MLDesigner Workshop

Mission Level Design GmbH
04.12./05.12.2007

Dipl.-Inf. Thomas Lohfelder

Agenda

- Introduction
- Modeling Concepts
- Application Examples
- Modeling Projects
- Discussion

MLDesigner: Introduction

- Modeling and simulation System
- Support for different modeling domains (Domains)
  - Discrete Time (static and dynamic data flow): SDF, DDF
  - Continuous Time: CTDE
  - Discrete Event: DE
  - Hybrid
- Modeling paradigms
  - Hierarchical block diagrams
  - FSM / Statecharts primitives
  - ptlang (C++) primitives
- Code generation
  - ANSI C Code
  - VHDL

Agenda

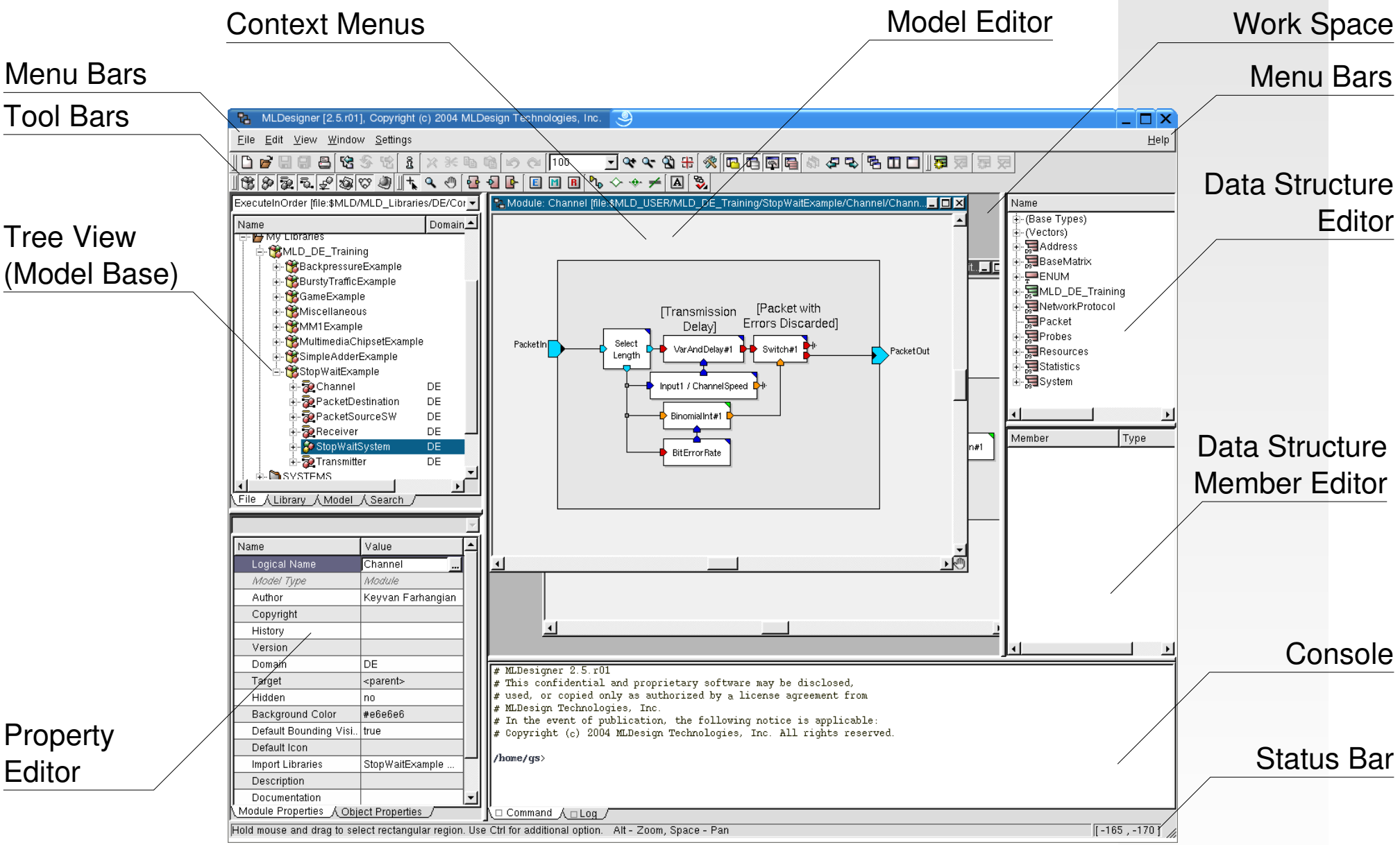
- Introduction
- Modeling Concepts
- Application Examples
- Modeling Projects
- Discussion

MLDesigner: Model Types

- Library
  - container for all other model types
  - may contain sub-libraries
- System
  - structure defines function
    - set of instances
    - connected via ports
  - doesn't contain formal ports
  - doesn't contain external arguments
  - executable (simulatable)
- Module
  - structure defines function
    - set of instances
    - connected via ports
  - may define formal ports
  - may define external arguments
  - not executable
- Primitive
  - atomic model type
  - doesn't contain model structure
  - function defined in ptlang / FSM
  - may define formal Ports
  - may define external arguments
  - not executable

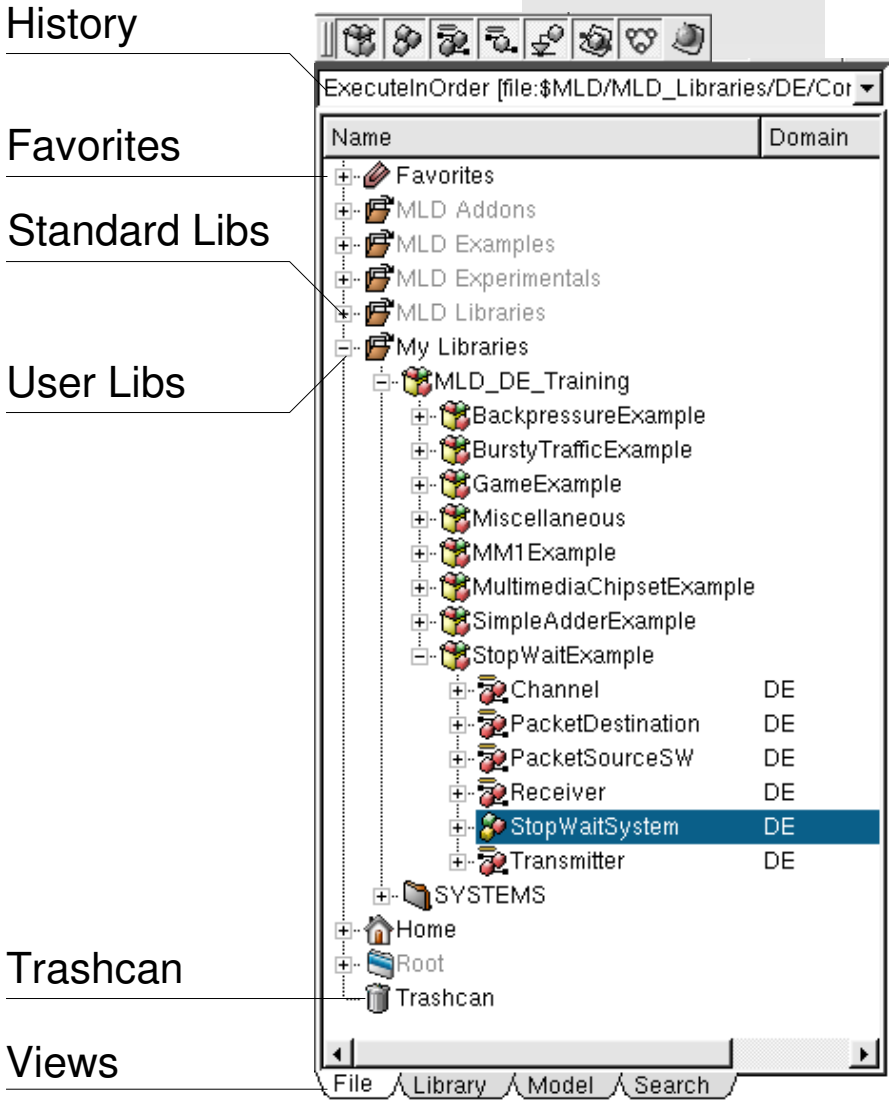
MLDesigner: GUI

Mission Level Design GmbH



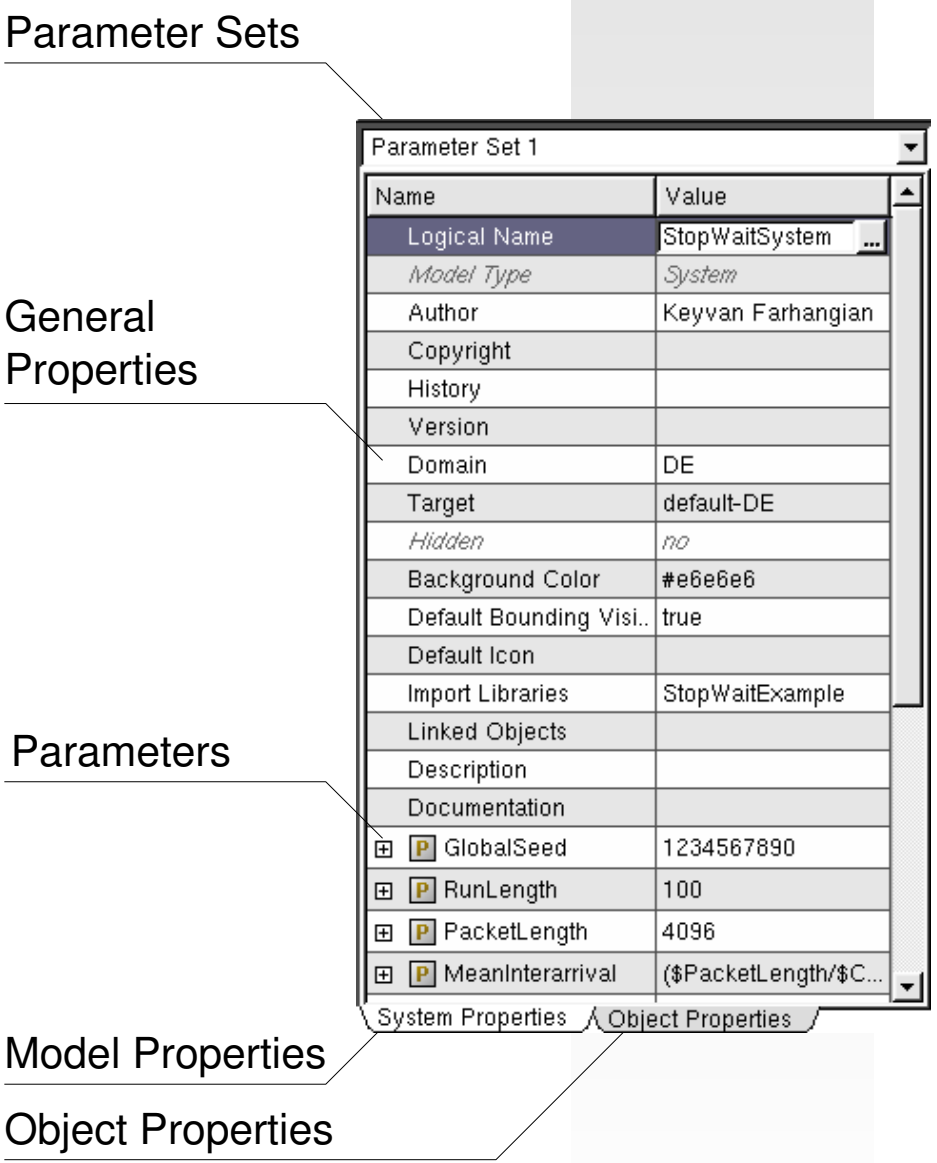
MLDesigner: Tree View

- Tree View
  - different views on the model base
  - additional functions
- Views
  - File: physical structure of model base
  - Library: logical structure of model base
  - Model: structure of opened models
  - Search: search in model base
- Additional functions
  - Model view filter (filter buttons)
  - History of last used models
  - Favorites
  - Trashcan with restore function
  - Import/Export of top-level Libraries



