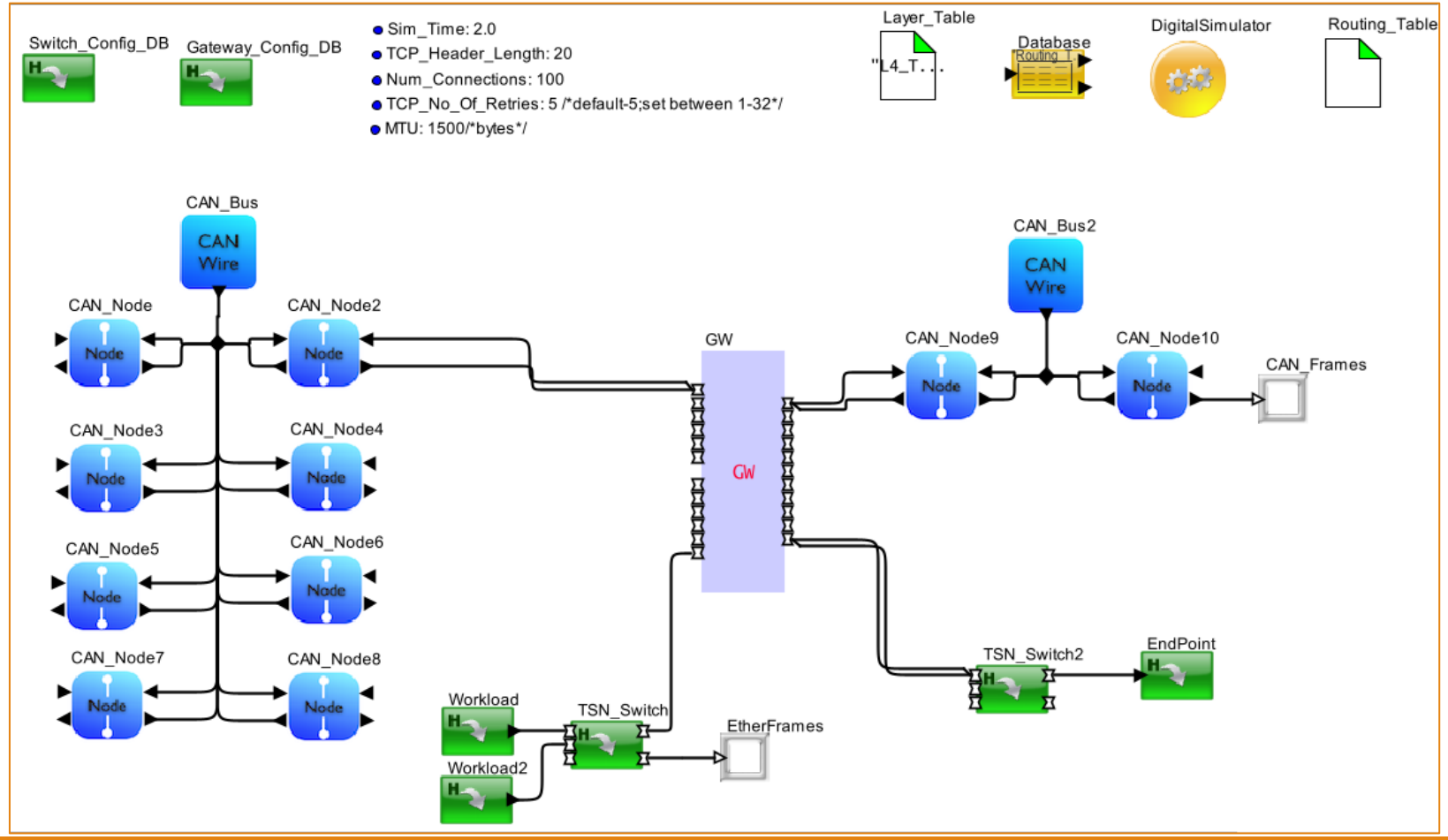
Different shapers of interest , Different Scheduling algorithm like WRR can be selected here

MIRABILIS design

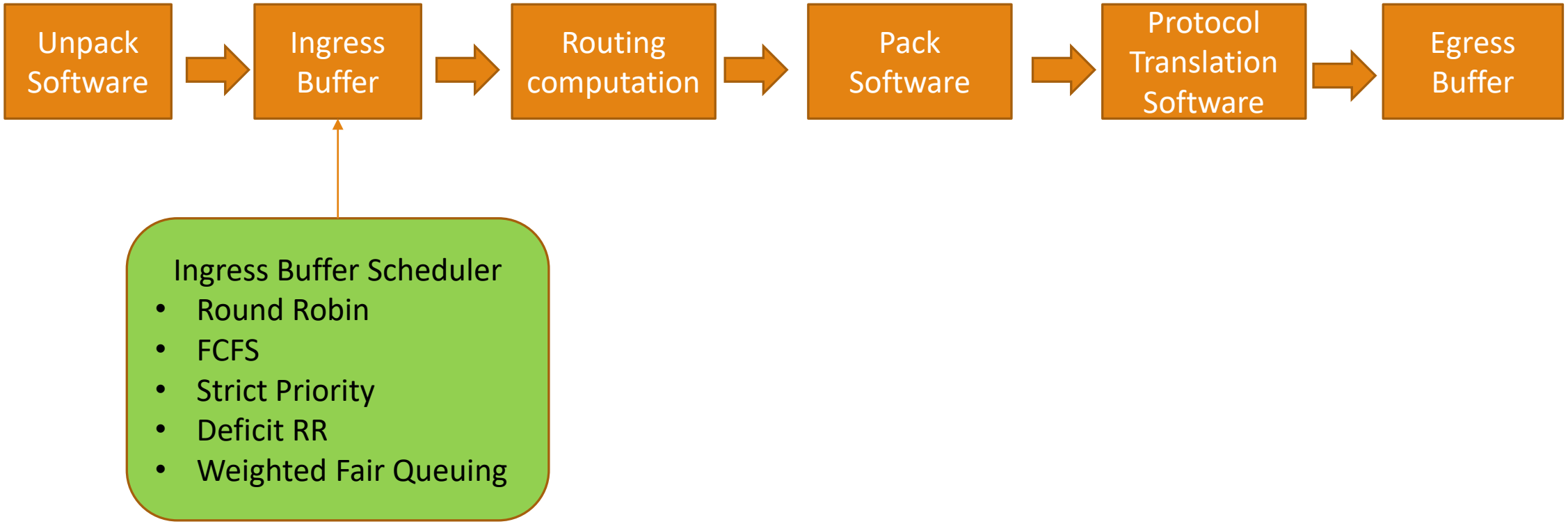
Block Diagram



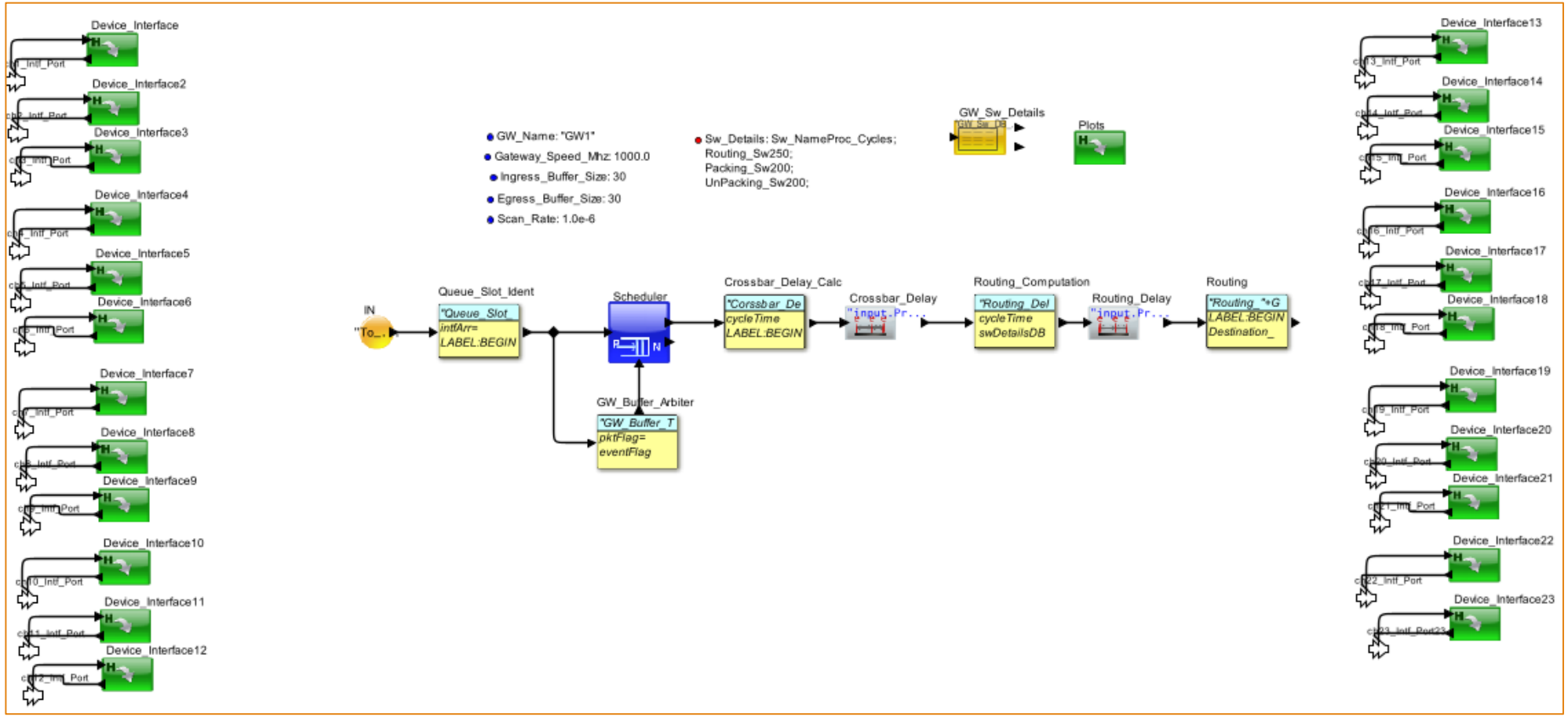
VisualSim Model



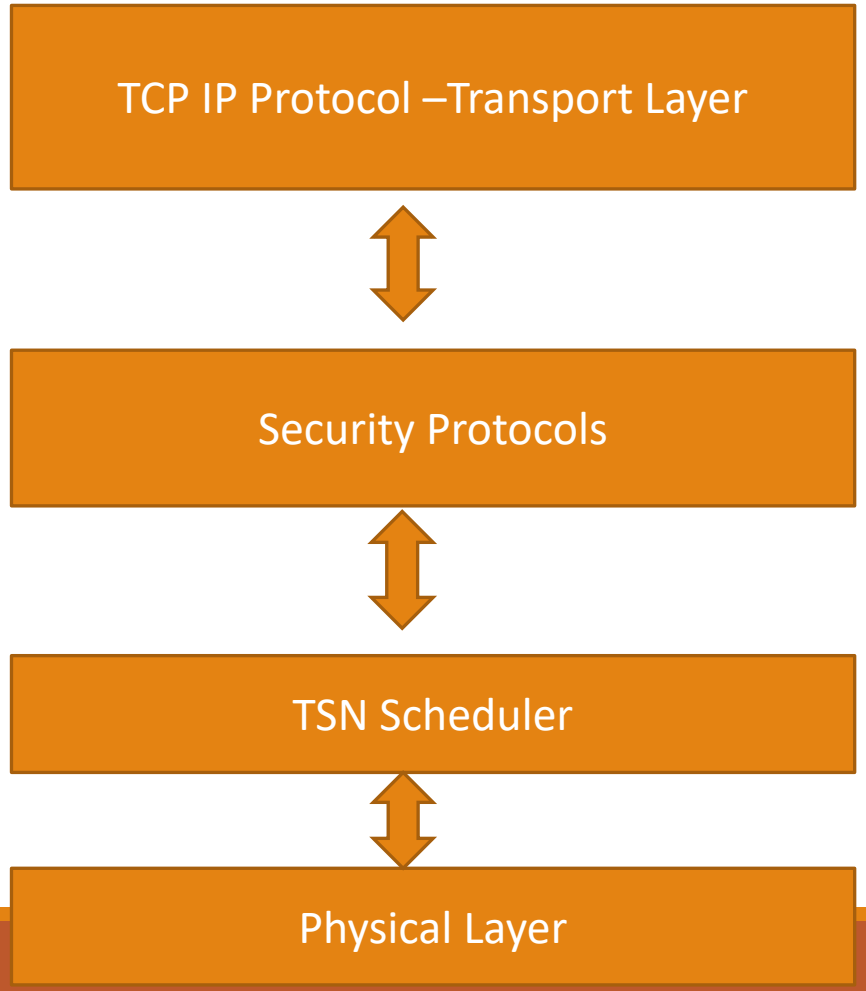
Gateway


Unpack Software
Ingress Buffer
Routing computation
Pack Software
Protocol Translation Software
Egress Buffer
Ingress Buffer Scheduler
• Round Robin
• FCFS
• Strict Priority
• Deficit RR
• Weighted Fair Queuing