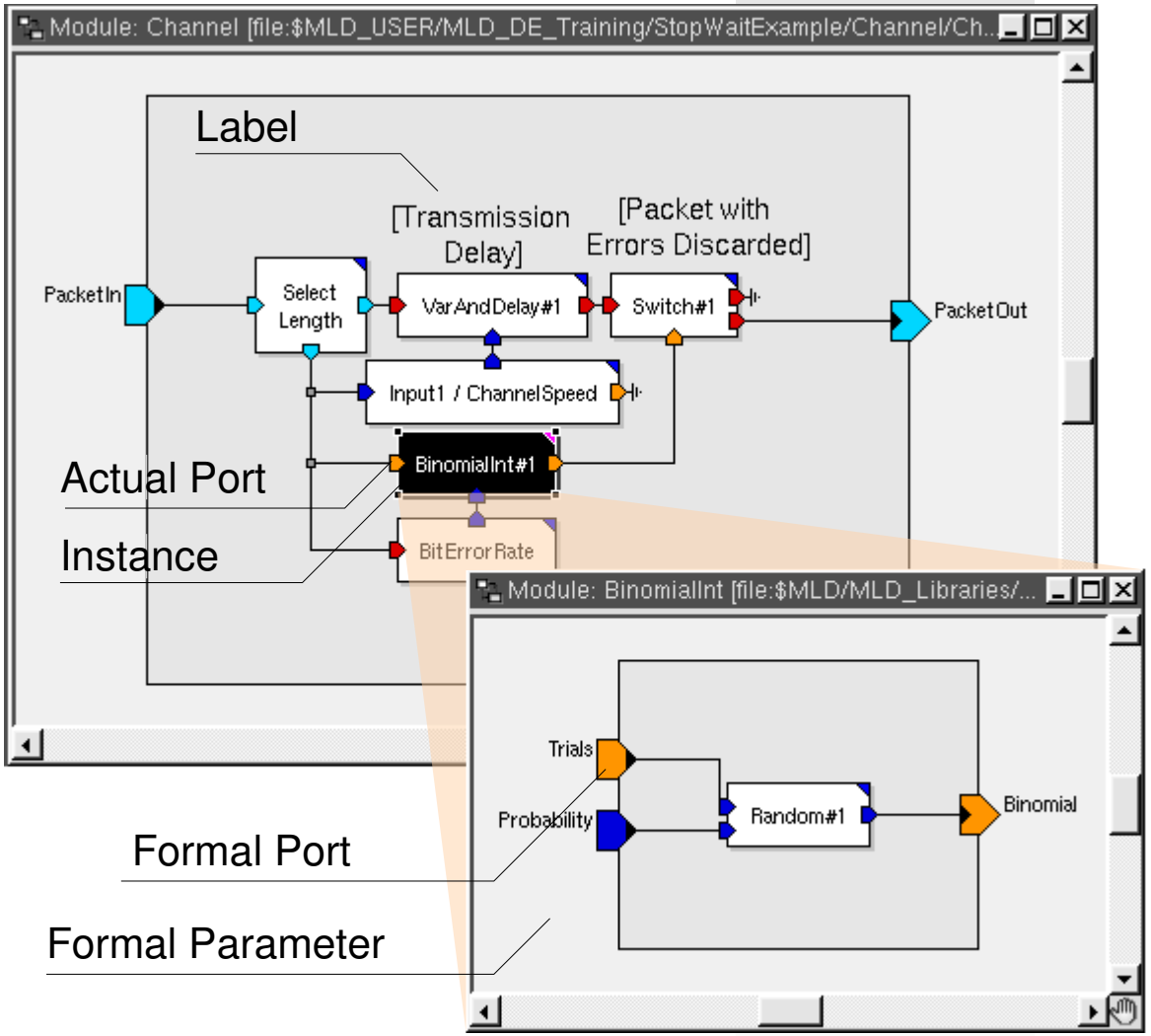
MLDesigner: Property Editor

- Properties
  - Model (System, Module, ...)
  - Selected Objects
  - Simulation
- Property Editor
  - General properties
  - Parameter properties
  - Memory properties
  - Event properties
  - Resource properties
- Parameter Sets
  - different sets of system parameters
  - parameter values are varied in different simulation runs



MLDesigner: Model Structure

- Model
  - hierarchical
  - describes structure and function
    - Instances
    - Relations
- Model Interface
  - formal ports
  - formal parameters
  - external arguments
    - Events
    - Memories
    - Resources

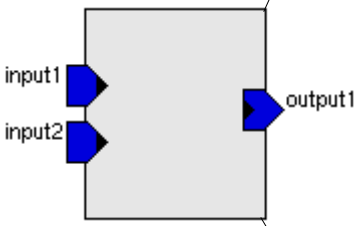


MLDesigner: Formal vs. Actual Model Element

- Formal Model Element
  - element of model definition
  - defines type and defaults
  - defines the model interface that is visible from outside
- Formal definition of
  - ports
  - external parameters
  - external arguments
    - Memories
    - Events
    - Resources
- Actual Model Element
  - element of an instance of a model
  - assignment of final value / usage
  - results from the model interface that is visible from outside
- Value assignment / Usage
  - connection of ports
  - fixed values or calculation for external parameters
  - linking of external arguments

MLDesigner: Primitives (1)

- ptlang Primitive
  - consists of
    - interface description
    - ptlang source code
  - compiled into C++ code
  - support inheritance
  - specialized editor



Source: $MLD_USER/MLD_DE_Training/SimpleAdderExample/Simple_Adc

File Edit Tools Settings Help

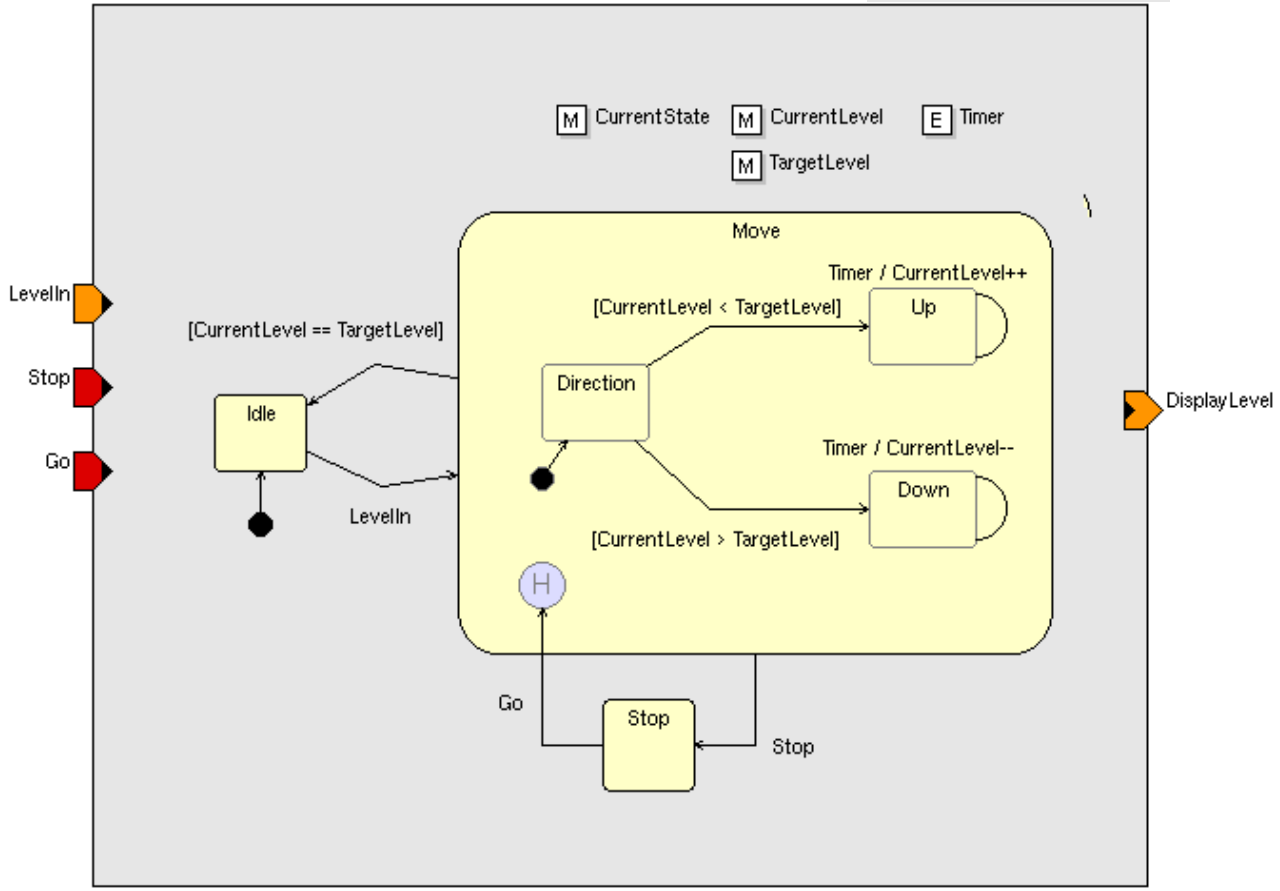
```
defprimitive
{
  // !!! The following items are generated automatically.
  // !!! Changes of these items are lost after the modifi-
  // !!! cation of the primitive model

  name        { Simple_Adder }
  domain      { DE }
  author      { Keyvan Farhangian }
  version     { 0.0 06/17/2002 }

  input
  {
    name { input1 }
    type { float }
  }

  input
  {
    name { input2 }
    type { float }
  }

  output
  {
    name { output1 }
    type { float }
  }

  // !!! The following items are untouched by primitive
  // !!! model modifications. You can change them as
  // !!! you want.

  go
  {
    if ( input1.dataNew && input2.dataNew )
    {
      output1.put(arrivalTime) << ((double)input1.get() + (double)input2.get());
    }
  }
}
```

Modified Line: 41 Column: 1

- ptlang Language
  - C++ preprocessor language
  - complete C++ allowed
  - external C++ code can be used

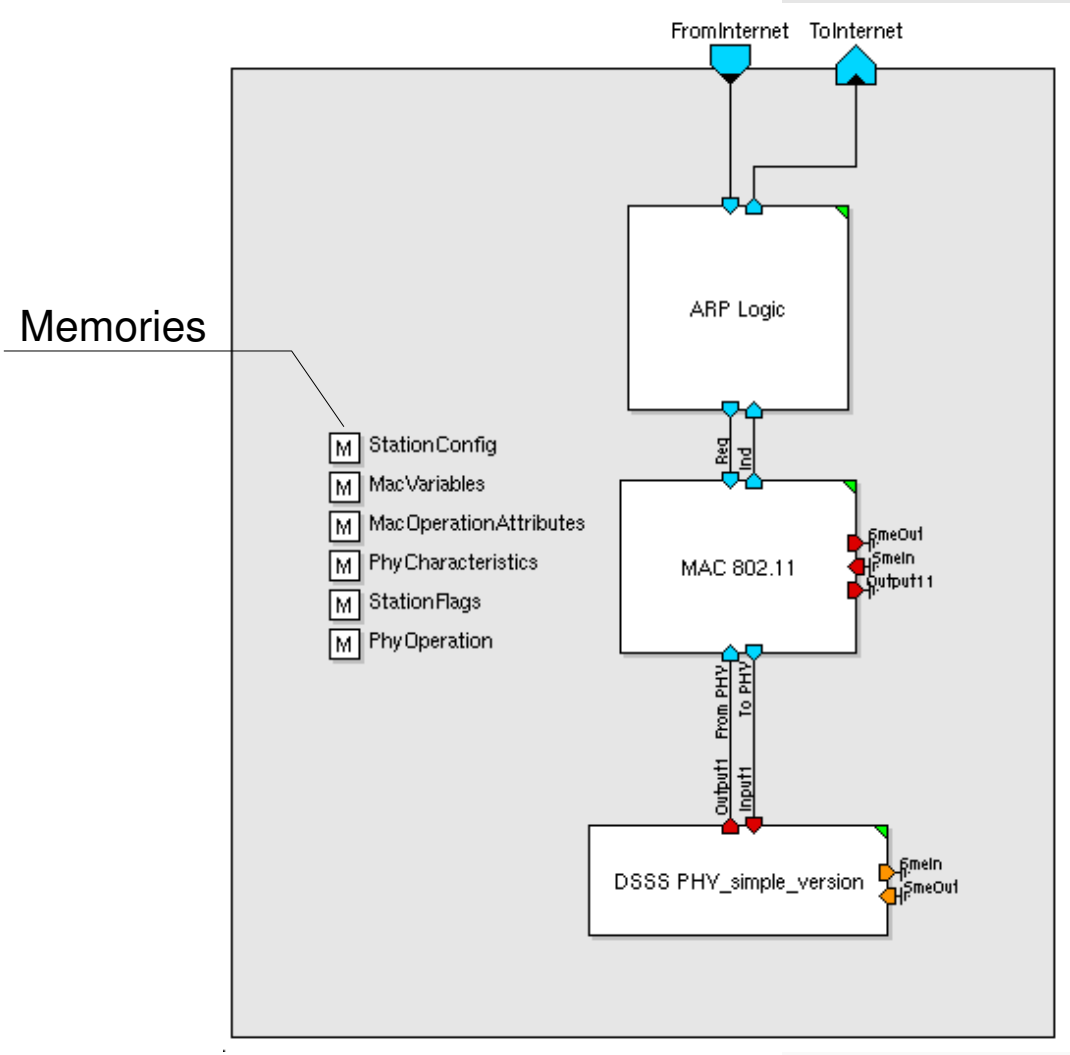
MLDesigner: Primitives (2)

- FSM Primitive
  - consists of
    - interface definition
    - Statechart
  - compiled into C++
- FSM / Statechart Primitive
  - Statechart compatible
  - hierarchical states
  - Slave models
  - C++ in actions



MLDesigner: Memories

- Memory
  - used to share information between MLDesigner models
  - can be linked over several model hierarchy levels
- Access
  - various built-in models for memory access are shipped with MLDesigner
  - easy to access from within MLDesigner primitives
- Support Memory Active Read