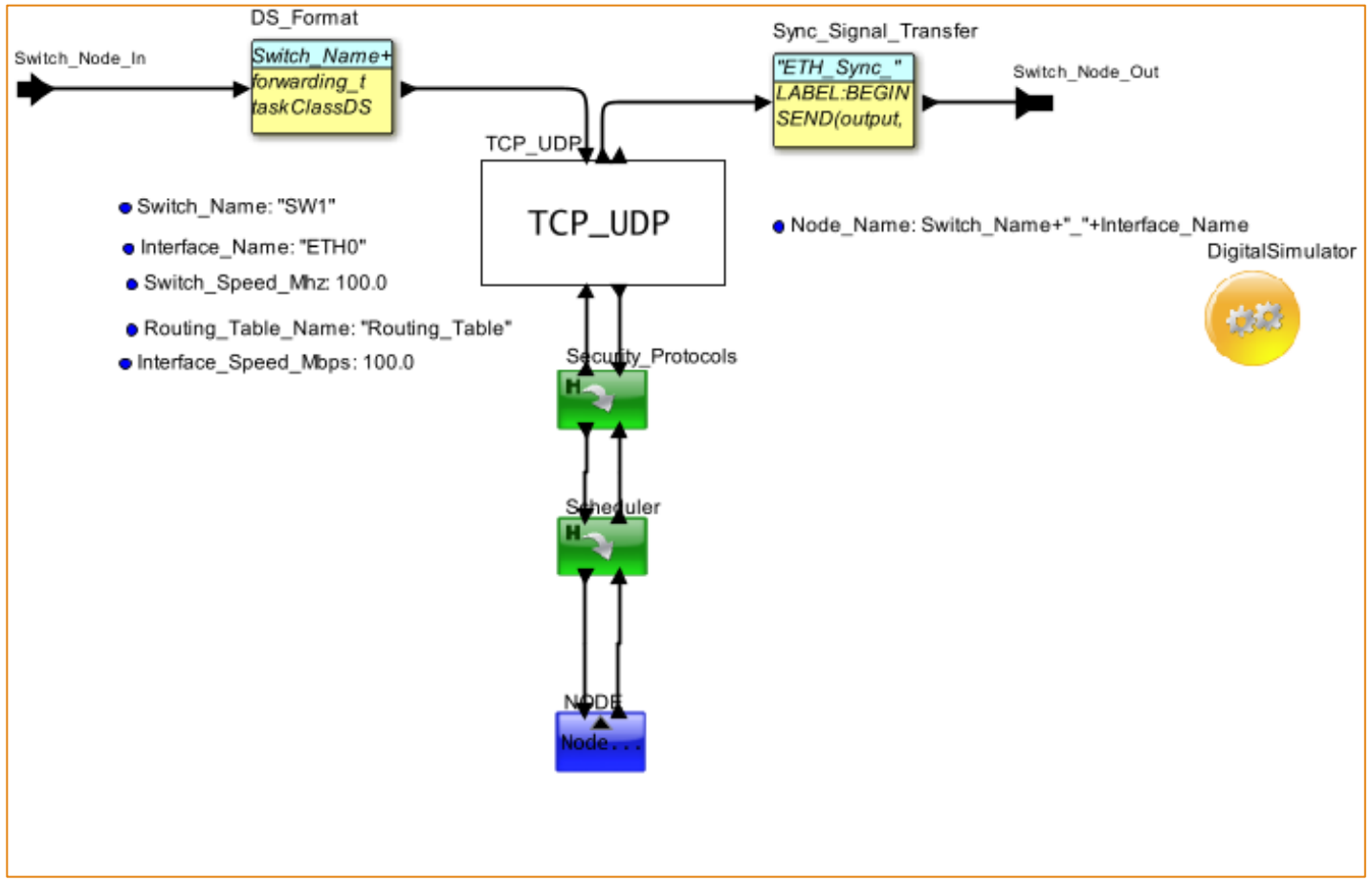
VisualSim Gateway Overview



TSN compliant Ether switch design



VisualSim Model



Integration

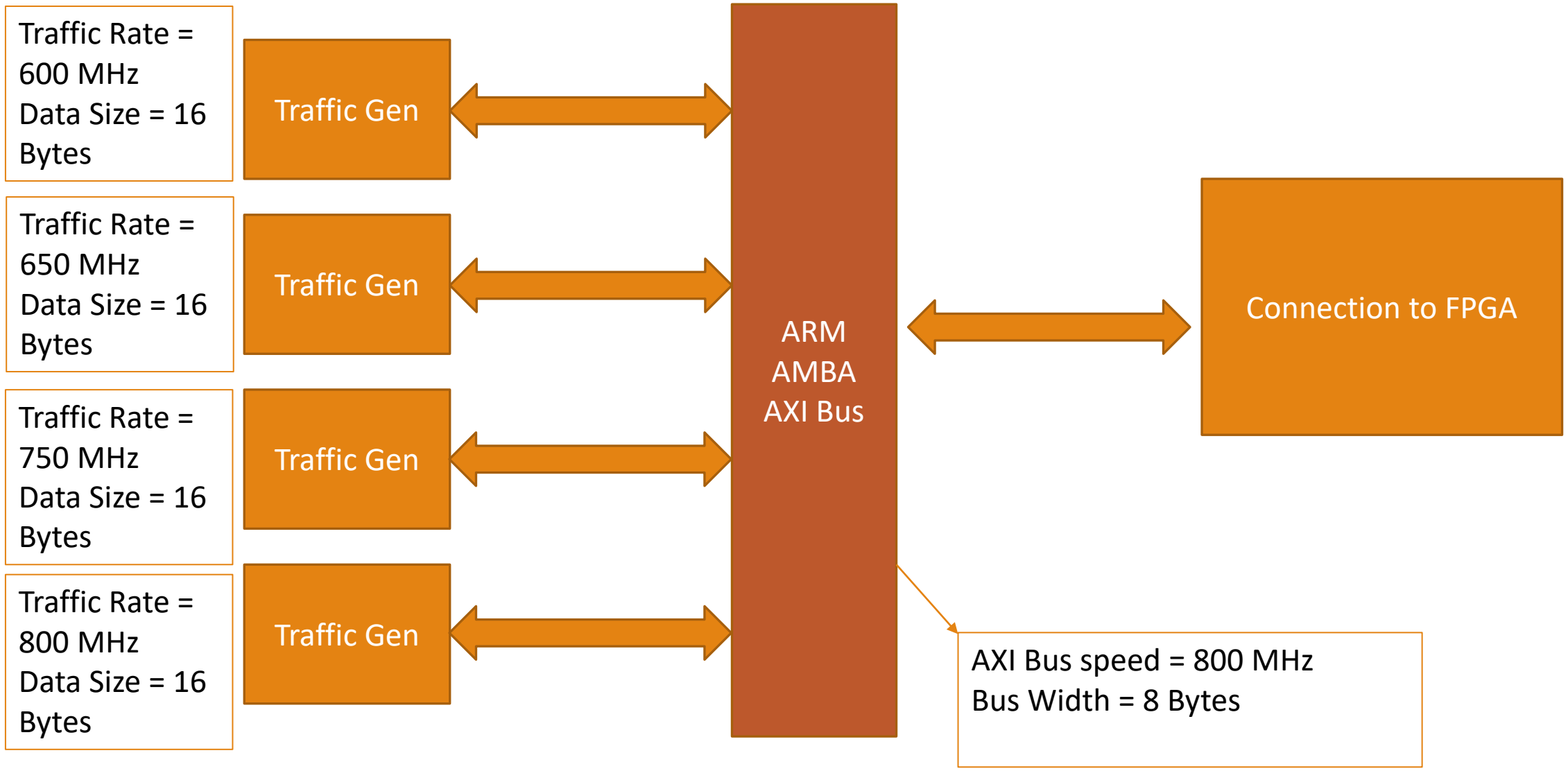
Hardware in the loop - Goals

- Create a test case where synthetic traffic generated from VisualSim Environment is sent to the external Hardware and use the response from the external hardware as input to the VisualSim environment.

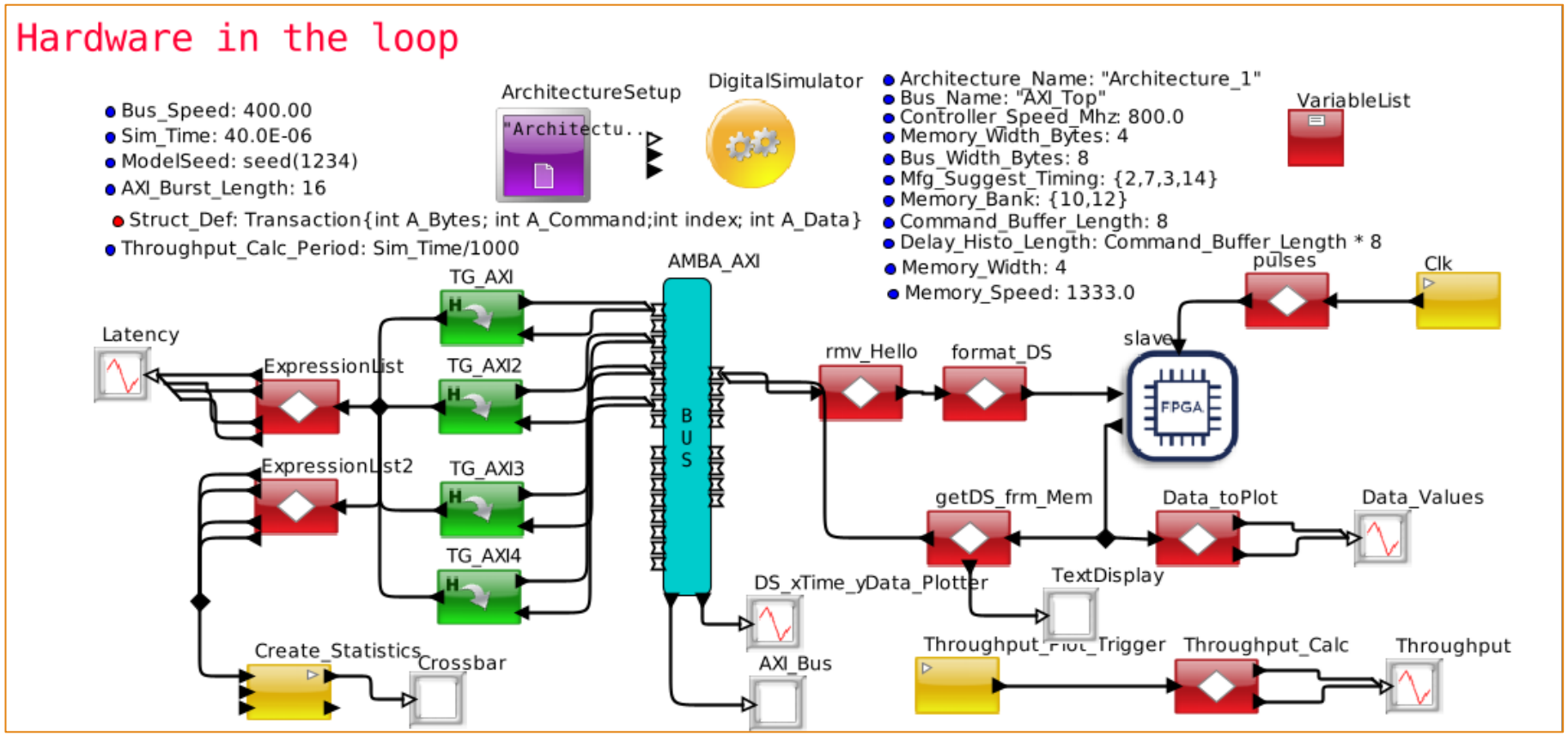
- Two modes of operation:
  - Once we press a dedicated button on FPGA, constant SRAM values are read out
  - Otherwise, a normal Read operation is done

- Plots for Read and Write throughput as well as for the data value

Block Diagram



Block diagram realized in VisualSim Platform



Integration with GEM5

Purposes of the Integration

GEM5 users

- Extend research to cycle-models of the processor, cache, bus and memory

VisualSim users

- Test processor models with instruction sequence from real code execution

VisualSim provides

- Fully tested and commercially supported models of processor cores, cache, buses and memories

What is GEM5

Provides instruction set simulators for ARM, RISC-V, GPU, Power and x86

- Load Linux/Windows/Android and execute the compiled software code.
- Verify the correctness of code behavior on the target instruction set, not on a specific core

Simple branch predictor provided, not match vendor implementation

Unlike Fast models, GEM5 has an experimental platform with templates for caches, buses, memory and branch prediction

- User can customize the processor and peripherals to create proprietary version

Does not provide a specific processor core implementation

- Code execution is identical for ARM v8.1A in ARM Cortex A53, A72, A76 and A78

Common usage

- Academic research and teaching purposes
- Software development
- Creating customized research platform

Advantages & Disadvantages with GEM5

Advantages

- Large user community
- Support for ISS from ARM v8, Power, x86, RISC-V and GPU(AMD)

Disadvantage

- Lack of support
- Accuracy has not been tested

VisualSim with GEM5

Goal

- Execute software code on an emulated hardware system
- Test the software against the full system
- Current focus is performance and power of the full system
- Future focus is correctness of action
- Triggered the right device or sent data to the right interface

VisualSim-GEM5 Integration

Two modes of operation

Mode 1: GEM5 Wrapper

- Generate batches of requests to cache and memory
- GEM5 executes the code and wrapper feeds the addresses to VisualSim model

Mode 2: Trace File

- GEM5 writes the list of instructions and addresses to a file
- VisualSim reads the file using TrafficReader and provides this as input to the VisualSim Processor block

Mode 1: Wrapper

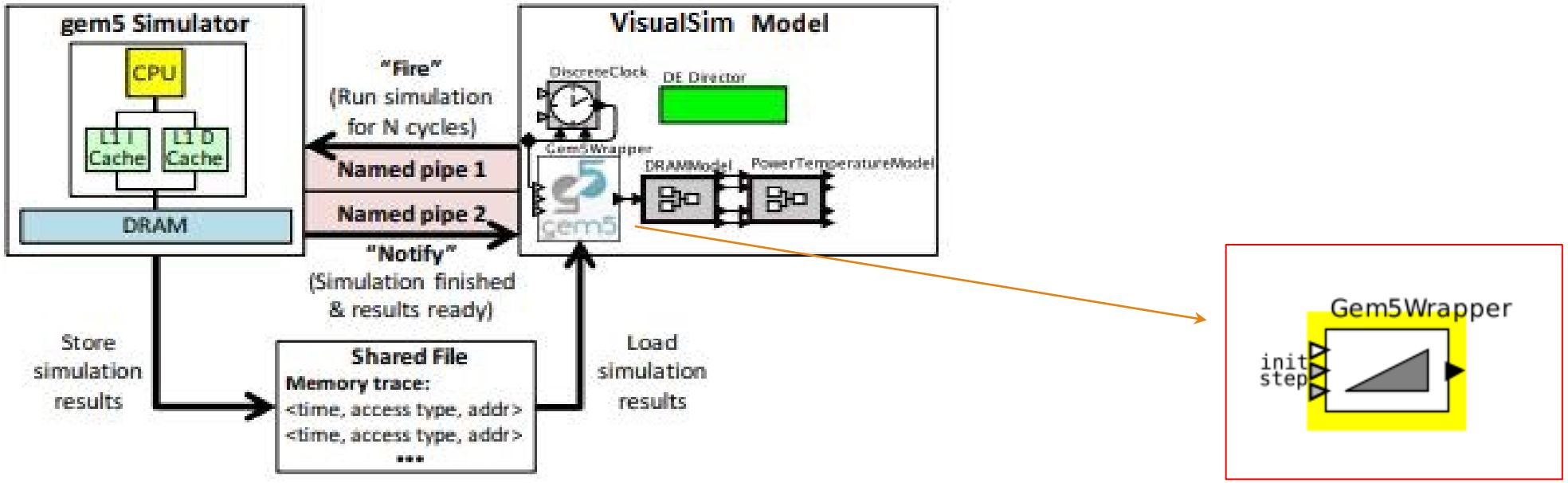
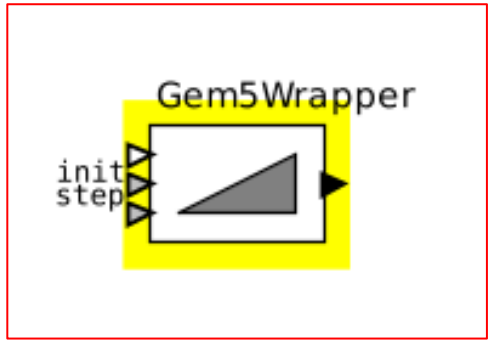
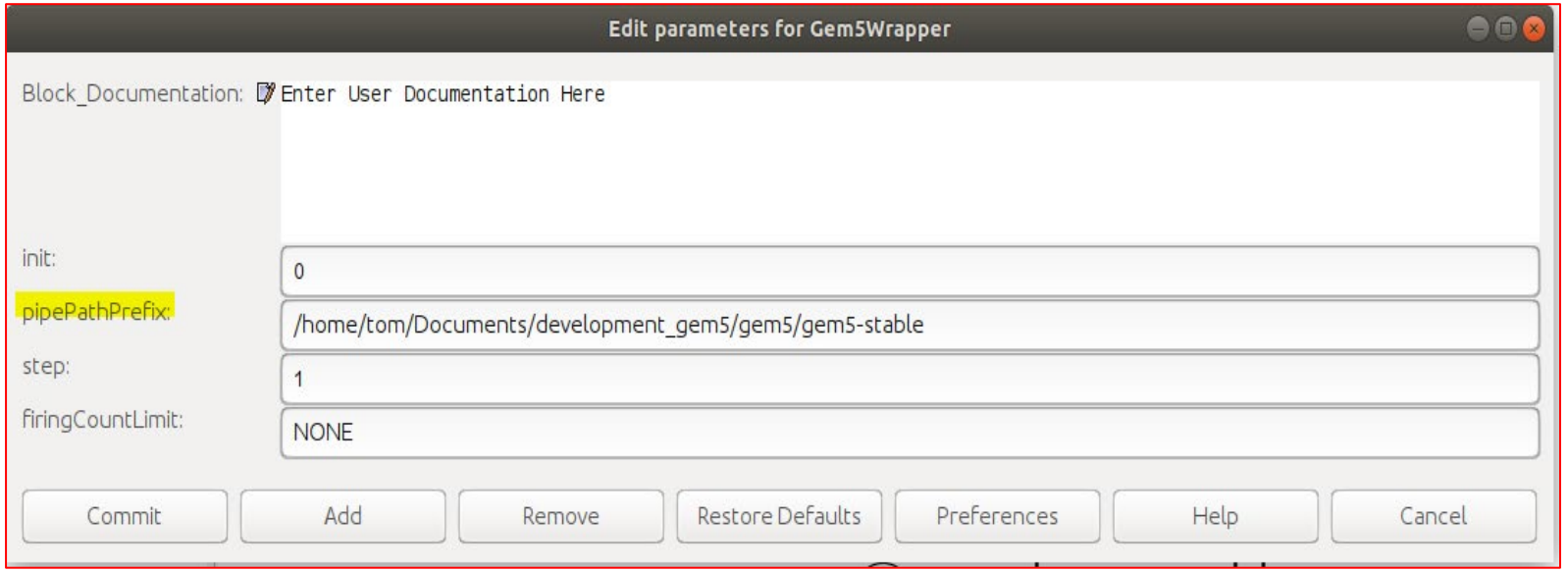


Fig. 2. An overview of gem5 and VisualSim integration

VisualSim GEM5 Wrapper Parameters



*Double click on the block , to edit the **pipePathPrefix***



User has to provide the path to the GEM5 directory where gem5 build is done

Mode 2: Traces generated from GEM5 – *in shared file*

```
info: Entering event queue @ 0.  Starting simulation...
   2000: system.cpu.icache: getBusPacket created ReadReq addr 0x440 size 64
   2000: system.cpu.dcache: recvTimingSnoopReq for ReadReq addr 0x440 size 64
   2000: system.cpu.dcache: handleSnoop for ReadReq addr 0x440 size 64
   2000: system.cpu.dcache: handleSnoop snoop miss for ReadReq addr 0x440 size 64
   2000: system.mem_ctrls: recvTimingReq: request ReadReq addr 1088 size 64
   2000: system.mem_ctrls: Read queue limit 32, current size 0, entries needed 1
   2000: system.mem_ctrls: Address: 1088 Rank 0 Bank 0 Row 0
   2000: system.mem_ctrls: Read queue limit 32, current size 0, entries needed 1
   2000: system.mem_ctrls: Adding to read queue
   2000: system.mem_ctrls: Request scheduled immediately
   2000: system.mem_ctrls: Single request, going to a free rank
   2000: system.mem_ctrls: Timing access to addr 1088, rank/bank/row 0 0 0
   2000: system.mem_ctrls: 2000,ACT2
   2000: system.mem_ctrls: VISUALSIM_LOG: Rank: 0 Bank: 0 SIZE: 64 ACT: 0 READ: 13750 Address: 1088 Row: 0
   2000: system.mem_ctrls: Activate at tick 2000
   2000: system.mem_ctrls: Activate bank 0, rank 0 at tick 2000, now got 1 active
   2000: system.mem_ctrls: Access to 1088, ready at 46250 bus busy until 46250.
  46250: system.mem_ctrls: processRespondEvent(): Some req has reached its readyTime
  46250: system.mem_ctrls: Responding to Address 1088..   46250: system.mem_ctrls: Done
  73250: system.cpu.icache: Handling response ReadResp for addr 0x440 size 64 (ns)
  73250: system.cpu.icache: Block for addr 0x440 being updated in Cache
  73250: system.cpu.icache: Block addr 0x440 (ns) moving from state 0 to state: 7 (E) valid: 1 writable: 1 readable: 1 dirty: 0 tag: 0
  73250: system.cpu.icache: Leaving recvTimingResp with ReadResp for addr 0x440
  79000: system.cpu T0 : @_start    :   mov   fp, #0             : IntAlu :  D=0x0000000000000000
  79000: system.cpu.icache: access for ReadReq addr 0x450 size 4
  79000: system.cpu.icache: ReadReq (ifetch) addr 0x450 size 4 (ns) hit state: 7 (E) valid: 1 writable: 1 readable: 1 dirty: 0 tag: 0
  81000: system.cpu T0 : @_start+4    :   mov   lr, #0             : IntAlu :  D=0x0000000000000000
  81000: system.cpu.icache: access for ReadReq addr 0x454 size 4
  81000: system.cpu.icache: ReadReq (ifetch) addr 0x454 size 4 (ns) hit state: 7 (E) valid: 1 writable: 1 readable: 1 dirty: 0 tag: 0
  83000: system.cpu.dcache: access for ReadReq addr 0x8de50 size 4
  83000: system.cpu.dcache: ReadReq addr 0x8de50 size 4 (ns) miss
  85000: system.cpu.dcache: getBusPacket created ReadReq addr 0x8de40 size 64
  85000: system.cpu.icache: recvTimingSnoopReq for ReadReq addr 0x8de40 size 64
  85000: system.cpu.icache: handleSnoop for ReadReq addr 0x8de40 size 64
  85000: system.cpu.icache: handleSnoop snoop miss for ReadReq addr 0x8de40 size 64
```

Model 2: Trace file Converted to VisualSim Format

TimeStamp	Source_Device_Name	Dest_Device_Name	Comment	Command	Address	size
2000	cpu	icache	getBusPacket	ReadReq	0x440	64
2000	cpu	dcache	recvTimingSnoopReq	ReadReq	0x440	64
2000	cpu	dcache	handleSnoop-snoop miss	ReadReq	0x440	64
2000		mem_ctrls	recvTimingReq	ReadReq	1088	64
2001		mem_ctrls	Responding to Address		1088	64
73250	cpu	icache	Handling response	ReadResp	0x440	64
154000	cpu	dcache	access	WriteReq	0x8de50	4

Cache and Memory stats

Time_Stamp	CPU_Core	Instructions	Execution_unit	
79000	T0	mov	IntAlu	D=0x0000000000000000
81000	T0	mov	IntAlu	D=0x0000000000000000
83000	T0	ldr	MemRead	D=0x0000000000000008
154000	T0	str	MemWrite	D=0x00000000befffe54 A=0xbefffe50
156000	T0	subi_uop	IntAlu	D=0x00000000befffe50
158000	T0	str	MemWrite	D=0x0000000000000000 A=0xbefffe4c
160000	T0	subi_uop	IntAlu	D=0x00000000befffe4c

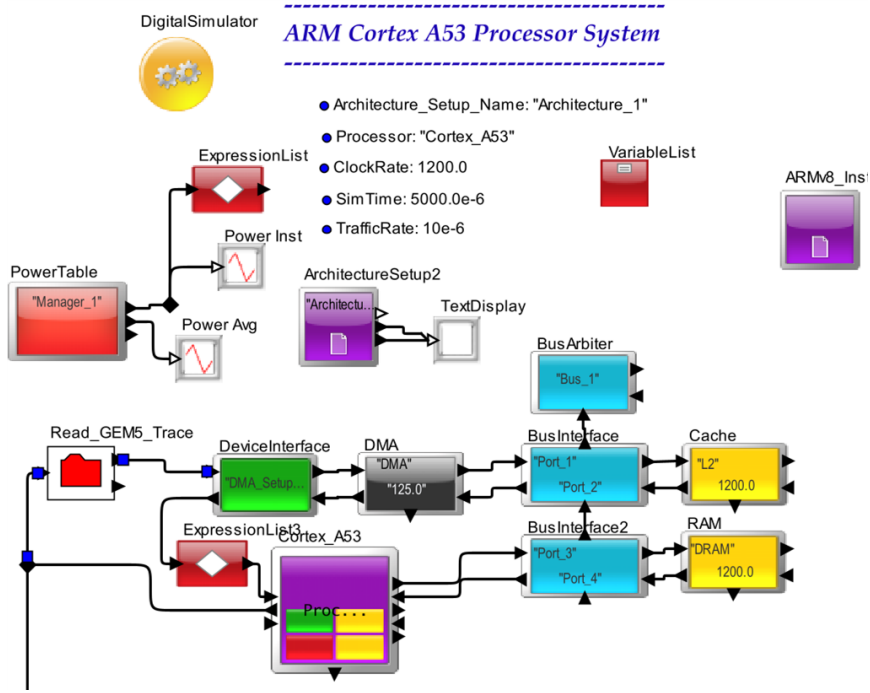
Demo output csv from gem5 traces.

Different cpu cores stats

Mode 2: Using Trace in VisualSim

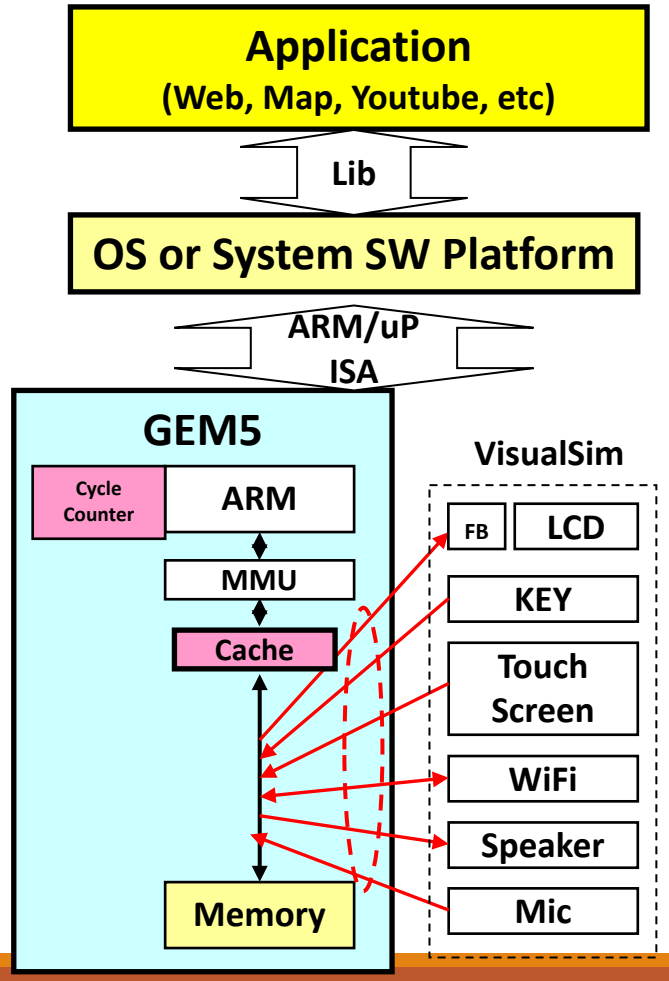
TimeStamp	Source_Device_Name	Dest_Device_Name	Comment	Command	Address	size
2000	cpu	icache	getBusPacket	ReadReq	0x440	64
2000	cpu	dcache	recvTimingSnoopReq	ReadReq	0x440	64
2000	cpu	dcache	handleSnoop-snoop miss	ReadReq	0x440	64
2000		mem_ctrls	recvTimingReq	ReadReq	1088	64
2001		mem_ctrls	Responding to Address		1088	64
73250	cpu	icache	Handling response	ReadResp	0x440	64
154000	cpu	dcache	access	WriteReq	0x8de50	4

Time_Stamp	CPU_Core	Instructions	Execution_unit	
79000	T0	mov	IntAlu	D=0x0000000000000000
81000	T0	mov	IntAlu	D=0x0000000000000000
83000	T0	ldr	MemRead	D=0x0000000000000008
154000	T0	str	MemWrite	D=0x00000000befffe54 A=0xbefffe50
156000	T0	subi_uop	IntAlu	D=0x00000000befffe50
158000	T0	str	MemWrite	D=0x0000000000000000 A=0xbefffe4c
160000	T0	subi_uop	IntAlu	D=0x00000000befffe4c



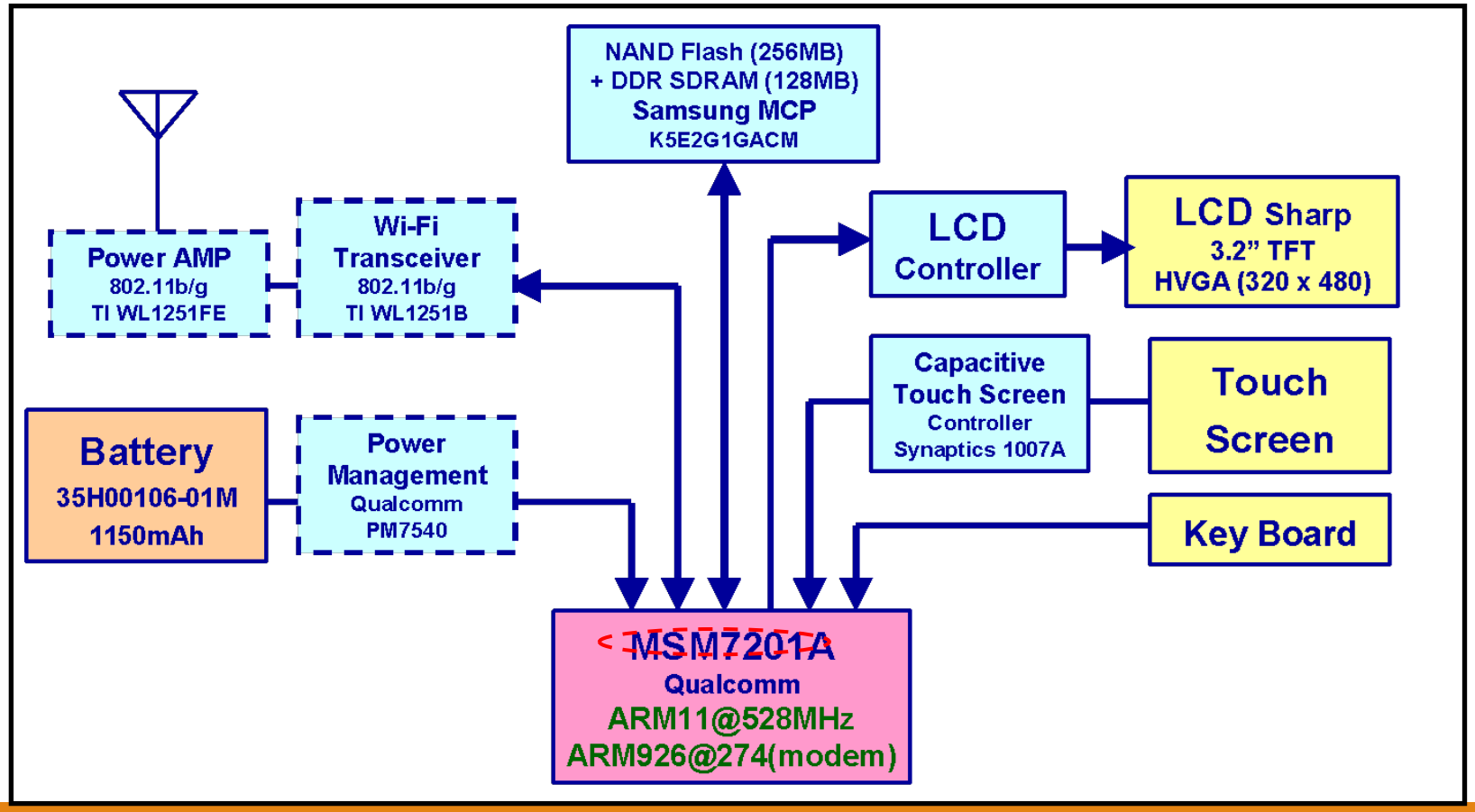
These instructions read via TrafficReader as input to the Processor block