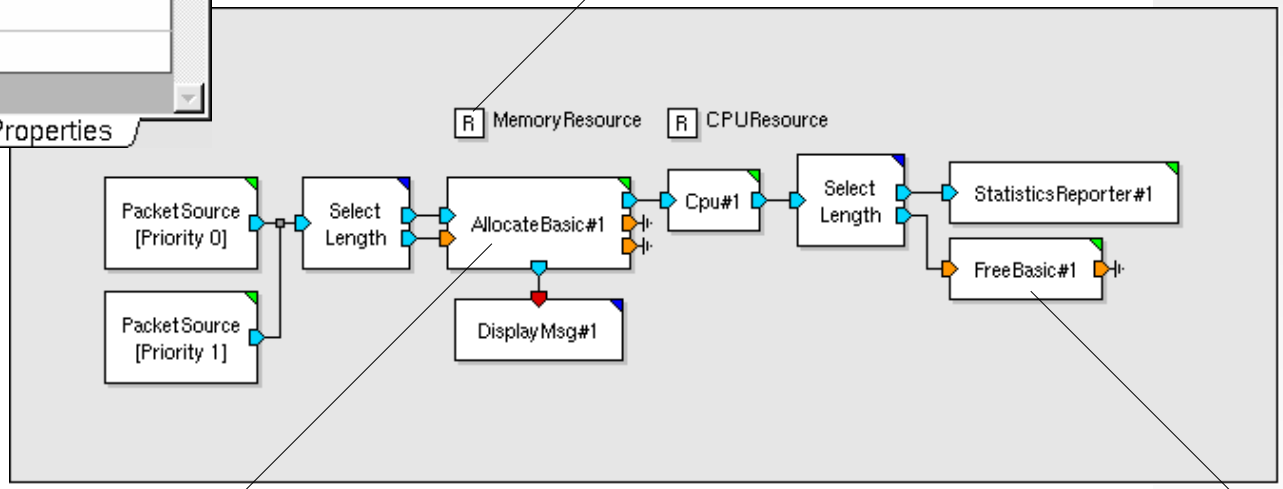
MLDesigner: Resources (1)

Mission Level Design GmbH

Name	Value
MemoryReso...	
Name	MemoryResource
Type	Quantity
Scope	*Internal*
Description	
Number of Dimens...	1
Initial Capacity	100000
Maximum Queue ...	100
Blocking Mechani...	Wait_for_Resource
Queue Discipline	First_In_First_Out
Queue Reject Me...	Incoming_DS_Rejected
Addressing Mode	NonIndexed
Addresssed Fit P...	FirstFit
Annotation	

System Properties | Resource Properties

Quantity Ressource



Allocation

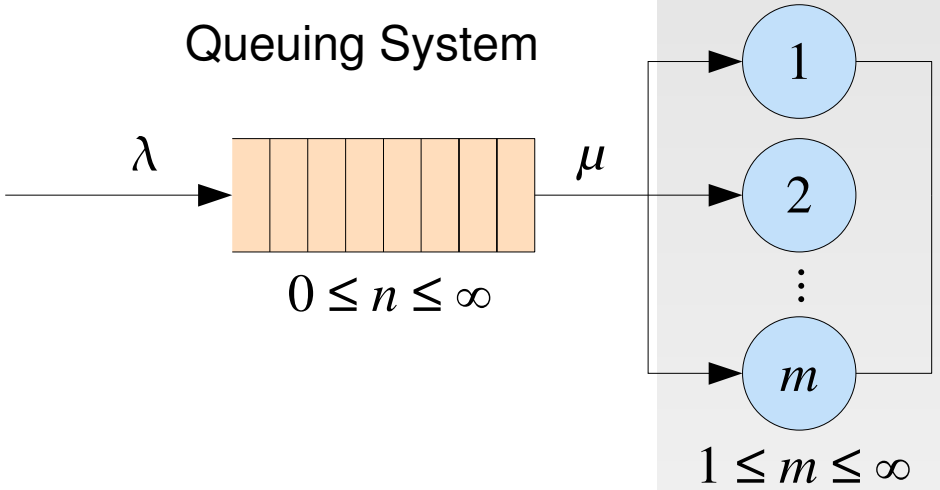
Release

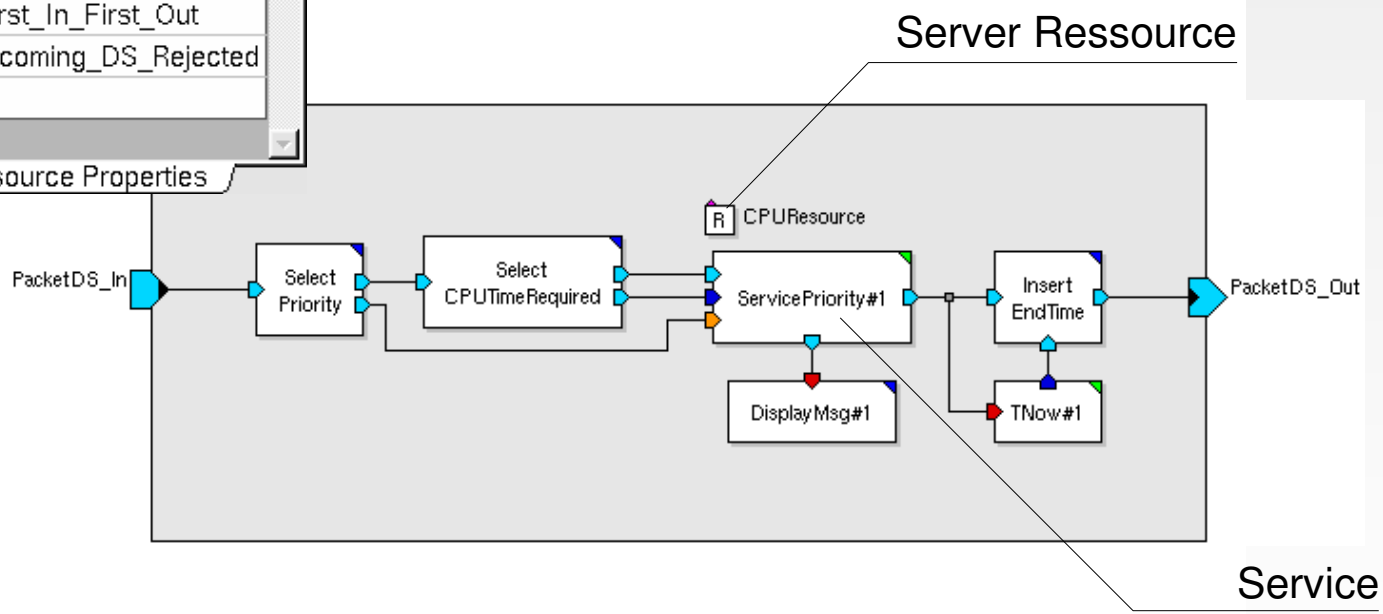
MLDesigner: Resources (2)

Mission Level Design GmbH

Name	Value
CPUResource	
Name	CPUResource
Type	Server
Scope	*Internal*
Description	
Number of Dimens...	1
Initial Number of S...	10
Initial Service Rat...	1.0
Server Mechanism	Dedicated_Server
Maximum Occupa...	100
Context Switching ..	0.01
Preempt Discipline	Allow_Preemption
Queue Discipline	First_In_First_Out
Queue Reject Me...	Incoming_DS_Rejected
Annotation	

System Properties | Resource Properties

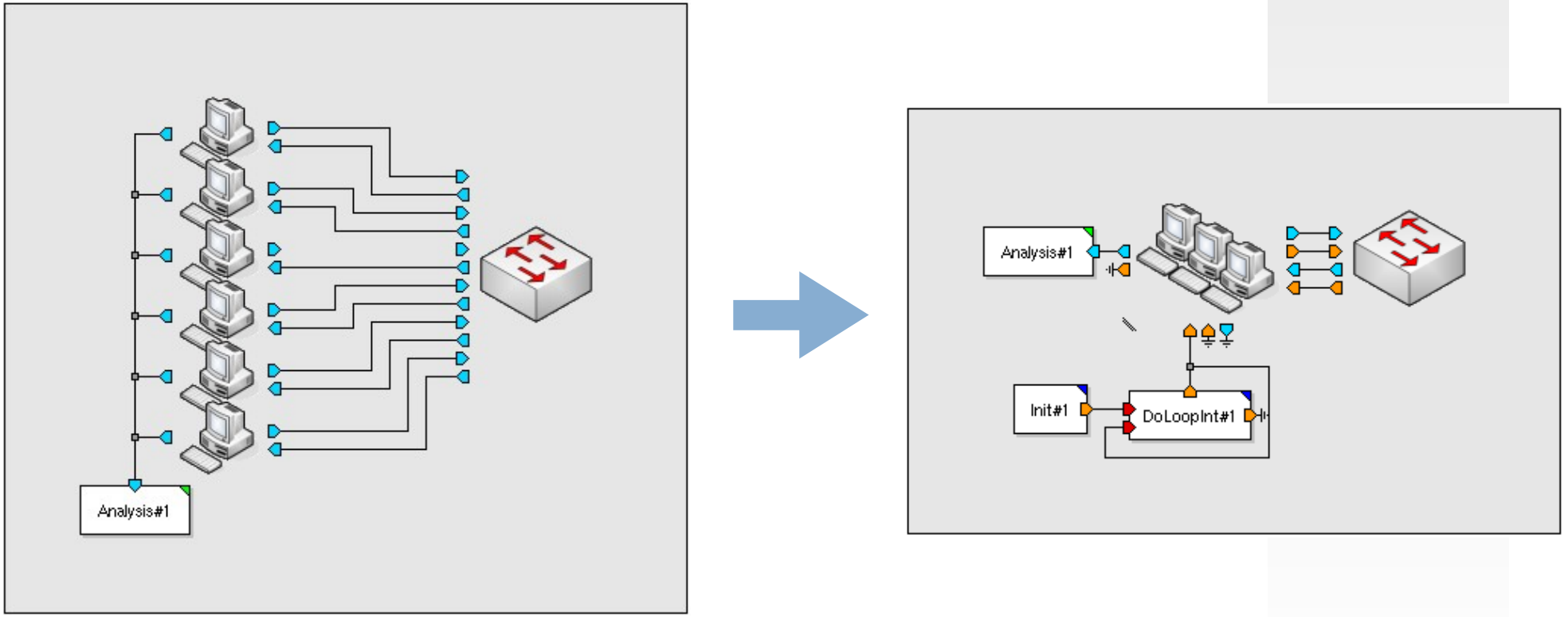




MLDesigner: Dynamic Instances

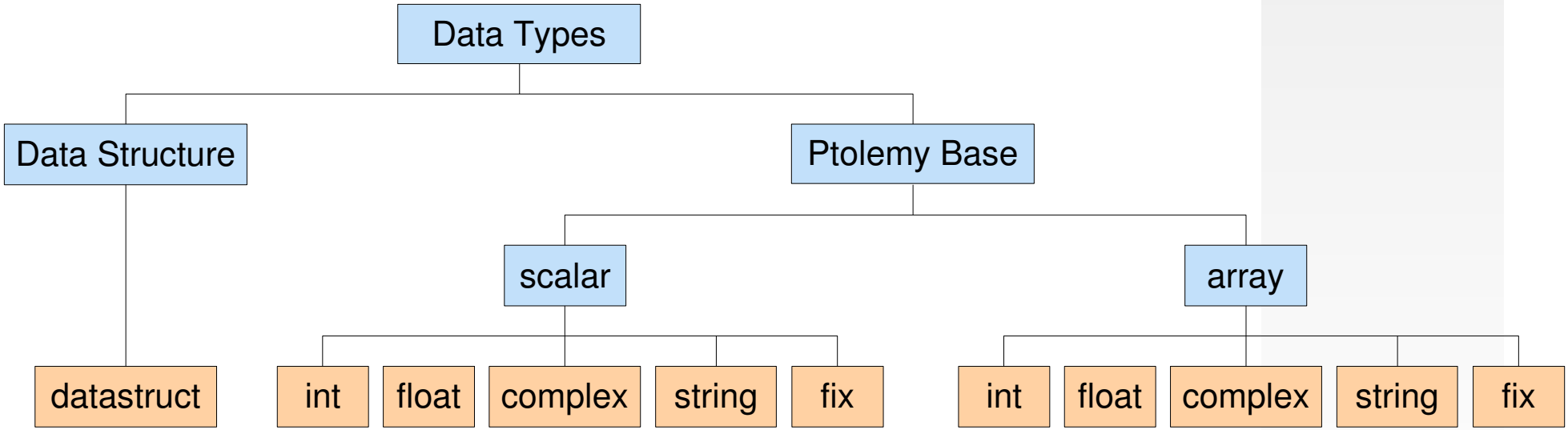
Mission Level Design GmbH

- Dynamic number of instances
  - handling a big number of instances
  - changing the number of instances during simulation