Linking GEM5 to VisualSim



Representative Example on VisualSim

Hardware Platform on VisualSim



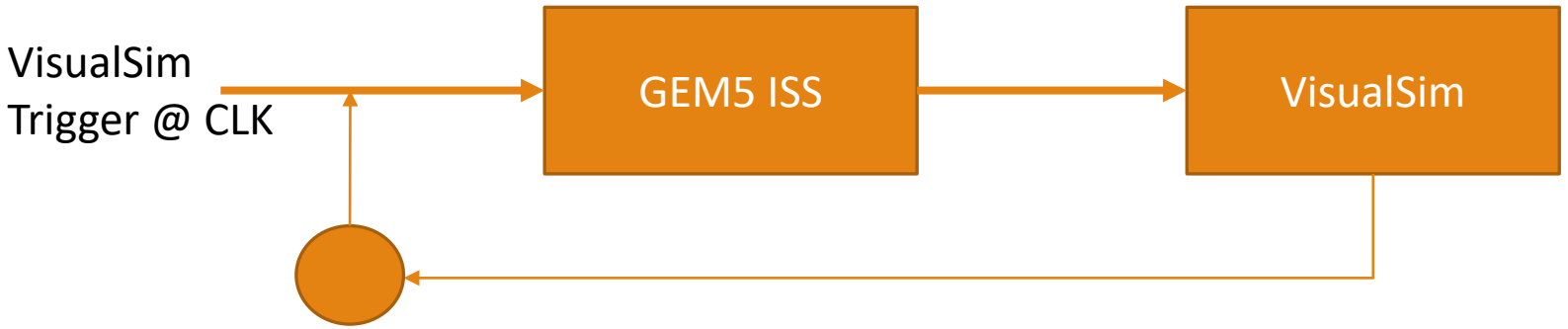
How this works

VisualSim triggers the software to execute

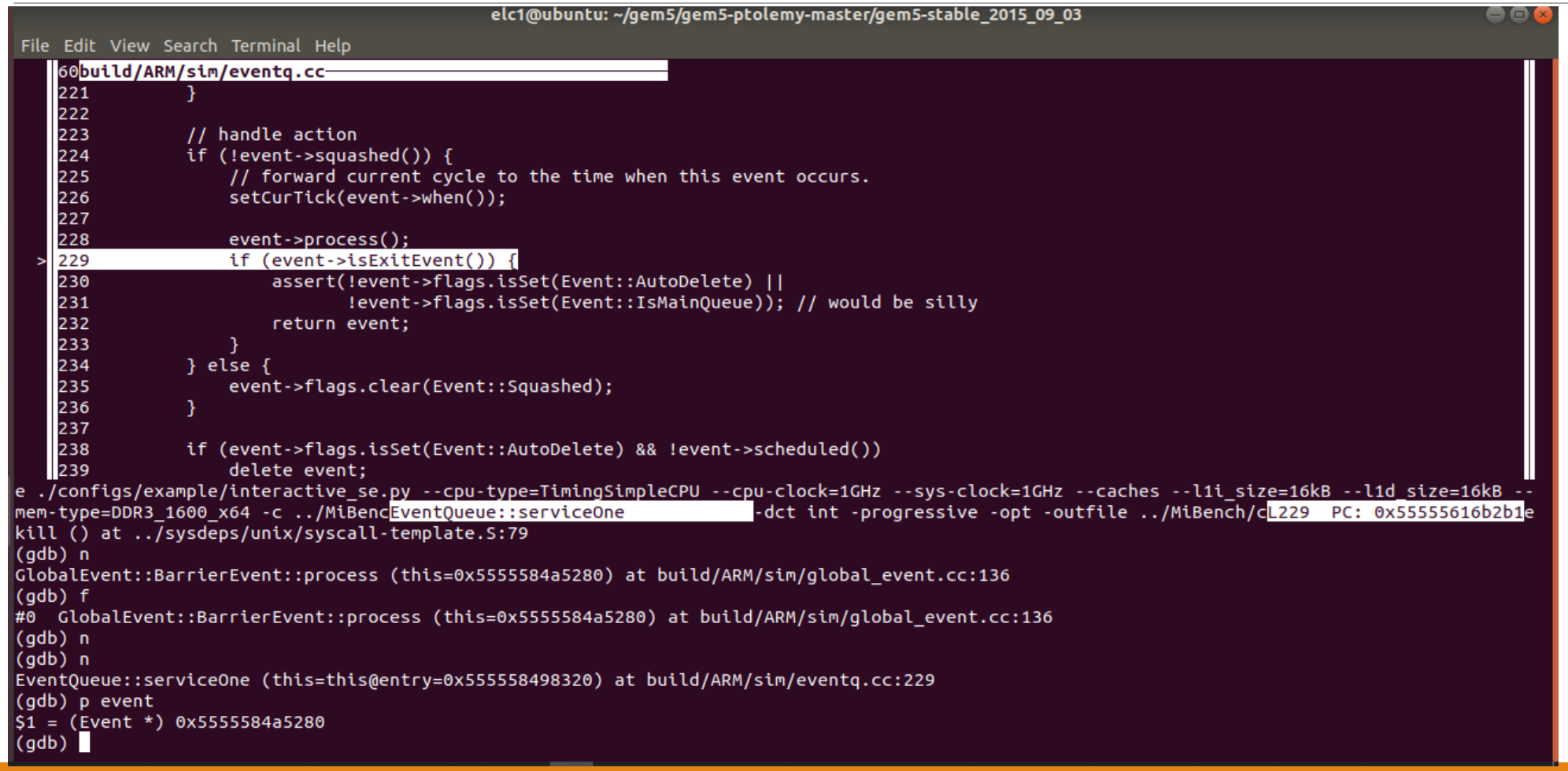
GEM5 executes for a time duration

Output the addresses, service time and the time stamp

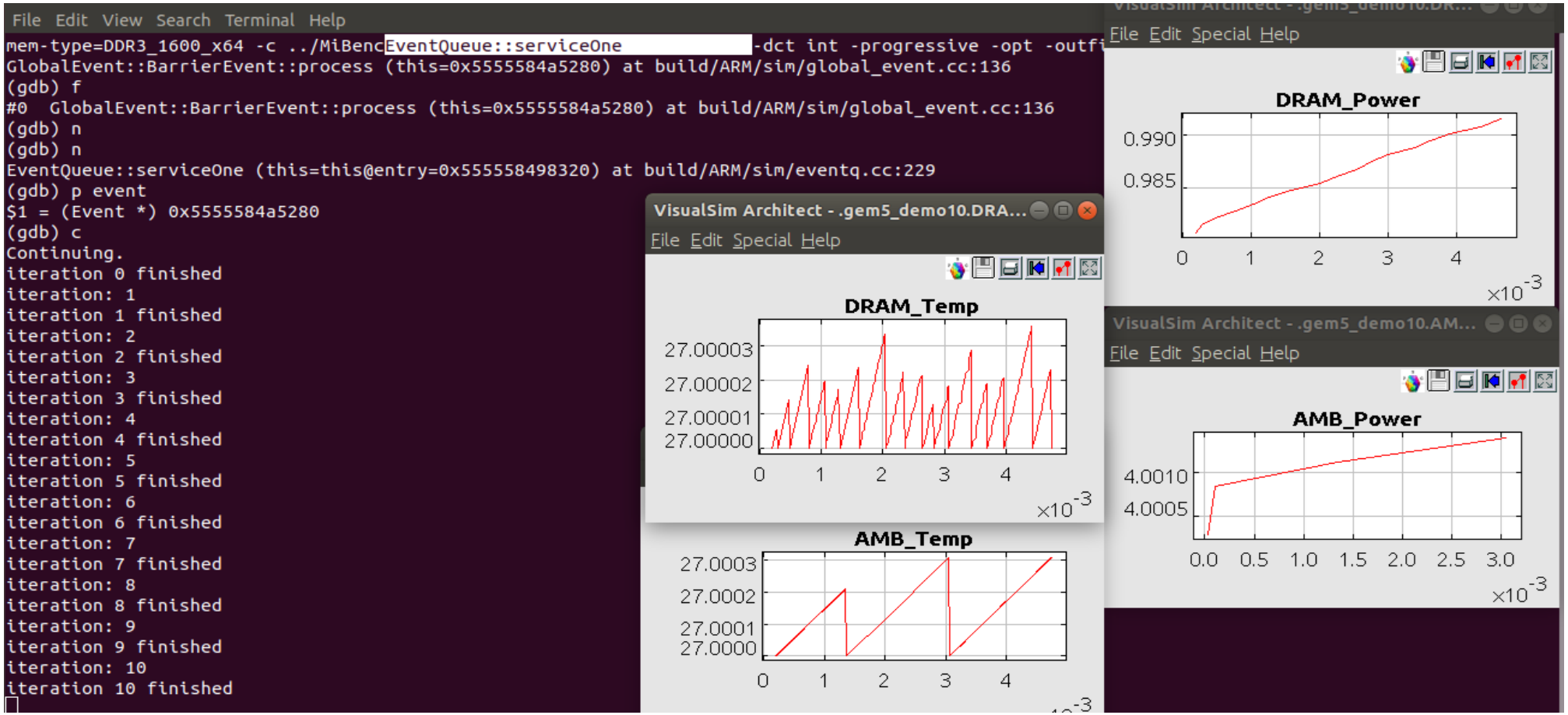
GEM5 can be triggered on a fixed schedule like real-time software or can be triggered after the operation is completed in VisualSim



Debug Software in System Context



Integrate Debugging and System Analysis



Enhancements

GEM5

- Multi-core with different software on each core
- Add RISC-V and GPU models
- Trigger software instances as opposed to full program
- Add support for more debuggers

Provide services to develop new ISS

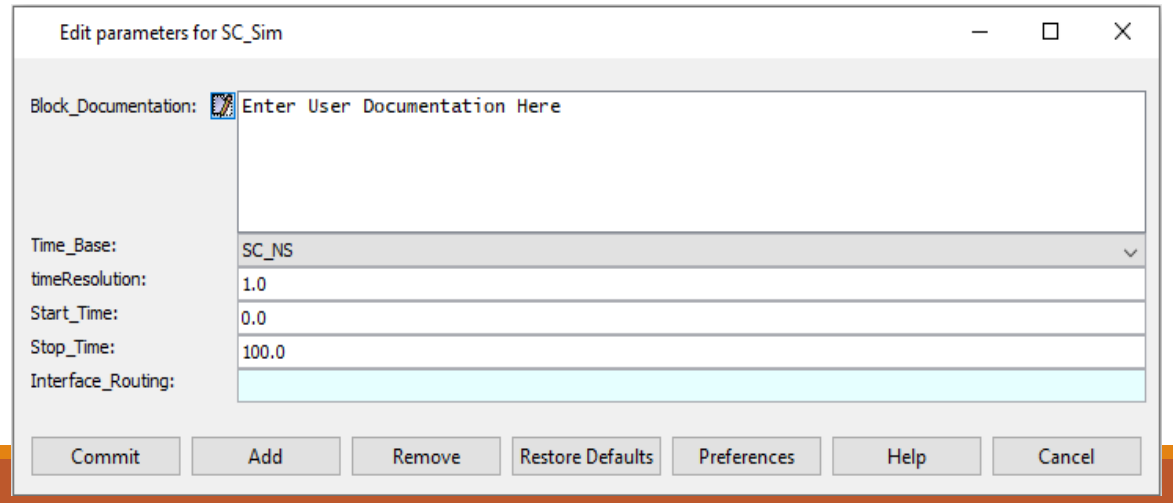
Integrate ARM Fast Models

SystemC package to add processors like CEVA and Tensilica

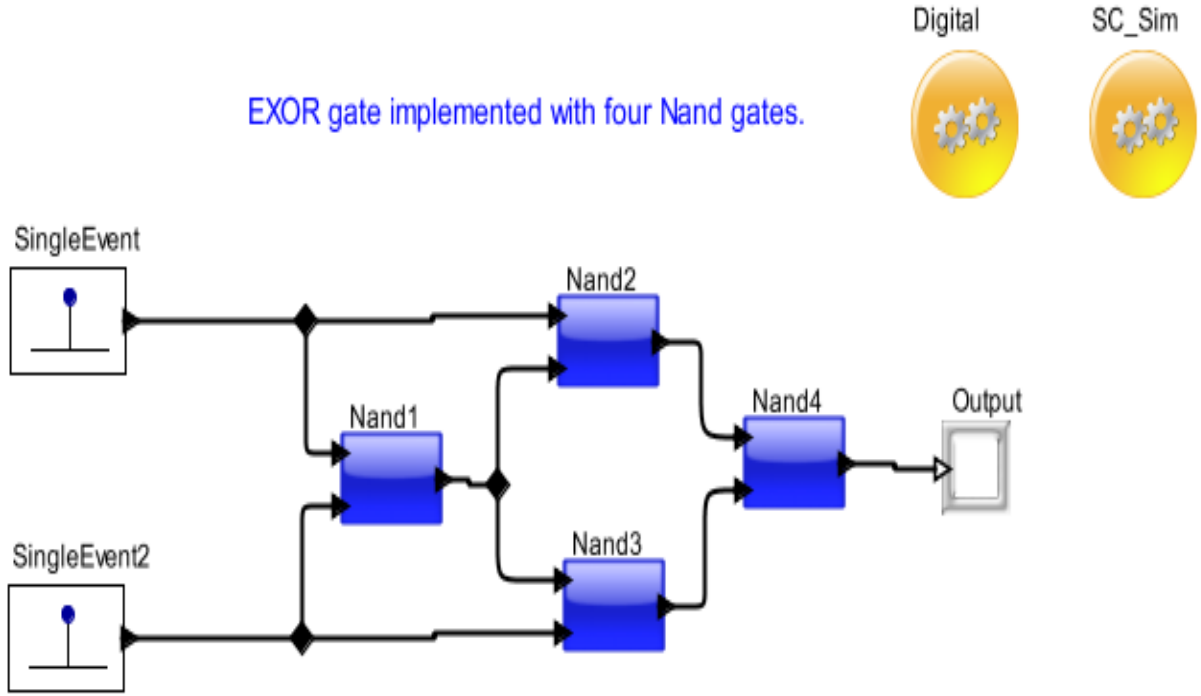
- Using existing SystemC integrate

Integration with SystemC

- **Full Library -> Hardware Language -> SystemC -> SC_Sim**
- Provides timed interface between VisualSim and SystemC
- Timed interface - Synchronization between VisualSim and SystemC simulator

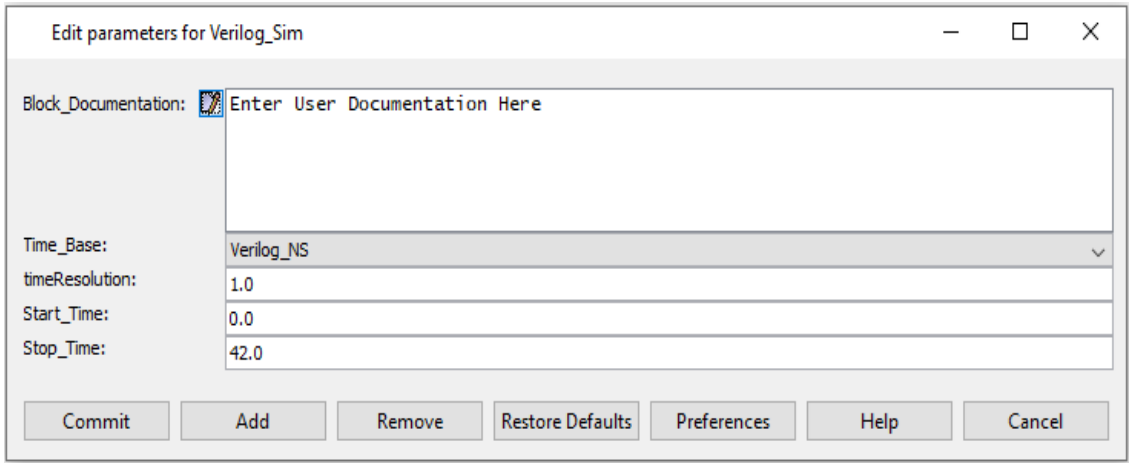


SystemC model- Example

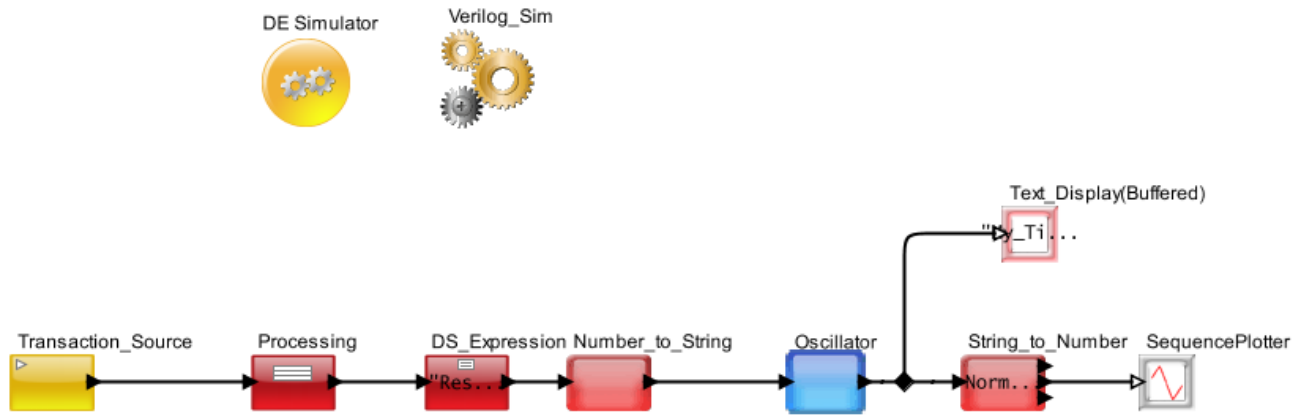


Integration with Verilog

- **Full Library -> Hardware Language -> Verilog-> Verilog_Sim**
- Provides timed interface between Visualsim and Verilog
- Timed interface - Synchronization between VisualSim and Verilog simulator



Verilog model- Example



MIRABILIS design

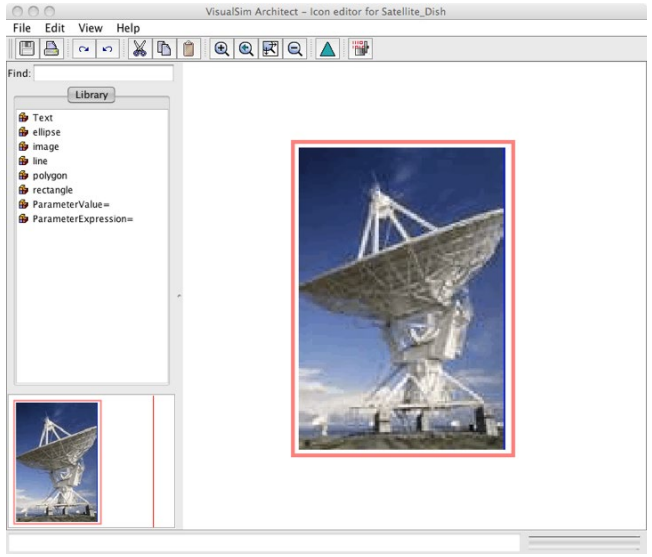
Version Control

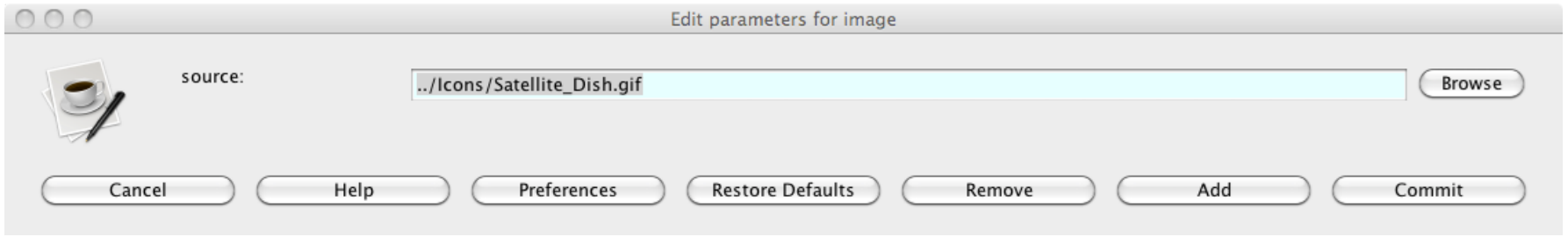
Version Control

- **Version Control and VisualSim Architect**
  - ✔ Library Structure and Self-Contained Classes
- **Issues in Version Control**
  - ✔ Consistent library structure (for a given design) amongst all team members
  - ✔ Self-contained classes
- **One potential self-contained library structure**
  - ✔ Icons
  - ✔ Components
  - ✔ Designs
- **Self-contained classes**
  - ✔ Must only contain references to internal constructs (e.g., parameters, Variables, virtual connections, etc.)
  - ✔ Must NOT contain references or have dependencies on external parameters, variables, virtual connections, etc.

Version Control and VisualSim Architect

Custom Icons and Library Structure





Version Control and VisualSim Architect
Version Control using SVN

- **Step 1: Create a new repository**
  - ✔ cd /var/svn
  - ✔ svnadmin create repos
- **Step 2: Import local tree of data for the first time**
  - ✔ svn import <Path to MyProject> file:///var/svn/repos/<MyProject> -m “initial import”
- **Step 3: Checkout MyProject**
  - ✔ svn checkout file:///var/svn/repos/<MyProject> <MyProject>
- **Other commands**
  - ✔ svn diff
  - ✔ svn commit
  - ✔ svn update

Version Control Using CLASSPATH

- Select the Master Directory and the Working Directory
  - ✔ The Working directory would typically be on the desktop or local to the user.
  - ✔ The Master directory will be central and accessible by all users.
- Update VS_Model_Library setting in the VisualSim.bat and VisualSim.sh.
  - ✔ Make sure to enter the working directory first and then the Master directory

(*before) set VS_Model_Library=%INSTALL_PATH%\User_Library*

(after) :: For Working Directory setting

set WORKING_PATH= C:\Users\MYName\Desktop

:: For Master Directory setting

set MASTER_PATH=C:\Master

:: For adding Working Directory first and then the Master directory

set VS_Model_Library=%INSTALL_PATH%\User_Library;%WORK_PATH%;%MASTER_PATH%

Continued

- Now create a class Block1 and store in path below the Master Directory. The file will called Block1.xml. An example of the location would be <<Master Directory>>/Level1/Block1.xml
  - ✔ Note: One caveat is that the class hierarchy structure in the file system structure must be identical from the base of the relative path of the Working and Master settings in the VS_Model_Library. The class must not be an absolute folder definition.
    - It means that location of Block 1.xml should be

      <<Master Directory>>/Level1/Block1.xml

      <<Working Directory>>/Level1/Block1.xml

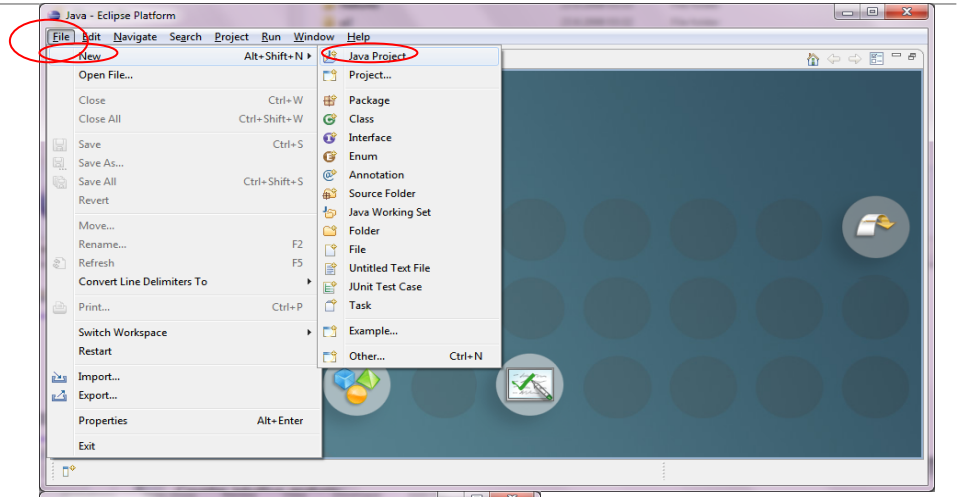
Continued

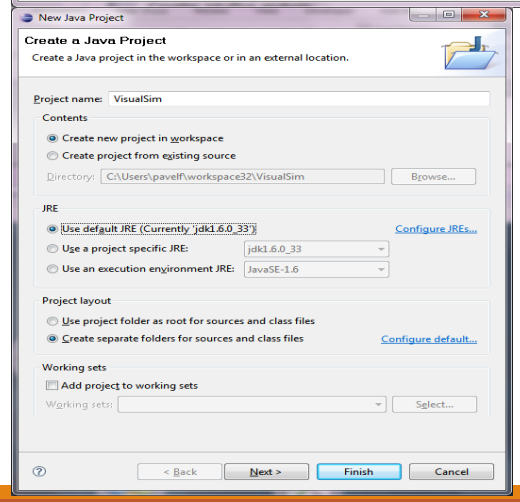
- Now create Model_A and Instantiate Class Block1.
- Save this model anywhere.
- Now copy Block1.xml to the <<Working Directory>>/Level1/Block1.xml.
- Now re-open the model. Open Block of the Class and you will see that it references the Working Directory file.
- Now edit the class Block1.xml in the Working Directory.
- When the Edit has been completed, copy the Block1.xml in working Directory to Master directory and delete the class in the Working Directory.
- Now open the model. You will see that the Class references Master directory location.

ECLIPSE DEBUGGER SETUP

Eclipse Debugger Setup

Create a new Java project

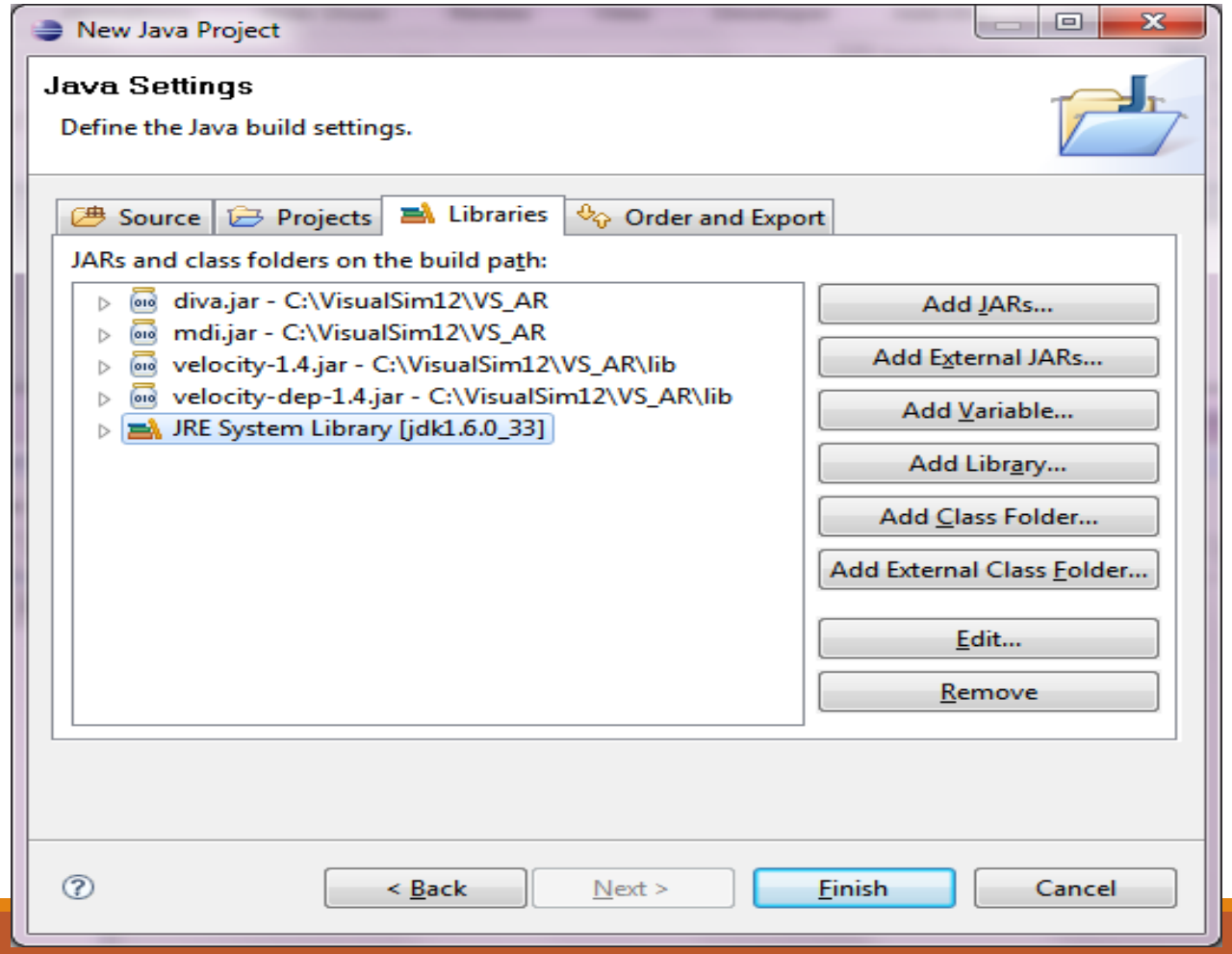




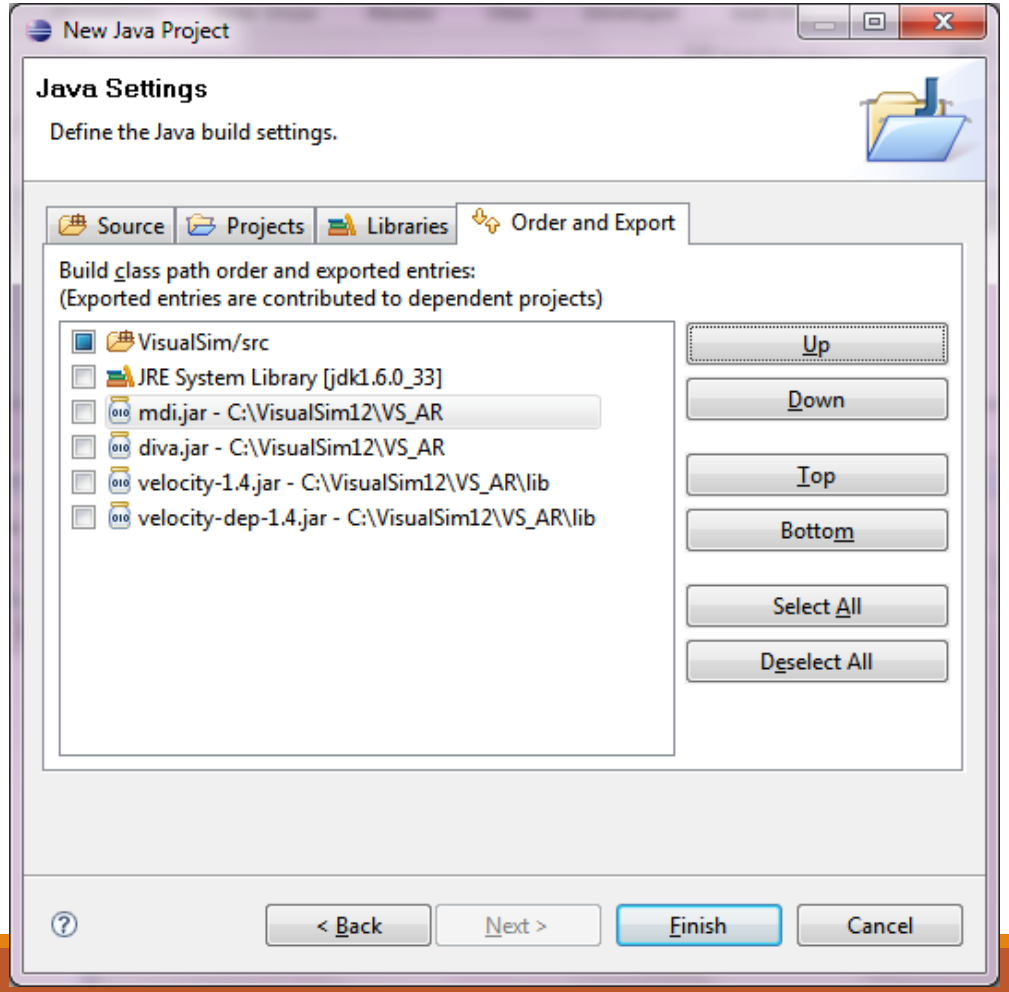
Set project name and JRE (JDK 1.6).

Eclipse Debugger Setup

Configure all necessary libraries



Set proper order of libraries. The sources should be on top.



Eclipse Debugger Setup

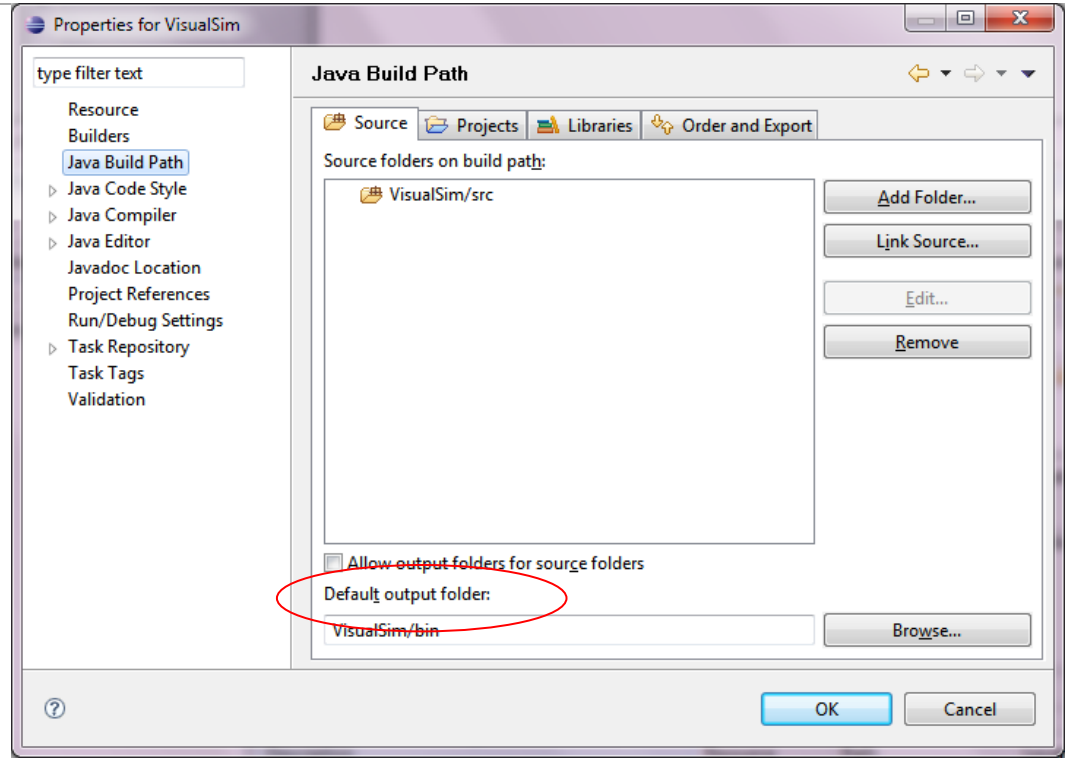
Update VisualSim start script

- Add compiled classes to the class path
  - ✔ set CLASSPATH=<path to complied classes>;%CLASSPATH%
- Prepare Java debug settings
  - ✔ set dbg=-Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,server=y,suspend=n, address=<debug port>
- Modify java command
  - ✔ java %dbg% … VisualSim.ModelBuilder.ModelBuilderApplication