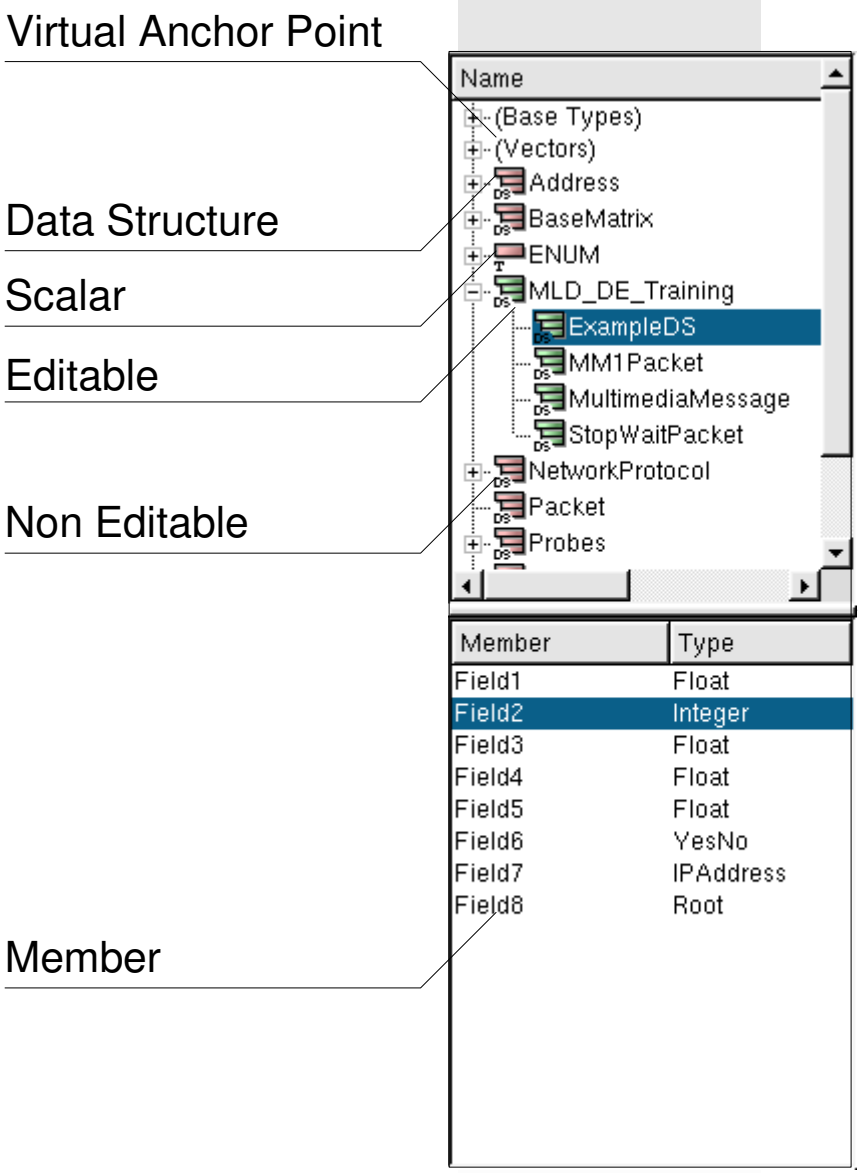
MLDesigner: Data Types

- Classes of data types
  - Ptolemy Base Types
  - Data Structures
- Ptolemy Base Types
  - direct mapping onto C types
  - efficient
- Data Structures
  - definition of structured data
  - allow paradigms of OOP



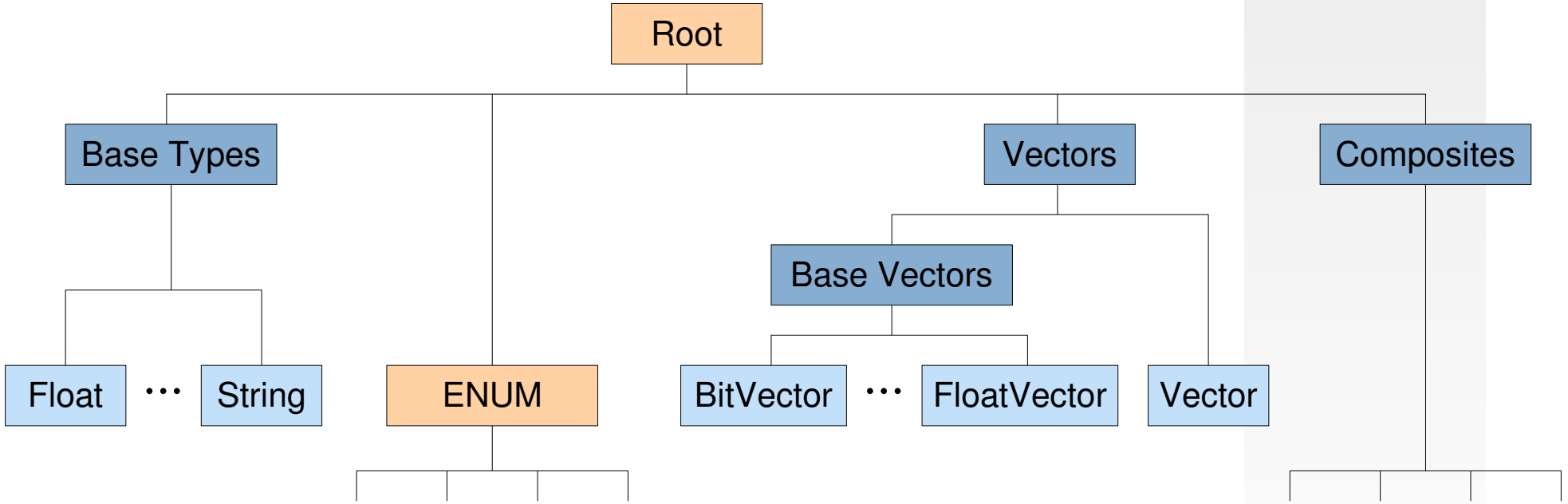
MLDesigner: Data Structures (1)

- Data Structure
  - consists of
    - Name
    - Parent (member of parents are inherited)
    - Members (of type data structure)
  - modeled within Data Structure Editor
  - stored in related library
  - can be member of other data structures
- Data Structure Editor
  - shows the data structure hierarchy
  - contains members and elements



MLDesigner: Data Structures (2)

- Data Stucture Tree
  - results from inheritance (from Root, ENUM, Composite and derivatives)
  - Full Name: Root.MyDataStructure.MyDerivedDatastructure
  - Unique Name: MyLibrary:Root.MyDataStructure.MyDerivedDatastructure



MLDesigner: Debugging (1)

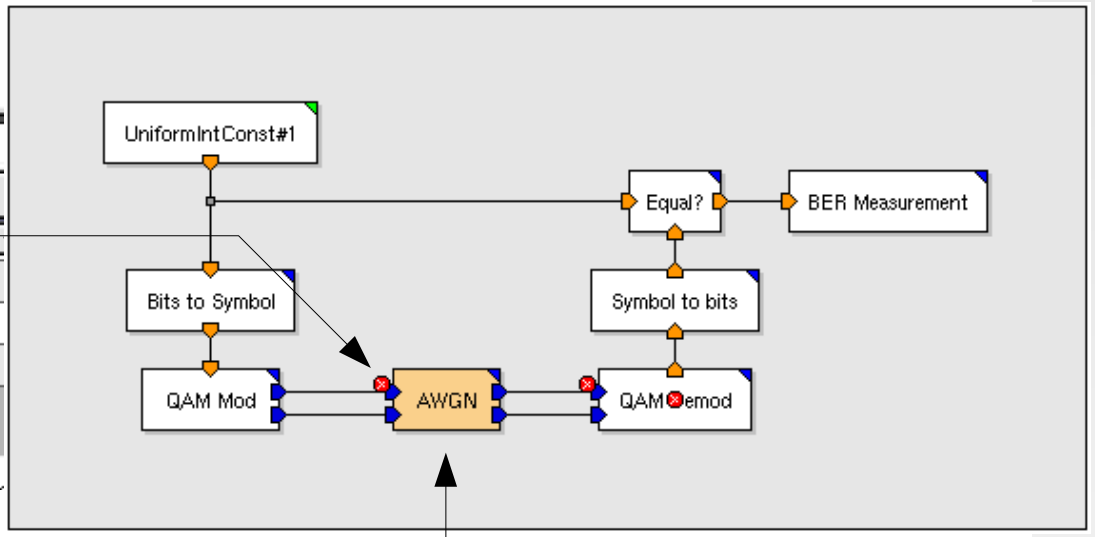
Mission Level Design GmbH

Breakpoints

Breakpoint#1

Name	Value
Enable	YES
Module Breakpoint	YES
Source Models	
Ignore Count	0

System Properties | Breakpoint Properties



Graphical Animation

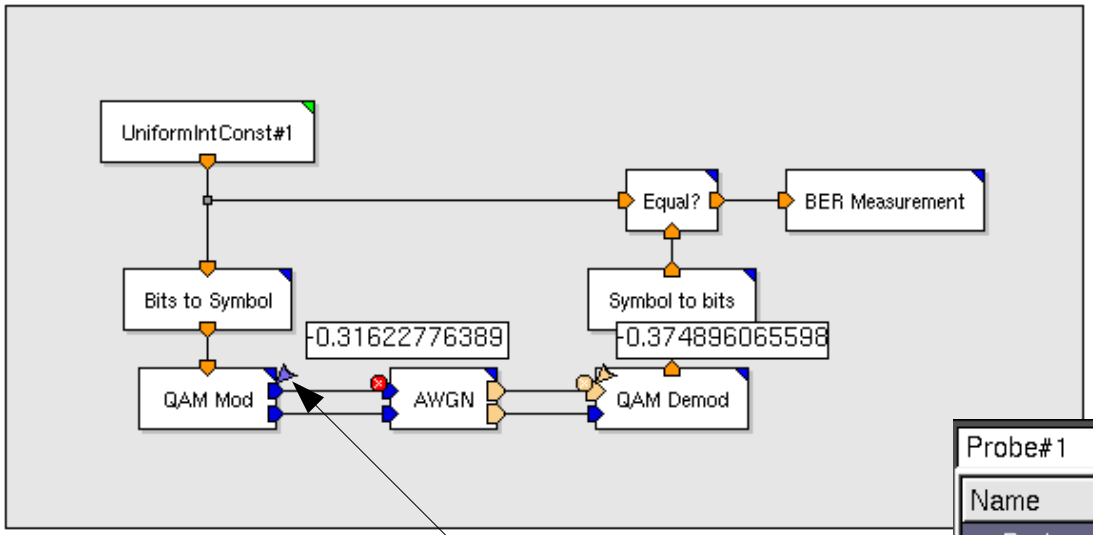
Simulation Control

Textual Animation

Iteration	Time	Entity	Name	Action	Value	Fellow
1	1.0	Instance	BitErrorRate.BitsToInt#1	will now execute		
1	1.0	Port	BitErrorRate.BitsToInt#1.output	send	11	
1	1.0	Port	BitErrorRate.QamMod#1.inSymbol	received	11	
1	1.0	Instance	BitErrorRate.QamMod#1	will now execute		
1	1.0	Port	BitErrorRate.QamMod#1.outReal	send	-0.31622776389122009	
1	1.0	Port	BitErrorRate.QamMod#1.outImag	send	-0.94868326187133789	
1	1.0	Port	BitErrorRate.AWGNCxRI#1.InReal	received	-0.31622776389122009	
1	1.0	Port	BitErrorRate.AWGNCxRI#1.InImag	received	-0.94868326187133789	
1	1.0	Instance	BitErrorRate.AWGNCxRI#1	will now execute		

Command | Log | Progress | Breakpoints | Animation

MLDesigner: Debugging (2)