Eclipse Debugger Setup

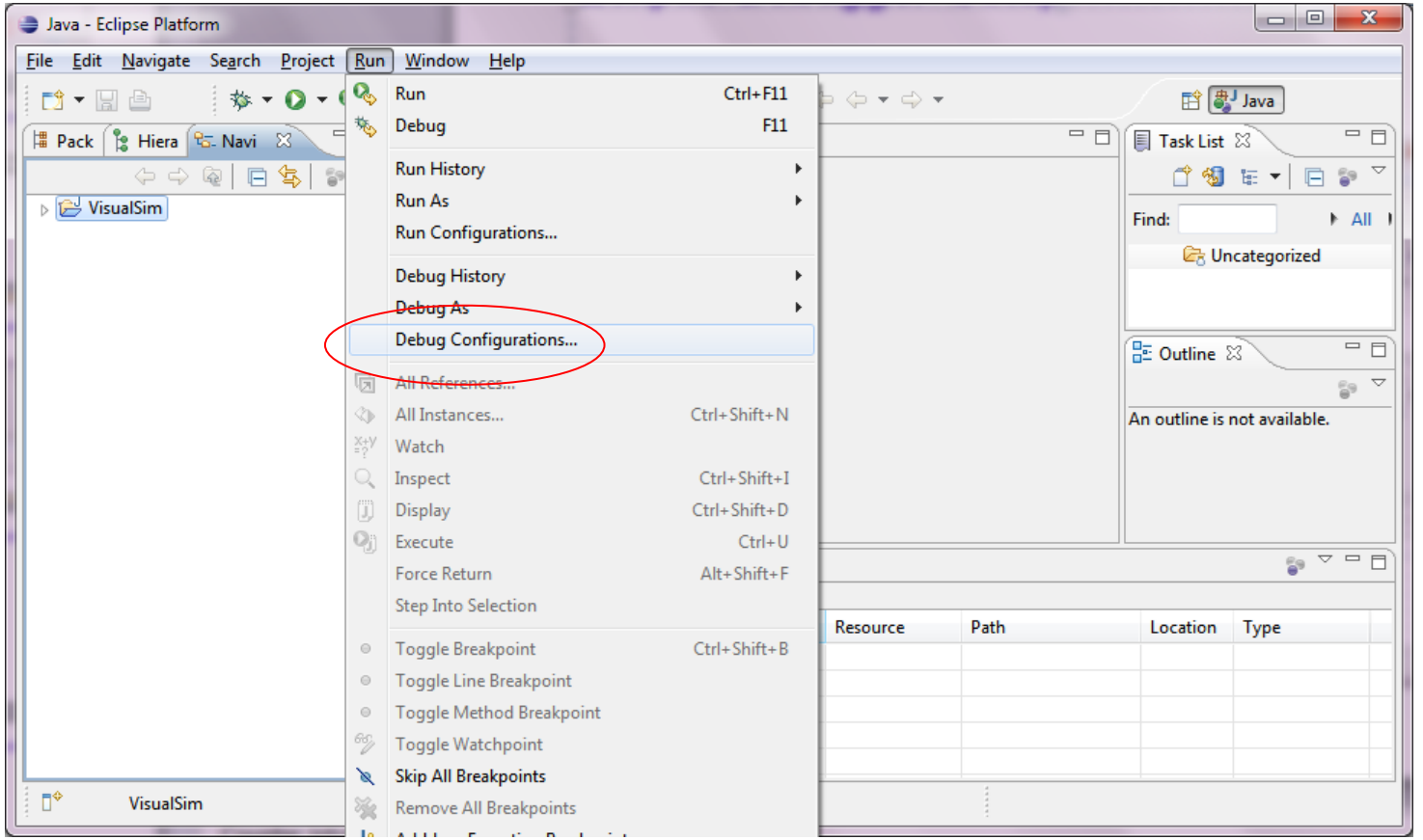
Path to compiled classes is full path to “Default output folder”

- Run the VisualSim using the script.
- Java is listening on debug port. Debugger is able to attach to the port.



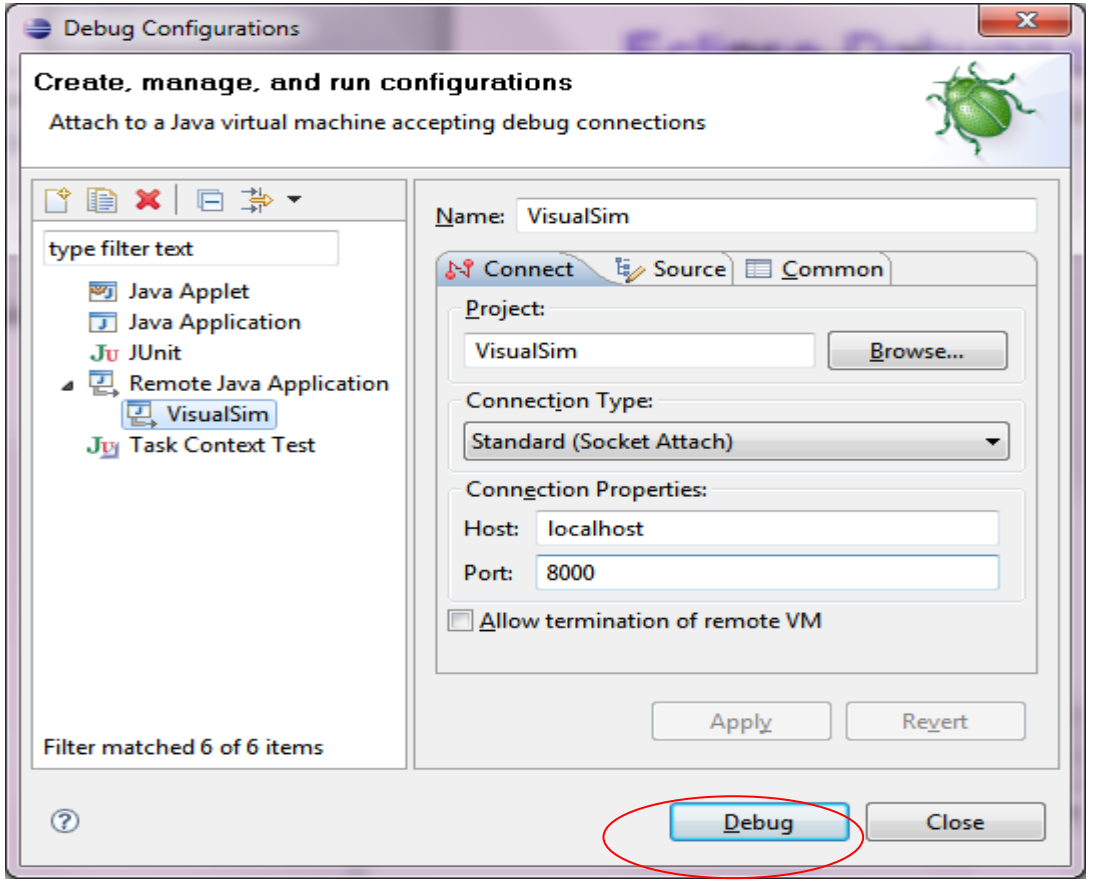
Eclipse Debugger Setup

Setup debug configuration



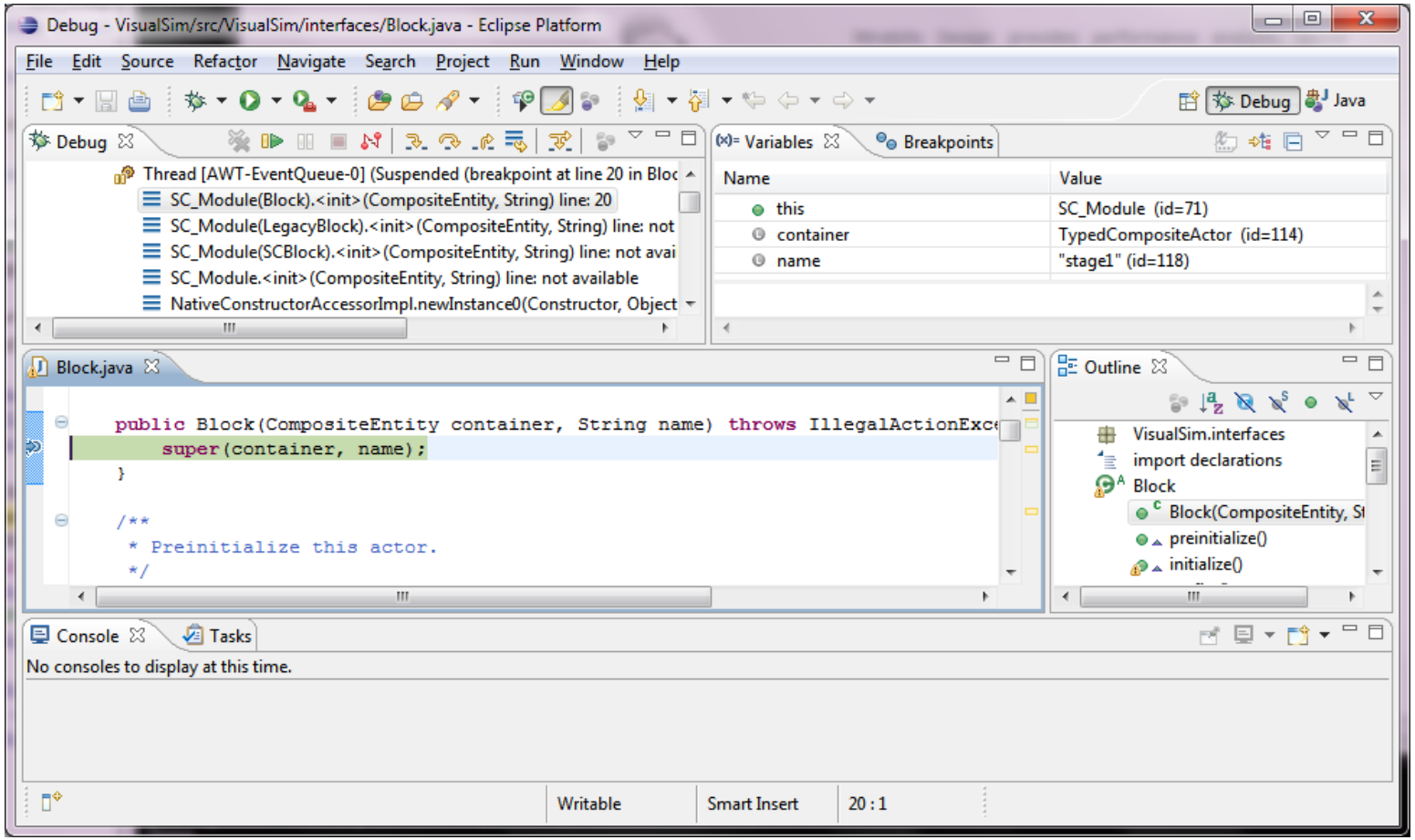
Eclipse Debugger Setup

- Choose Remote Java Application. You can use default settings. Port should correspond to <debug port> in Java debug options.
- Click Debug