Probes

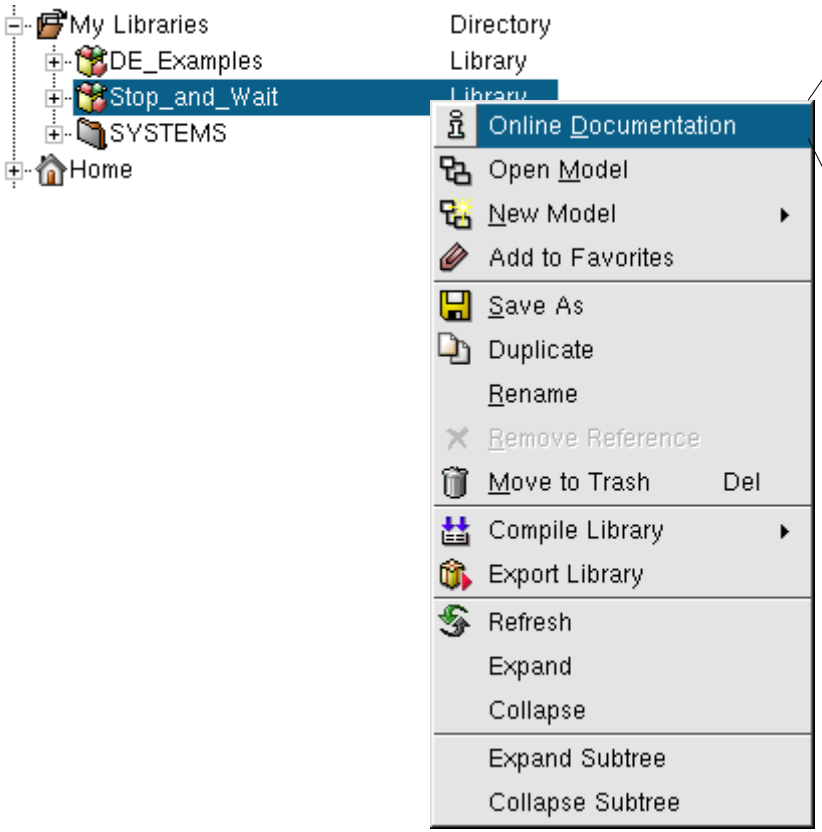
Probe#1

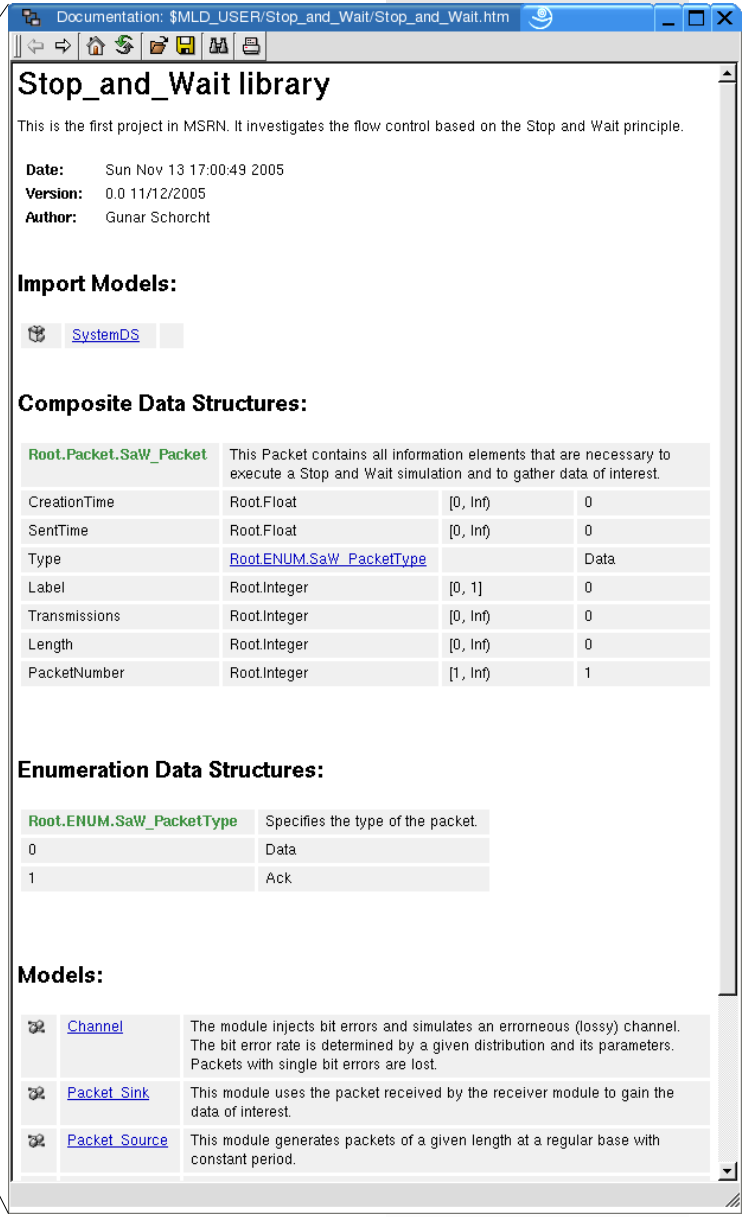
Name	Value
Probe	$MLD/MLD_Libraries
Source Model	BitErrorRate.QamMod...
Display	NO
File	NO
File Path	
Workspace	NO
Workspace Varia...	
Control Type	Time
⊞ Start Time	0
Stop Time	*$stopTime*
⊞ *Ignore Count*	*0*
⊞ *Trigger Count*	*1*

System Properties | Probe Properties

MLDesigner: Online Documentation

- Hypertext Online Documentation
  - generated automatically
  - references between documents
  - exportable





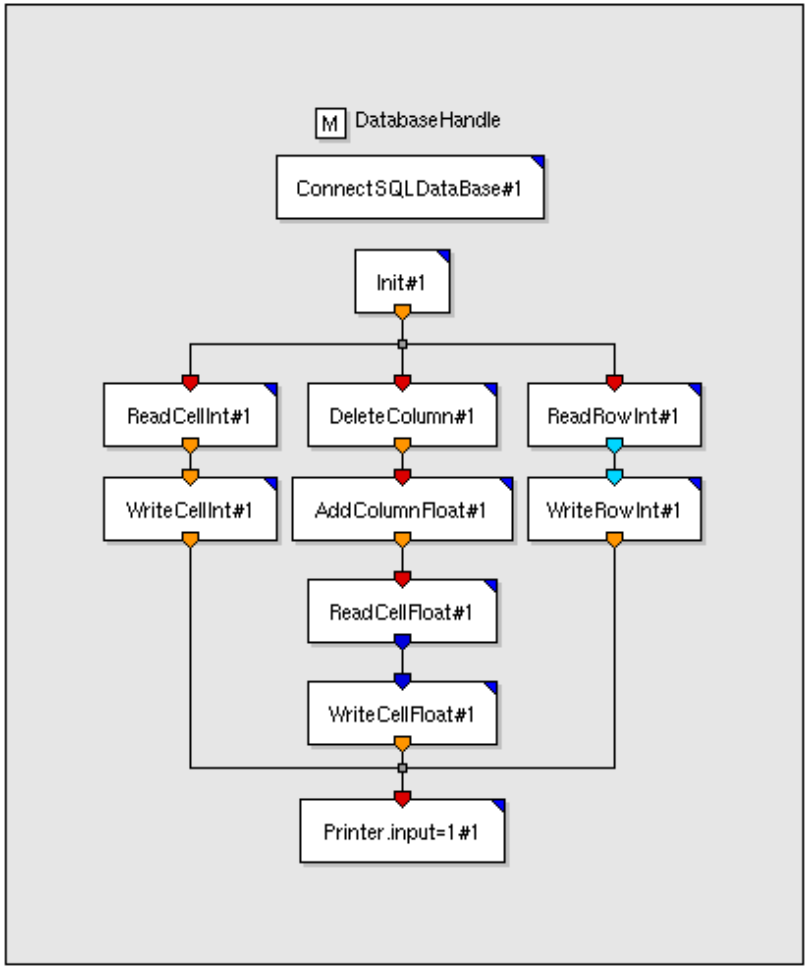
MLDesigner: MySQL-Interface

Integration

- Access to MySQL implemented in MLDesigner primitives using the C++ API
- MLDesigner includes various built-in models to operate on MySQL data bases

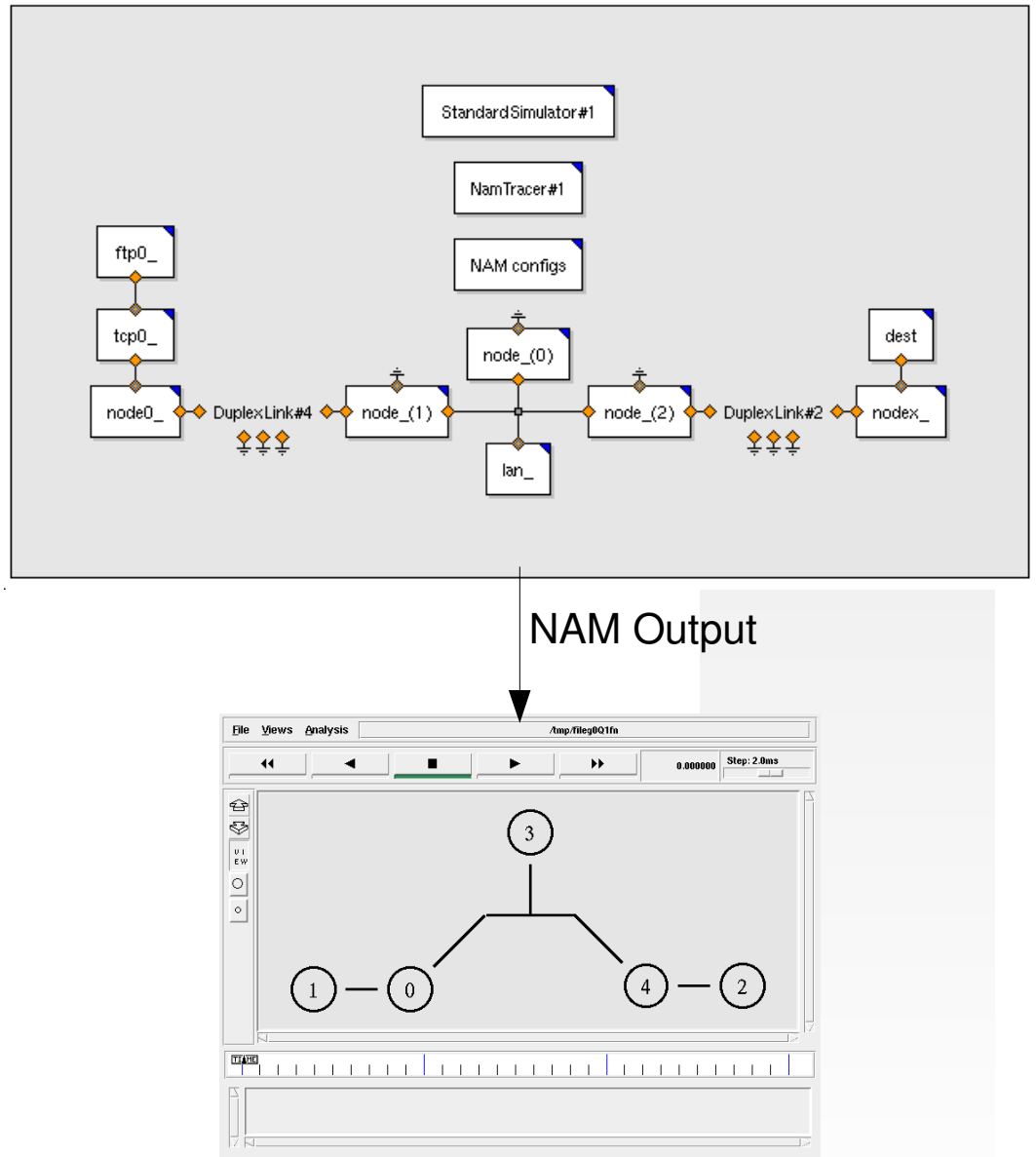
Benefits

- access to widely-spread data base format
- possibility to read simulation input values
- possibility to write simulation results



MLDesigner: NS2-Interface

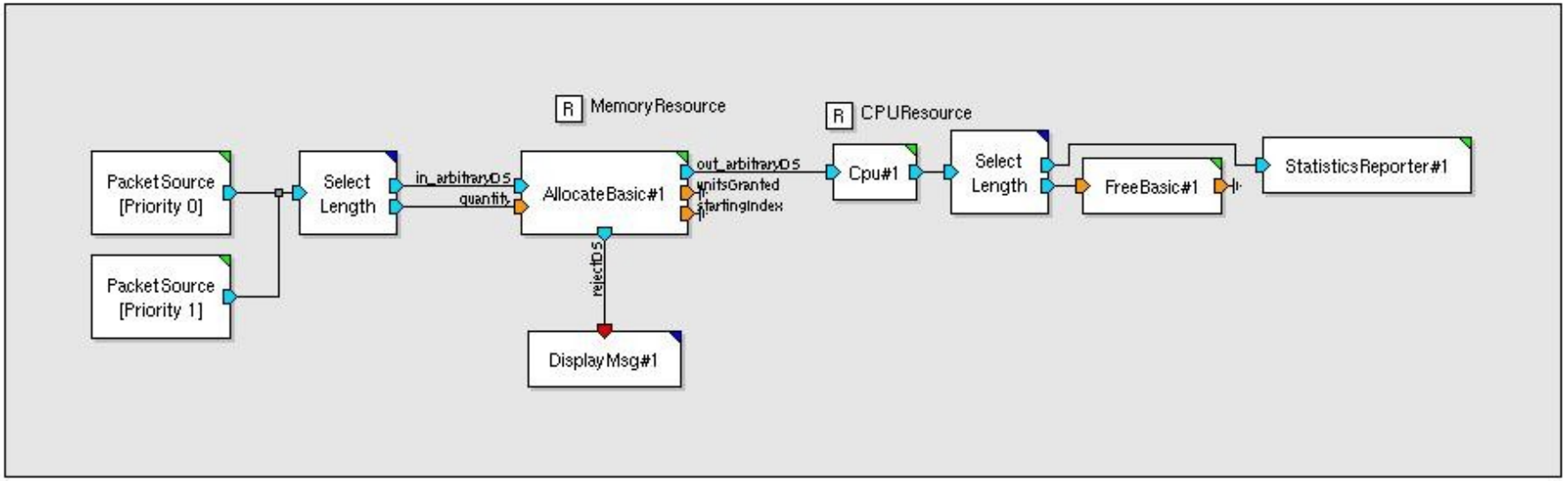
- Integration
  - MLDesigner creates, launches and controls the execution of NS2 models
  - NS2 is used as a co-simulator
- Benefits
  - very loose binding between MLDesigner and NS2 allows much flexibility on both sides
  - possibility to combine models from other MLDesigner domains with NS2 models

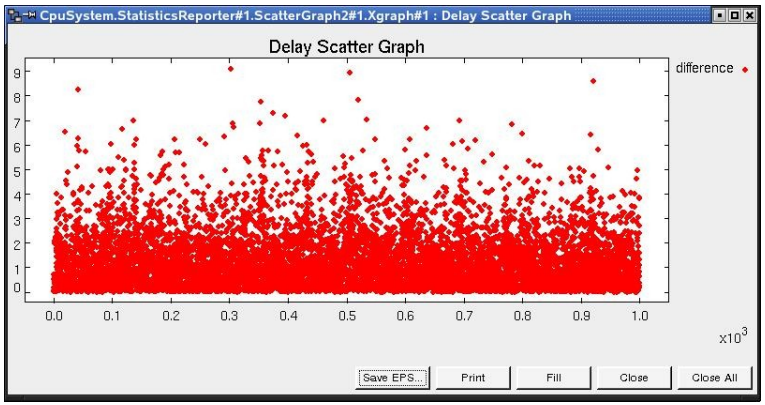


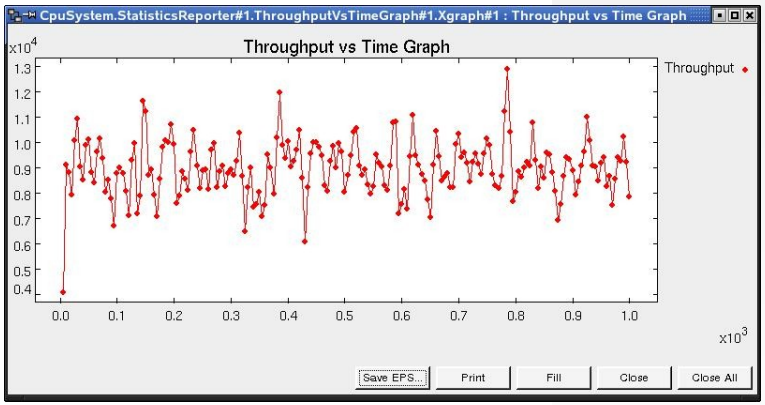
Agenda

- Introduction
- Modeling Concepts
- Application Examples
- Modeling Projects
- Discussion

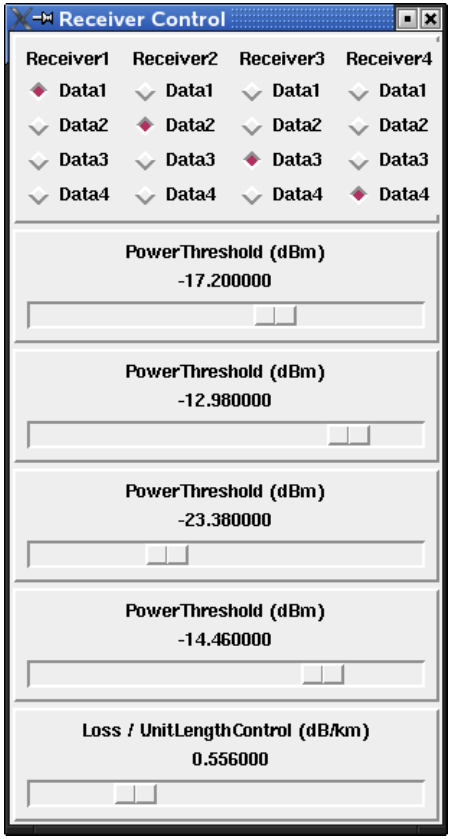
Application Examples – Cpu Demo

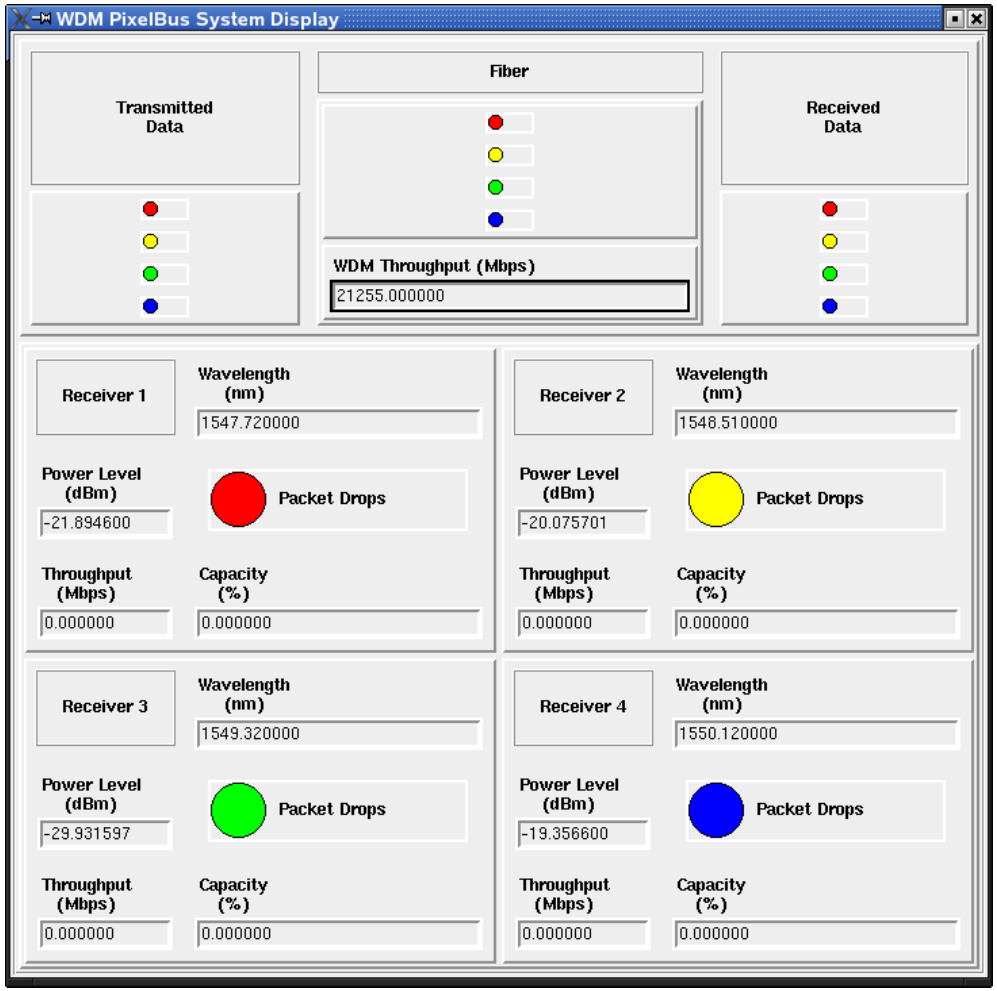


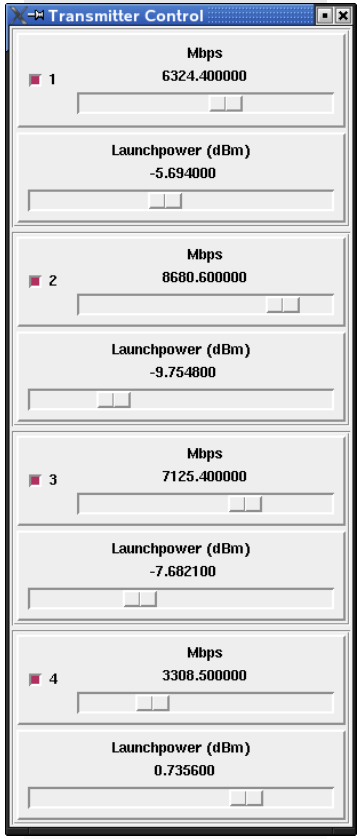




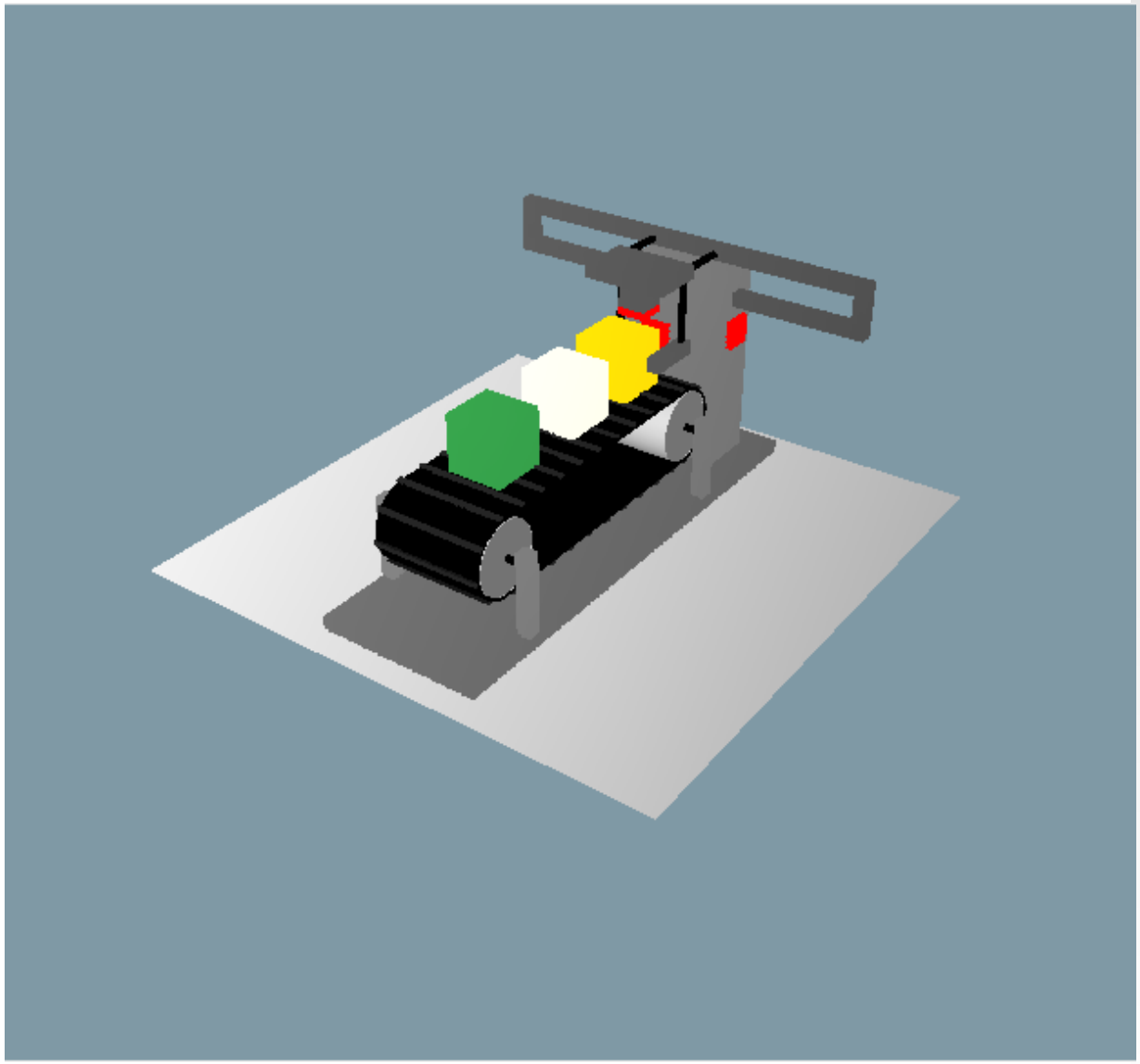
Application Examples – Lion



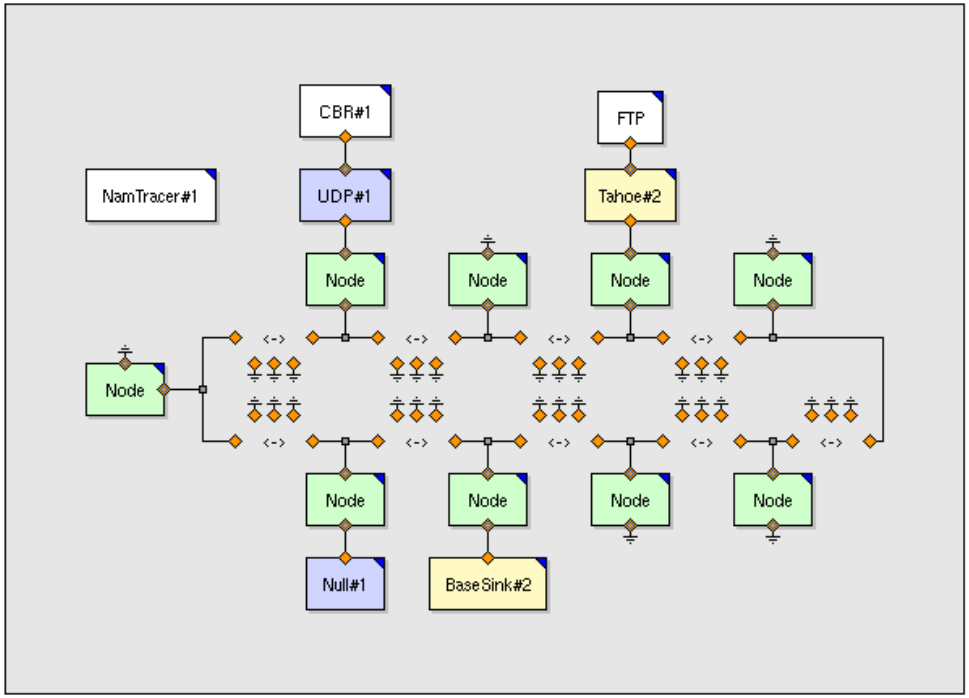


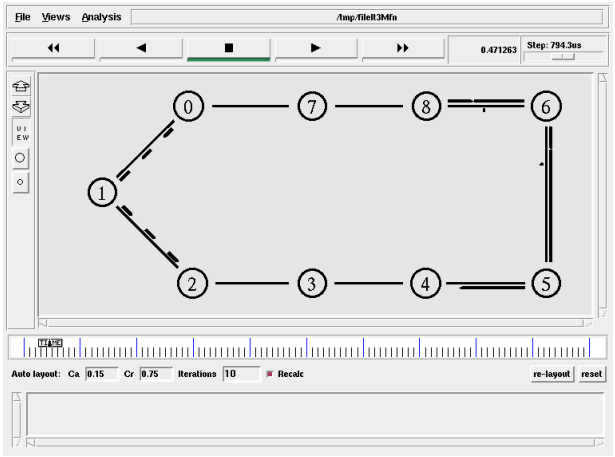


Application Examples – Block Sorter

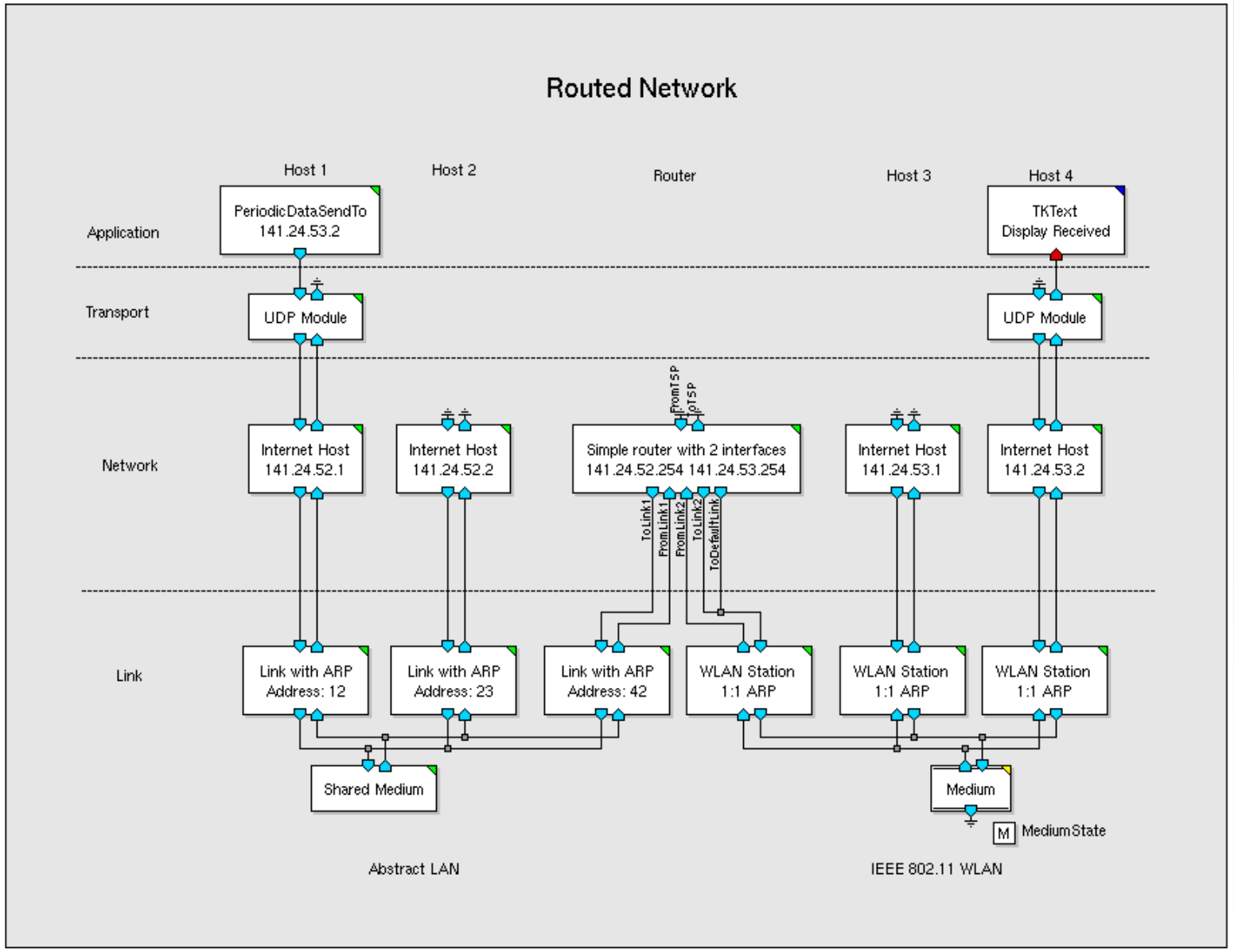


Application Examples – NS2 Demos





Application Examples – Network Building Set



Application Examples

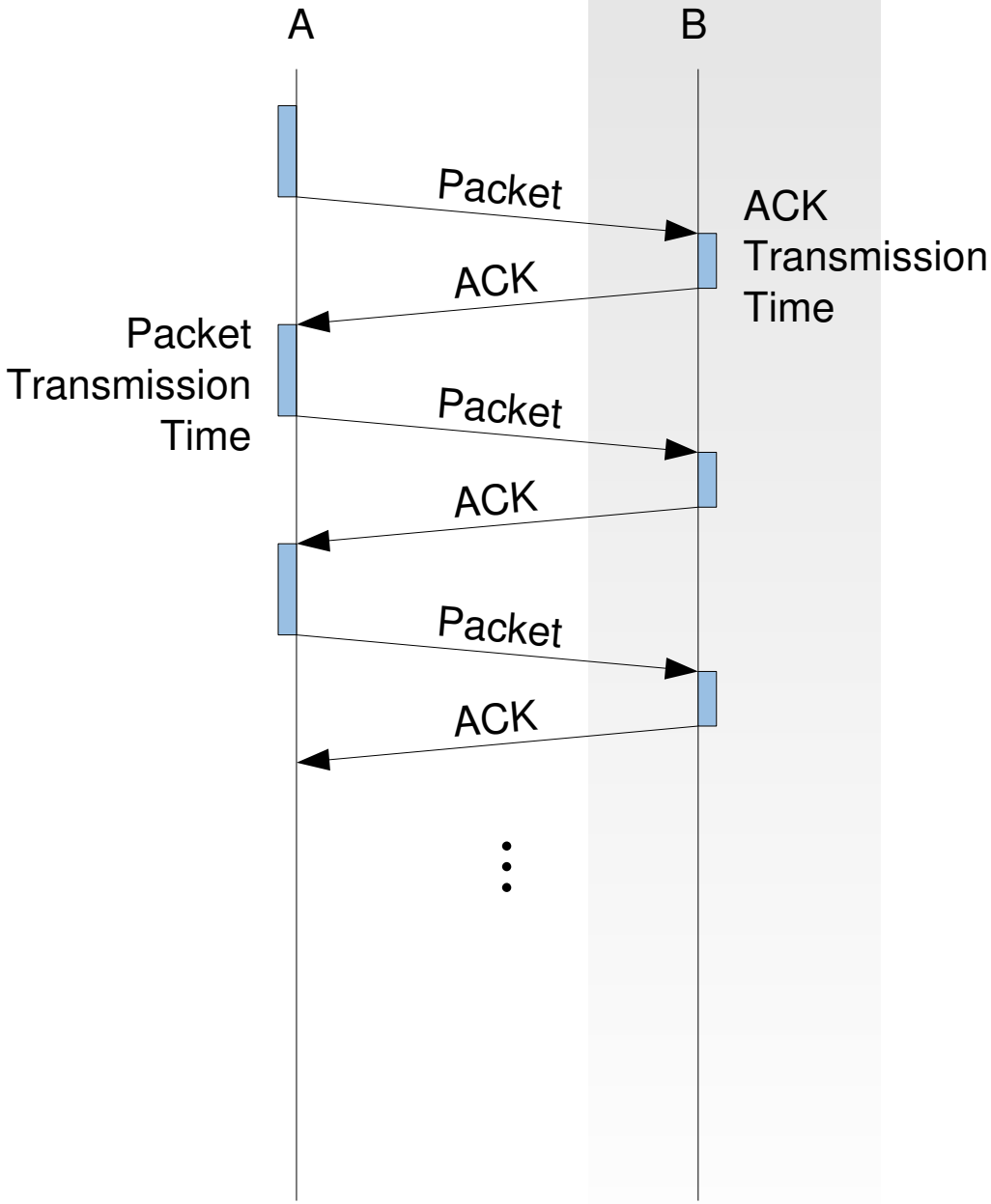