Eclipse Debugger Setup

Now you can debug the code



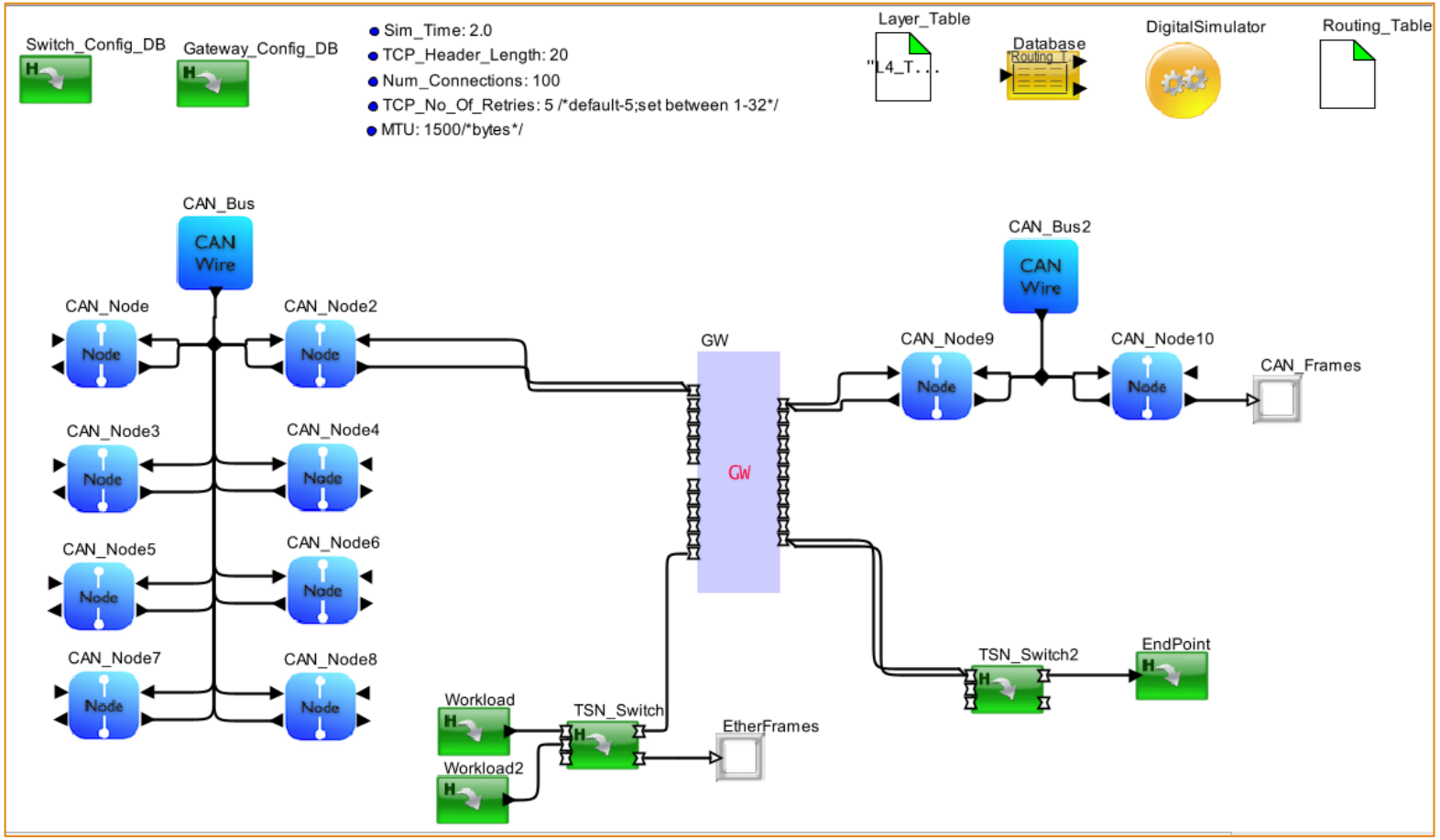
MIRABILIS design

Use Cases and Examples

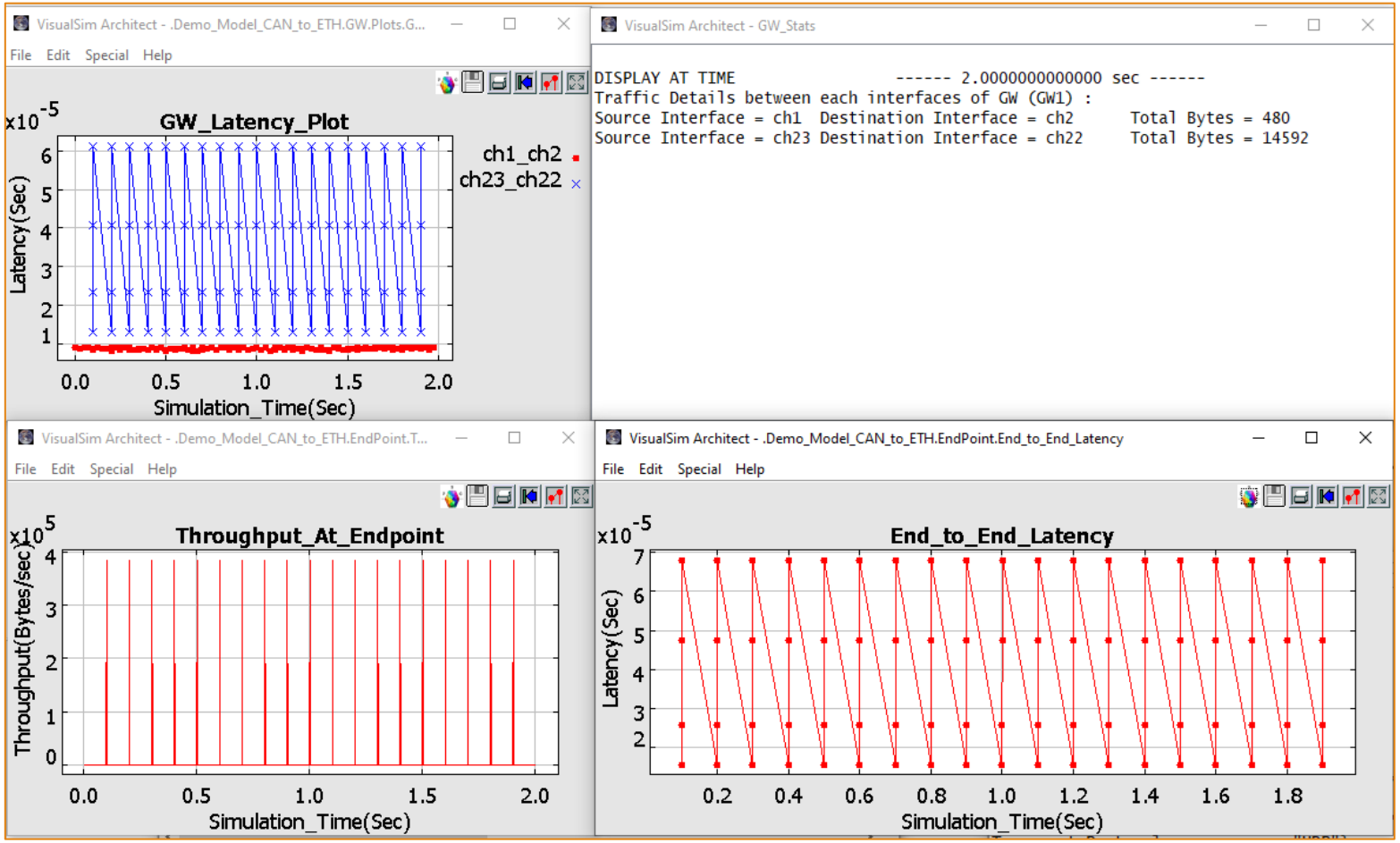
Use cases

20% NETWORK CAPACITY

VisualSim Model



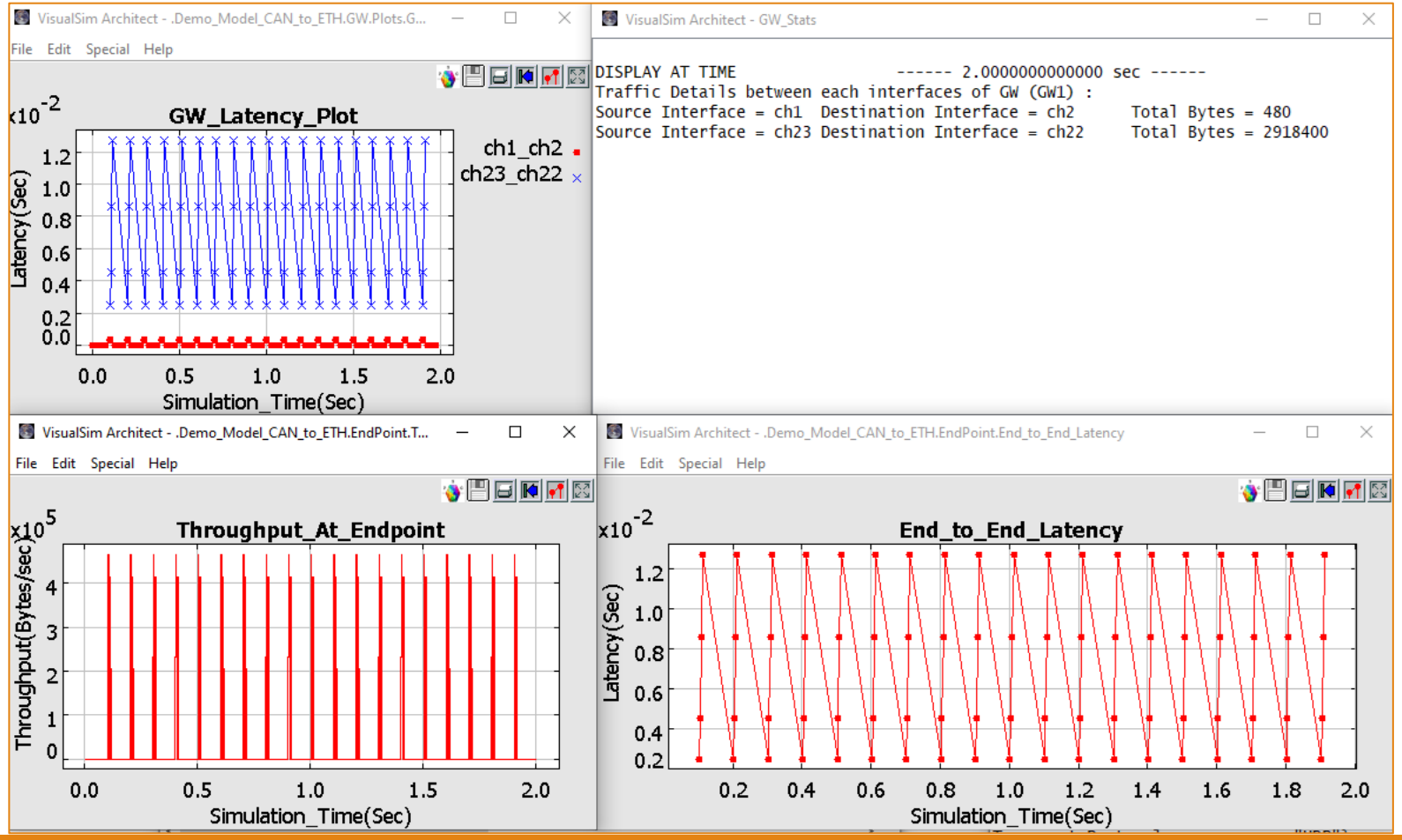
Results


VisualSim Architect - .Demo_Model_CAN_to_ETH.GW.Plots.G...
File Edit Special Help
GW_Latency_Plot
x10^-5
Latency(Sec)
Simulation_Time(Sec)
ch1_ch2
ch23_ch22
VisualSim Architect - GW_Stats
DISPLAY AT TIME ------ 2.0000000000000 sec ------
Traffic Details between each interfaces of GW (GW1) :
Source Interface = ch1 Destination Interface = ch2 Total Bytes = 480
Source Interface = ch23 Destination Interface = ch22 Total Bytes = 14592
VisualSim Architect - .Demo_Model_CAN_to_ETH.EndPoint.T...
File Edit Special Help
Throughput_At_Endpoint
x10^5
Throughput(Bytes/sec)
Simulation_Time(Sec)
VisualSim Architect - .Demo_Model_CAN_to_ETH.EndPoint.End_to_End_Latency
File Edit Special Help
End_to_End_Latency
x10^-5
Latency(Sec)
Simulation_Time(Sec)