- Network simulation
  - Ethernet / WLAN
  - FlexRay
  - CAN Bus
  - AFDX
  - Zigbee
  - TTP
- Process simulation
  - clinical
  - production
  - development
- Optimization
  - genetic algorithms
  - simulated annealing
  - Monte Carlo simulation

Agenda

- Introduction
- Modeling Concepts
- Application Examples
- Modeling Projects
- Discussion

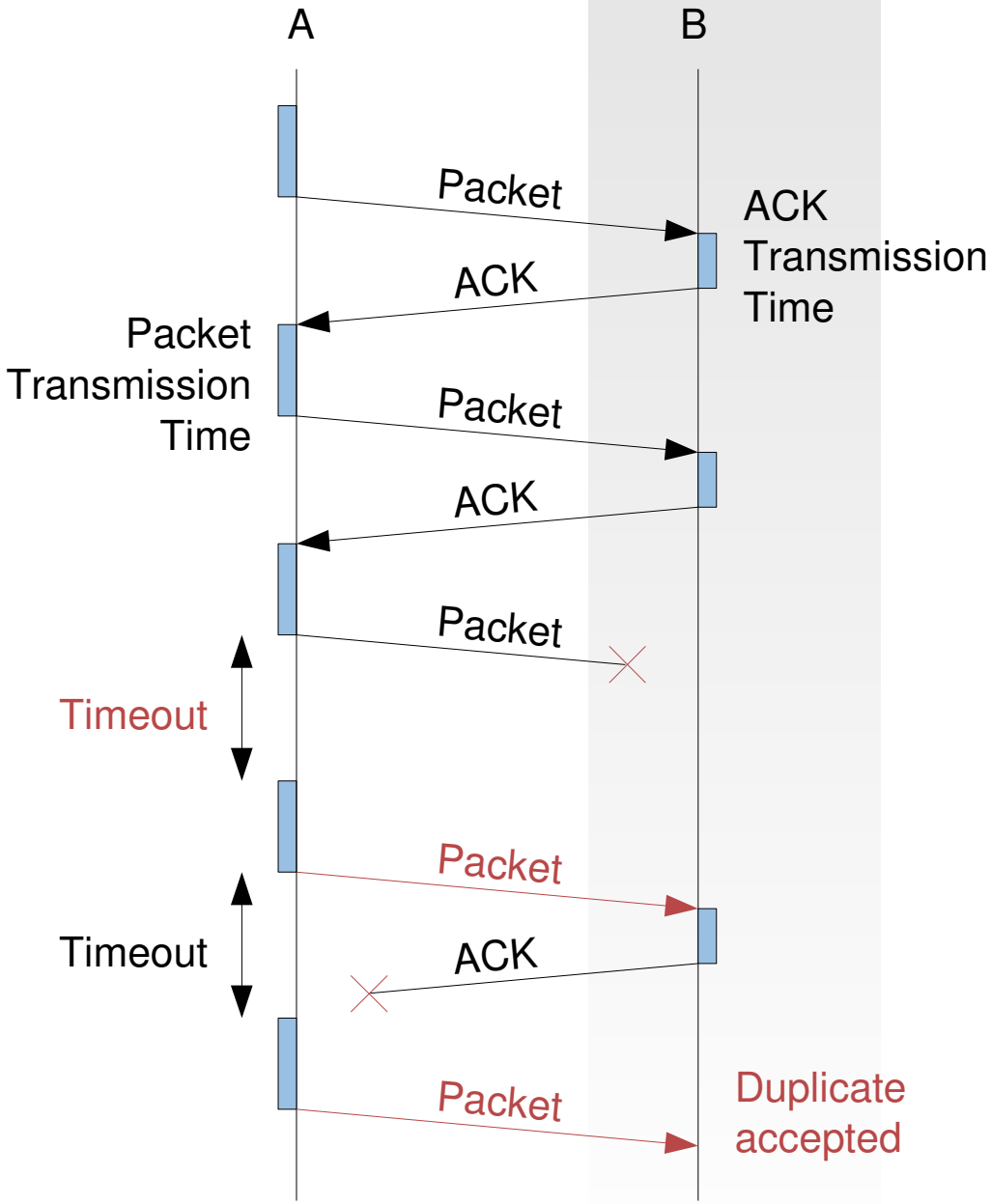
Stop and Wait (1)

- Principle
  - sending of a packet
  - waiting for acknowledge (ACK)
  - sending of the next packet
  - ...