Use cases

80% NETWORK CAPACITY

Results



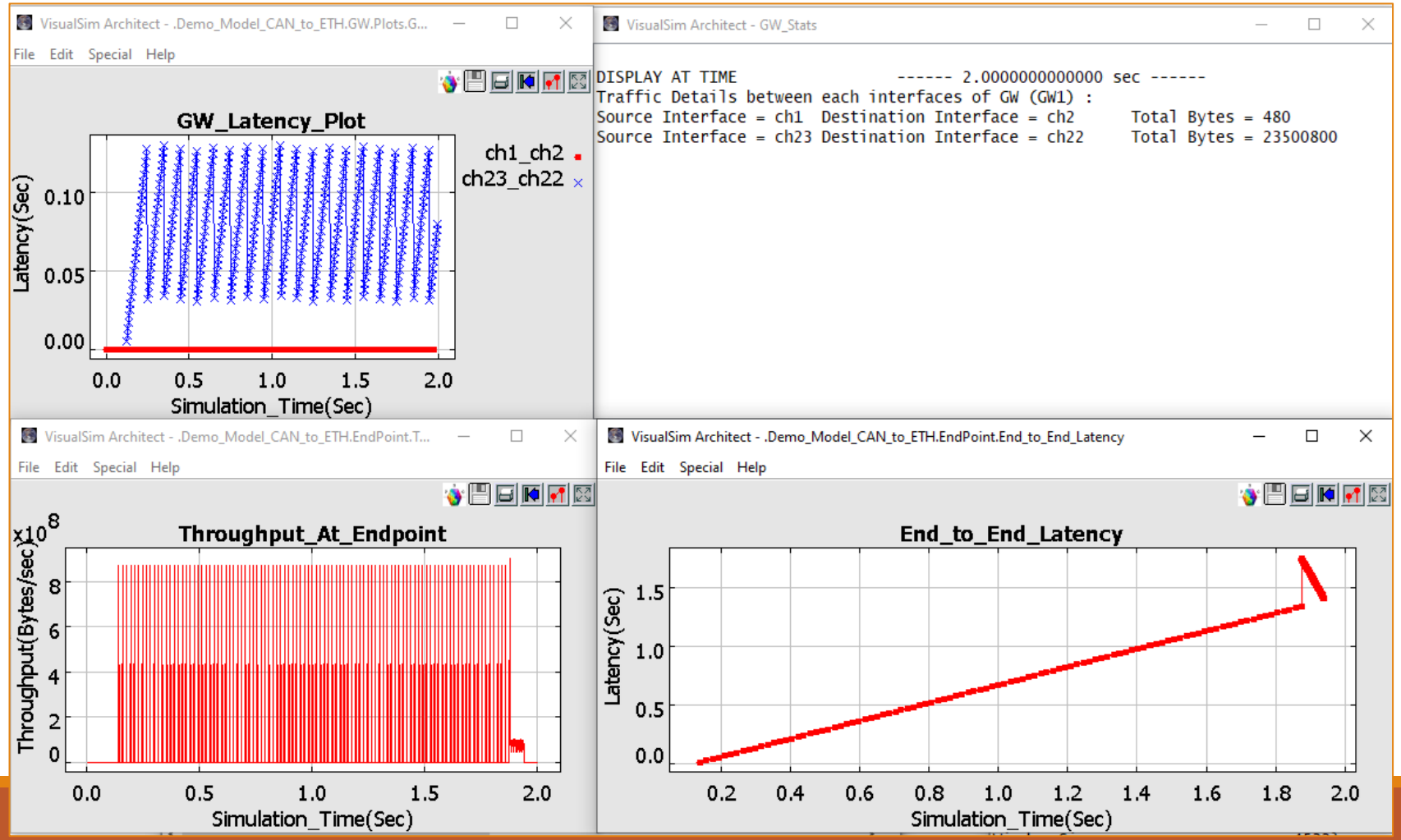
Total bytes increased

Increased Latency

Use cases

80% NETWORK CAPACITY AND TCP FRAMES

Results



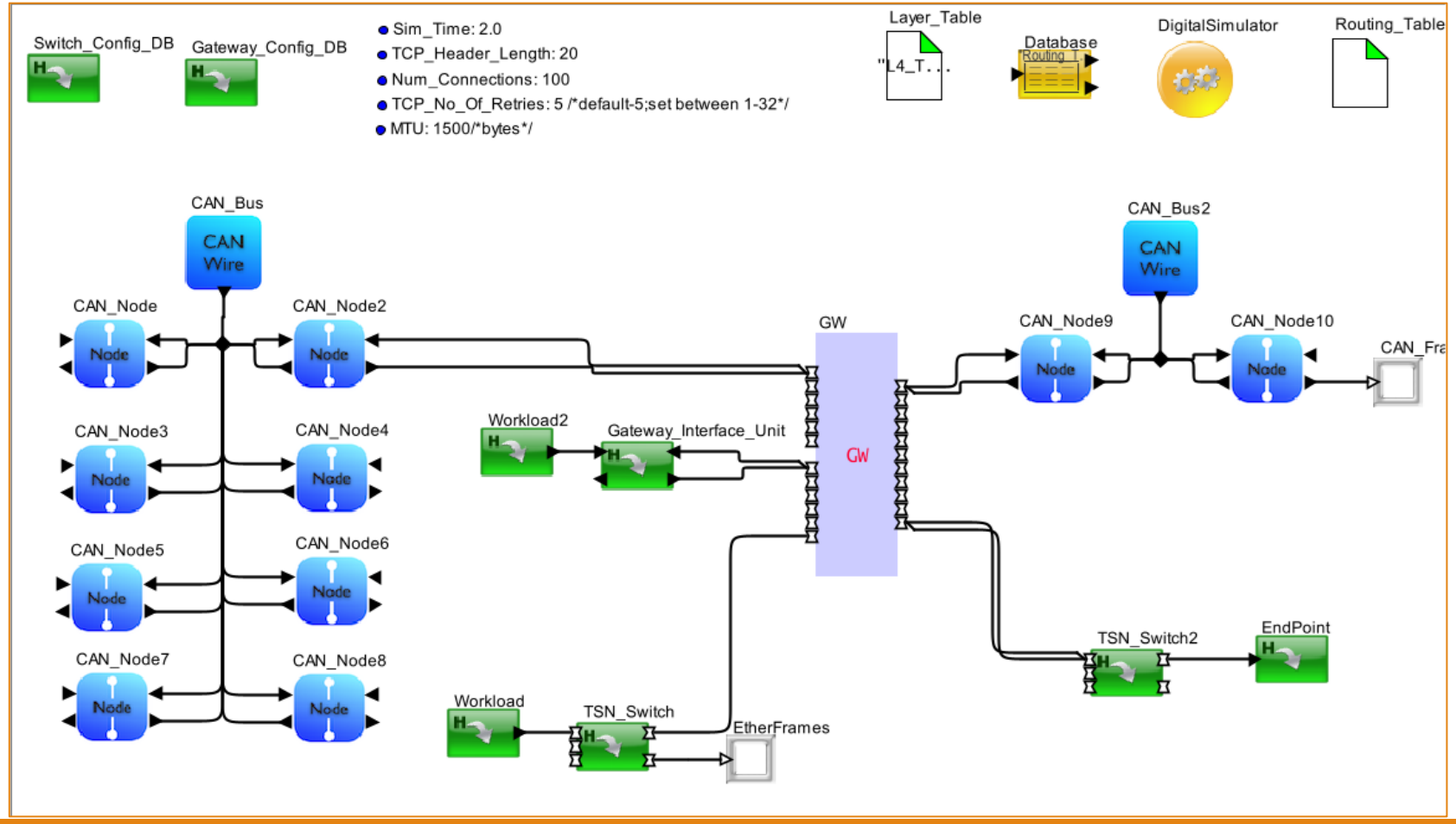
Congestion and Retransmissions

Use cases

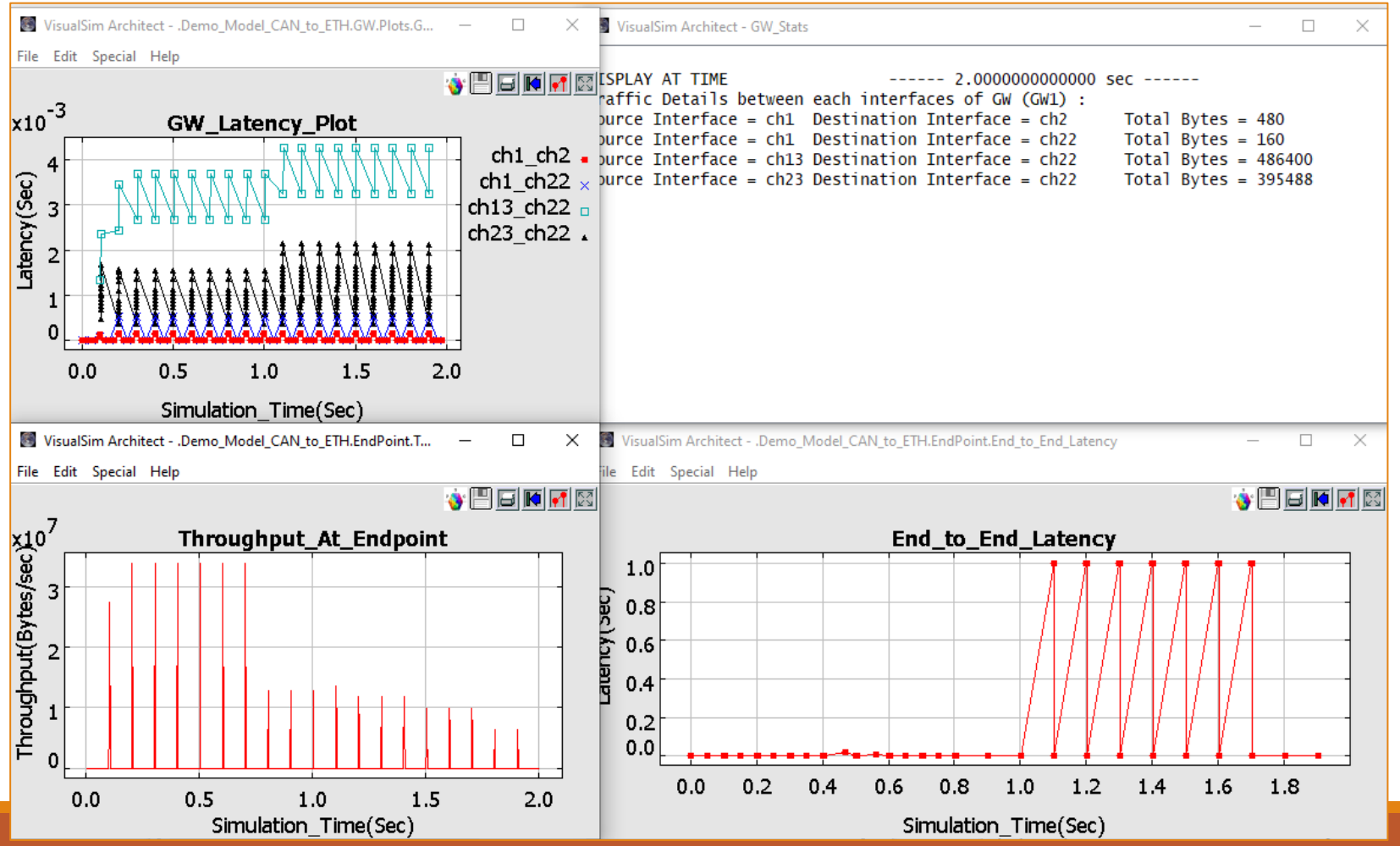
TCP, UDP AND CAN FRAMES

REARRANGING WORK LOADS

VisualSim Model



Results

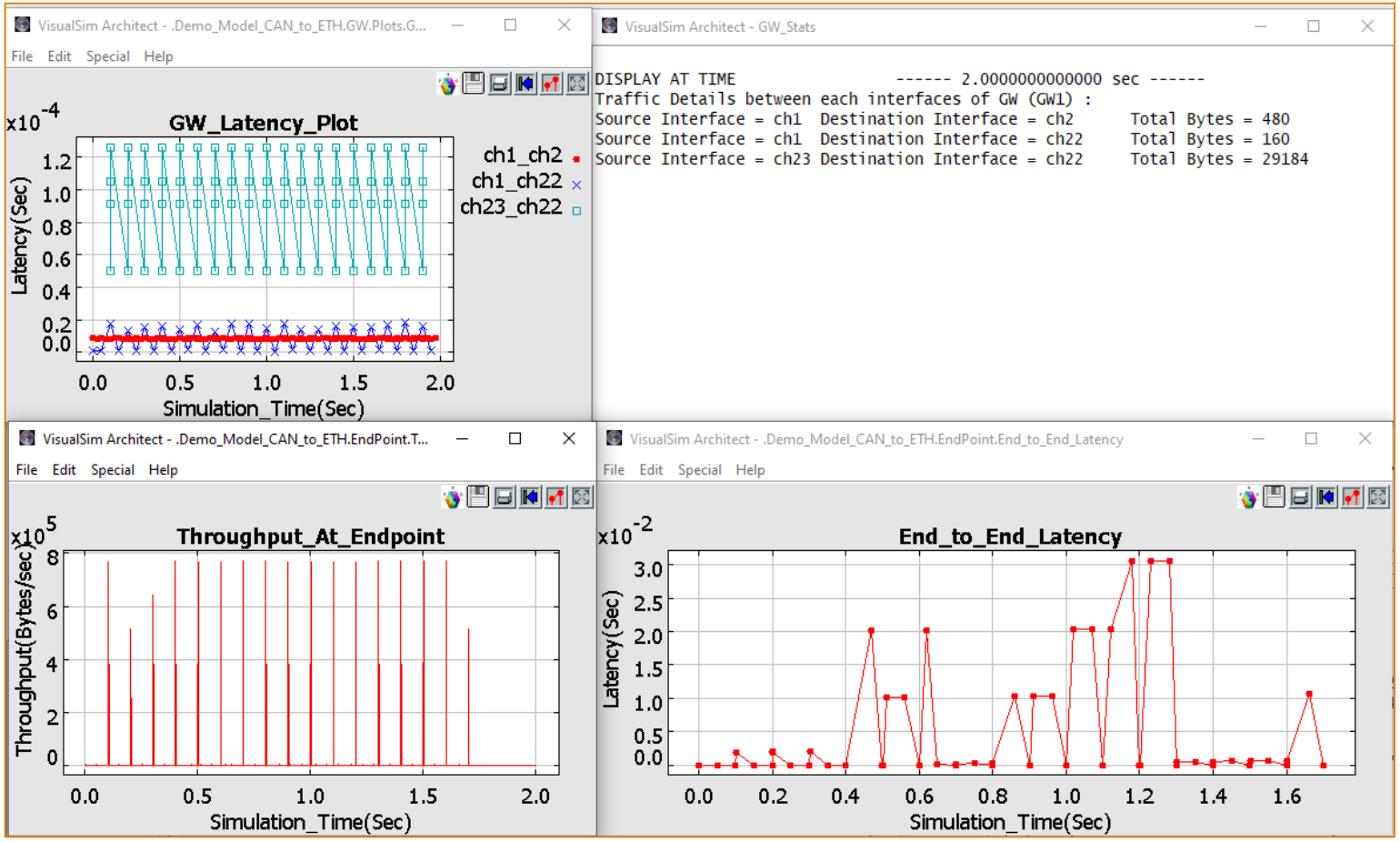

VisualSim Architect - .Demo_Model_CAN_to_ETH.GW.Plots.G...
File Edit Special Help
x10^-3
GW_Latency_Plot
Latency(Sec)
Simulation_Time(Sec)
ch1_ch2
ch1_ch22
ch13_ch22
ch23_ch22
VisualSim Architect - GW_Stats
ISPLAY AT TIME ------ 2.0000000000000 sec ------
raffic Details between each interfaces of GW (GW1) :
ource Interface = ch1 Destination Interface = ch2 Total Bytes = 480
ource Interface = ch1 Destination Interface = ch22 Total Bytes = 160
ource Interface = ch13 Destination Interface = ch22 Total Bytes = 486400
ource Interface = ch23 Destination Interface = ch22 Total Bytes = 395488
VisualSim Architect - .Demo_Model_CAN_to_ETH.EndPoint.T...
File Edit Special Help
x10^7
Throughput_At_Endpoint
Throughput(Bytes/sec)
Simulation_Time(Sec)
VisualSim Architect - .Demo_Model_CAN_to_ETH.EndPoint.End_to_End_Latency
File Edit Special Help
End_to_End_Latency
Latency(Sec)
Simulation_Time(Sec)

Use cases

TCP FRAMES

CAN TO ETHERNET

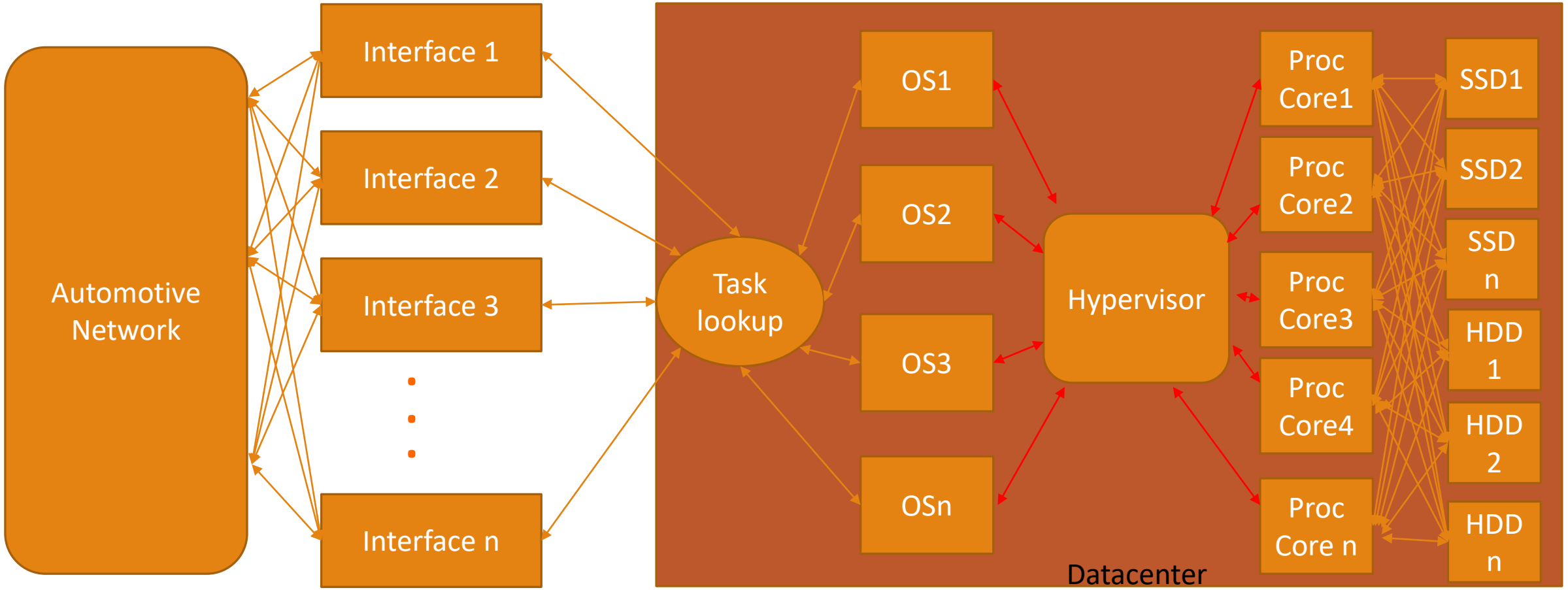
Results


VisualSim Architect - .Demo_Model_CAN_to_ETH.GW.Plots.G...
File Edit Special Help
GW_Latency_Plot
x10^-4
Latency(Sec)
Simulation_Time(Sec)
ch1_ch2
ch1_ch22
ch23_ch22
VisualSim Architect - GW_Stats
DISPLAY AT TIME ------ 2.0000000000000 sec ------
Traffic Details between each interfaces of GW (GW1) :
Source Interface = ch1 Destination Interface = ch2 Total Bytes = 480
Source Interface = ch1 Destination Interface = ch22 Total Bytes = 160
Source Interface = ch23 Destination Interface = ch22 Total Bytes = 29184
VisualSim Architect - .Demo_Model_CAN_to_ETH.EndPoint.T...
File Edit Special Help
Throughput_At_Endpoint
x10^5
Throughput(Bytes/sec)
Simulation_Time(Sec)
VisualSim Architect - .Demo_Model_CAN_to_ETH.EndPoint.End_to_End_Latency
File Edit Special Help
End_to_End_Latency
x10^-2
Latency(Sec)
Simulation_Time(Sec)

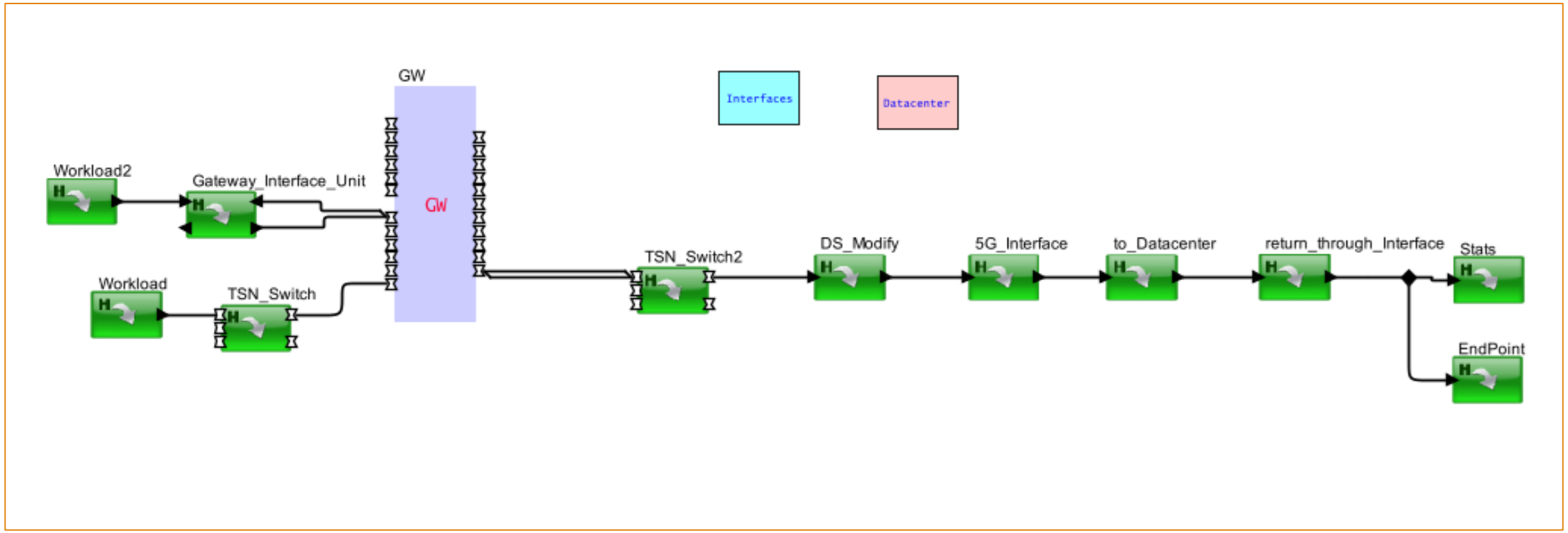
Use cases

CONNECT TO DATACENTRE

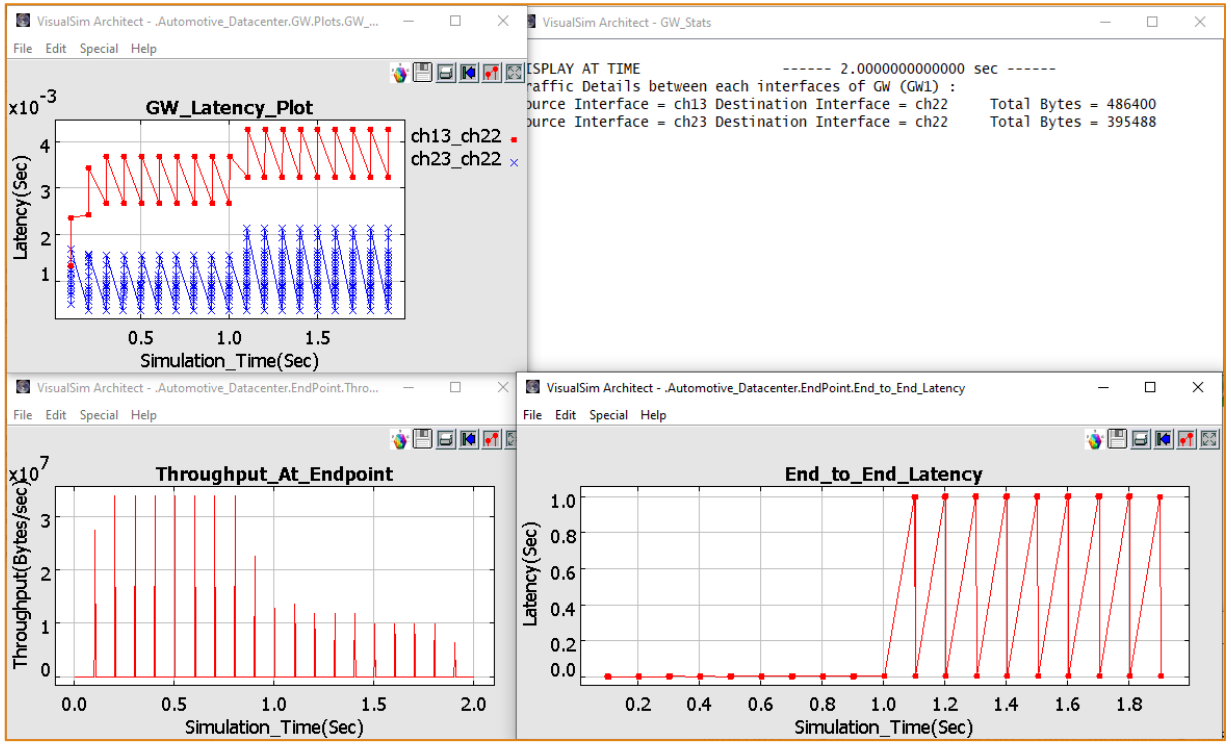
Datacenter modelling – Block diagram



VisualSim Model



Results




```
Datacenter ID = 0 :::::::::: Out of 36 Processor cores, a max of 2 were used
 Memory used by Processor Core 1 = 875356.0 Bytes
 Memory used by Processor Core 2 = 88572.0 Bytes

Interface Block ID = 1 :::::::::: Throughput To_Datacenter-

Interface 1 = 428718.0 Bytes/sec
Interface 2 = 50686.0 Bytes/sec
Interface 3 = 2560.0 Bytes/sec
Interface 4 = 0.0 Bytes/sec
Interface 5 = 0.0 Bytes/sec

Interface Block ID = 4 :::::::::: Throughput From_Datacenter-

Interface 1 = 436398.0 Bytes/sec
Interface 2 = 44034.0 Bytes/sec
Interface 3 = 1532.0 Bytes/sec
Interface 4 = 0.0 Bytes/sec
Interface 5 = 0.0 Bytes/sec

Datacenter ID = 0 :::::::::: Datacenter total Throughput = 481964.0 Bytes/sec


Datacenter ID = 0 :::::::::: SSD Memory Remaining = 102400.0 GB and HDD Memory Remaining = 204799.999036072 GB
```

VISUALSIM TRAINING