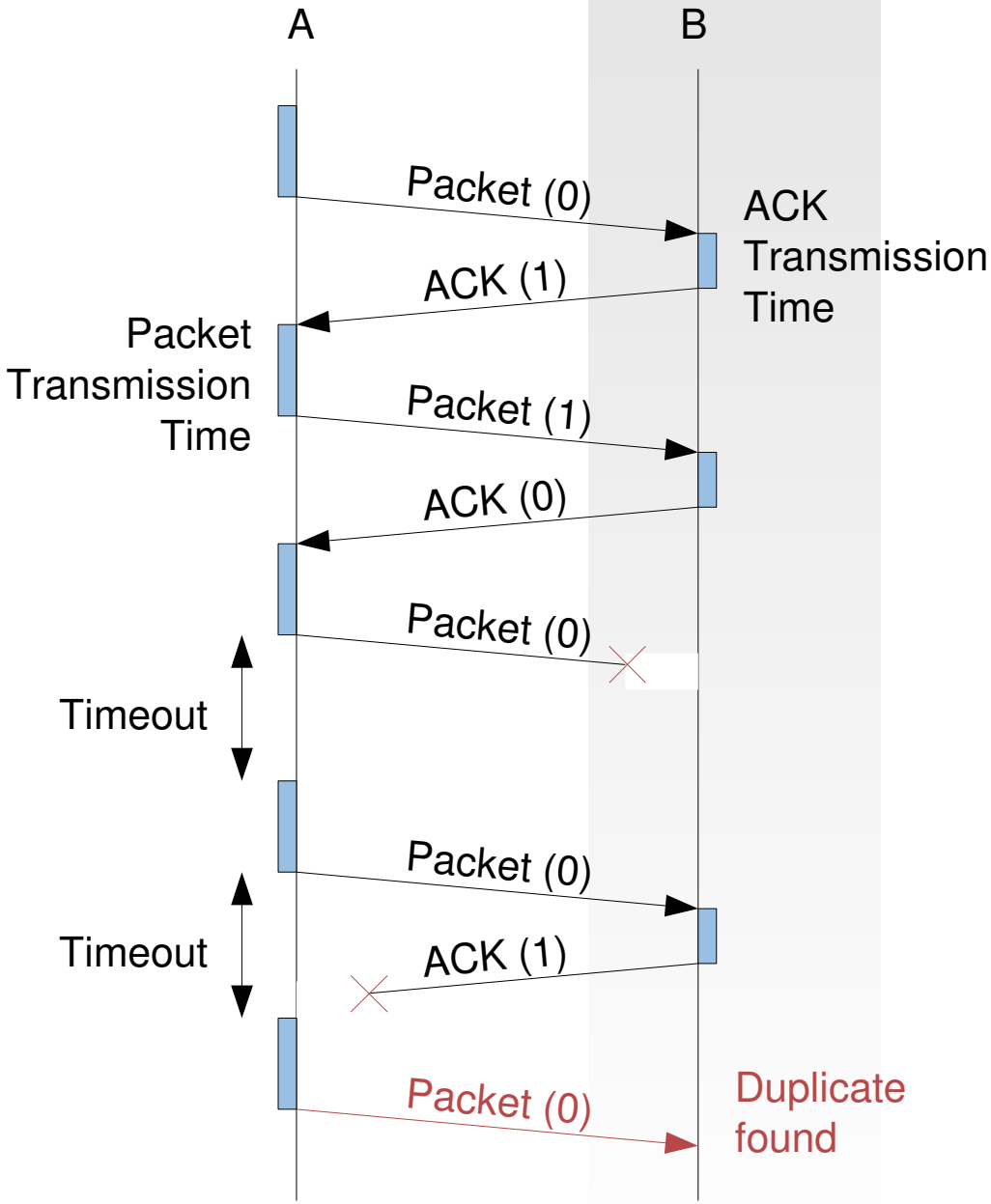
Stop and Wait (2)

- Principle
  - sending of a packet
  - waiting for acknowledge (ACK)
  - sending of the next packet
  - ...
- Possible errors
  - packet gets lost
  - ACK gets lost
- Problem
  - Packet Duplication
  - if ACK gets lost



Stop and Wait (3)

- Principle
  - sending of a packet
  - waiting for acknowledge (ACK)
  - sending of the next packet
  - ...
- Possible errors
  - packet gets lost
  - ACK gets lost
- Avoiding duplicated packets
  - every packet/ACK gets a label 0 oder 1
  - label in ACK indicates next expected packet
  - packets with wrong label are discarded

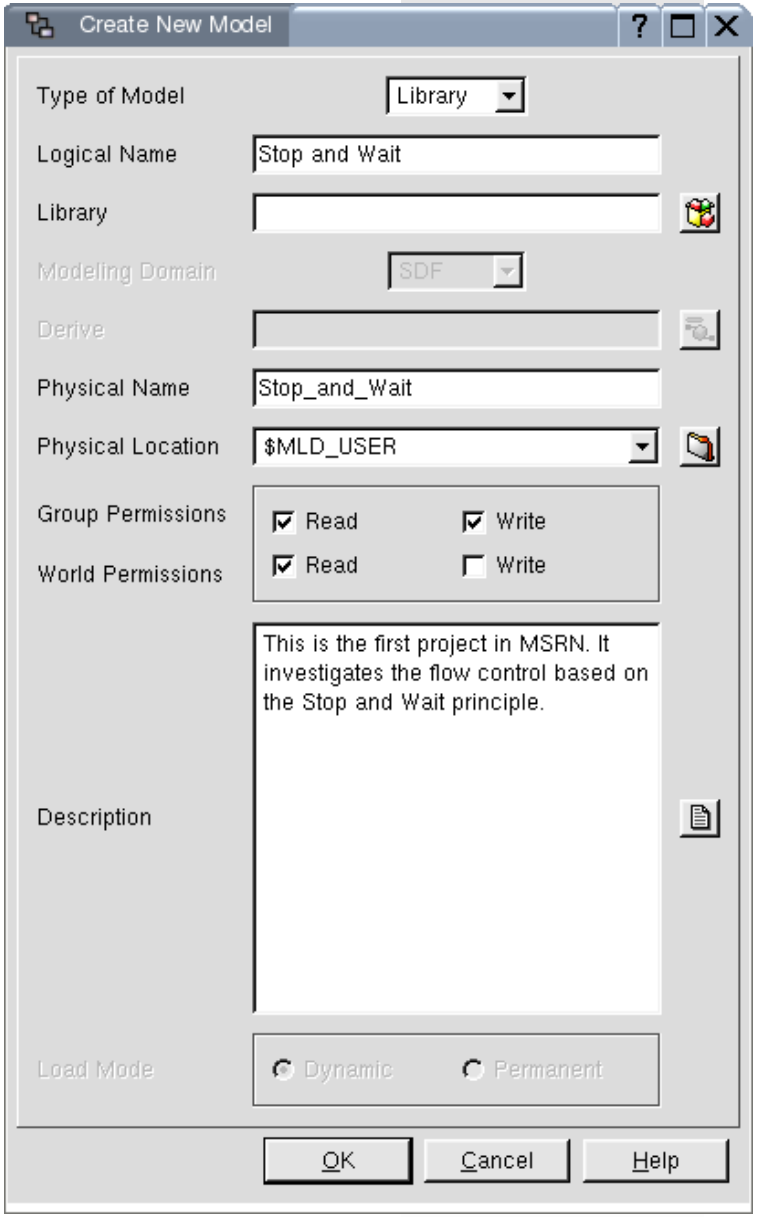


MLDesigner: Project

- Project-Elements
  - Library as the container
  - Data structures
  - Models
    - Module
    - Primitives
  - System models
  - Attributes for the Simulation
    - specific parameters
    - Probes

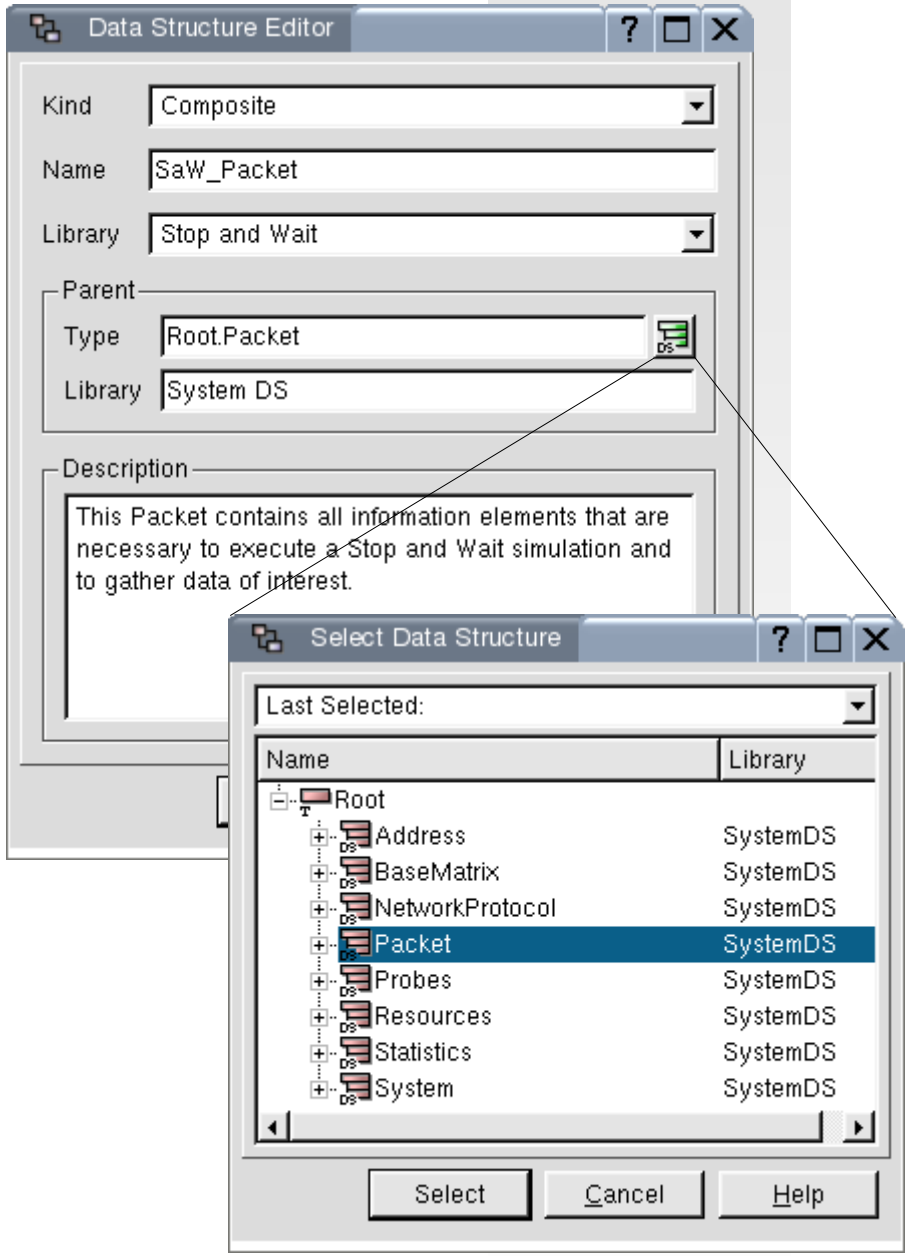
Stop and Wait: Creation of the project

- Create a Library
  - open New Model Dialog via
    - Tool Button: New Model
    - Menu: File – New
    - Tree View: context menu
  - Type of Model: Library
  - Logical Name: Stop and Wait
  - Description: ...
  - OK-Button



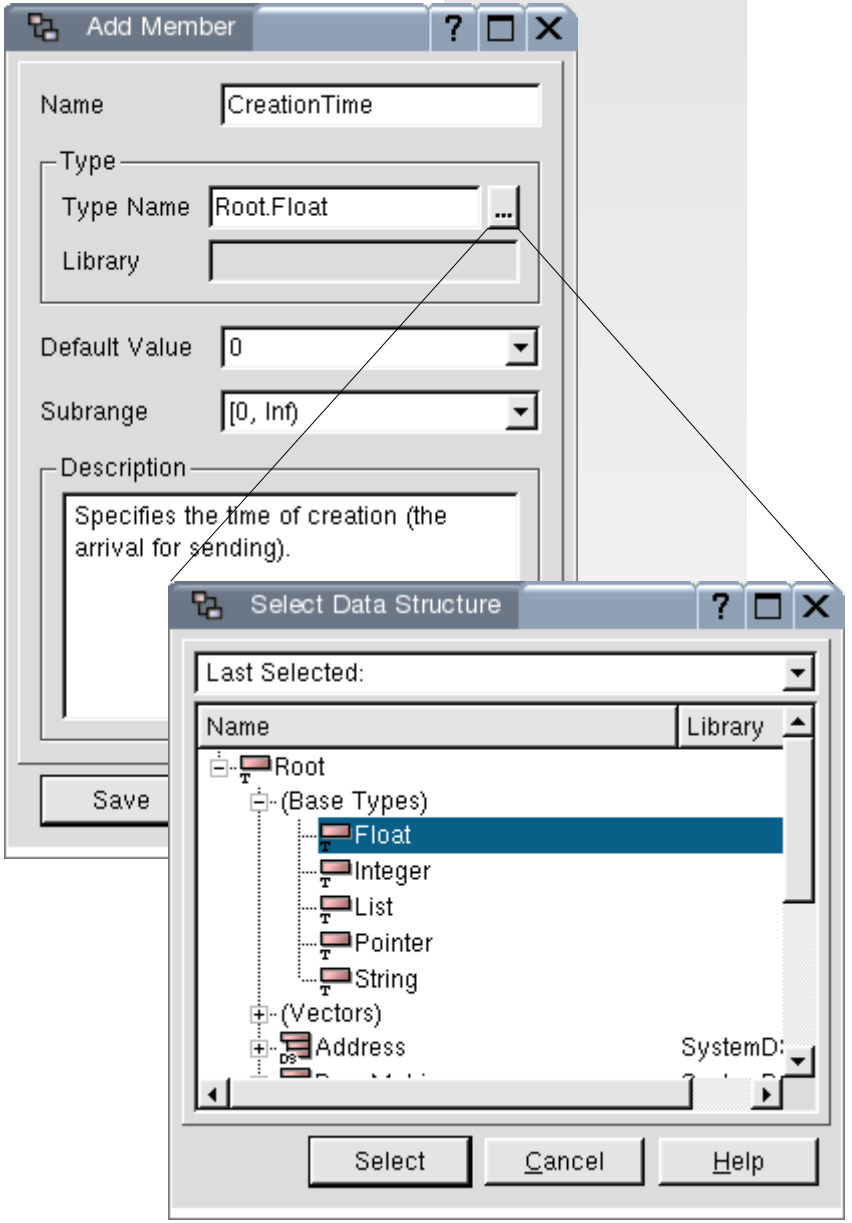
Stop and Wait: Data Structure for Packet (1)

- Open the Library Stop and Wait
  - File View: My Libraries – Stop and Wait
  - Double-Click
- Creating a Data Structure
  - Open New Data Structure Dialog 
    - Tool Button: New Data Structure
    - Context Menu in Data Structure Editor
  - Name: SaW_Packet
  - Parent Type: Root.Packet
  - Description: ...
  - Save-Button
- important: Library must be open
  - Model contained in Library is opened
  - Library itself is opened within a Model Editor



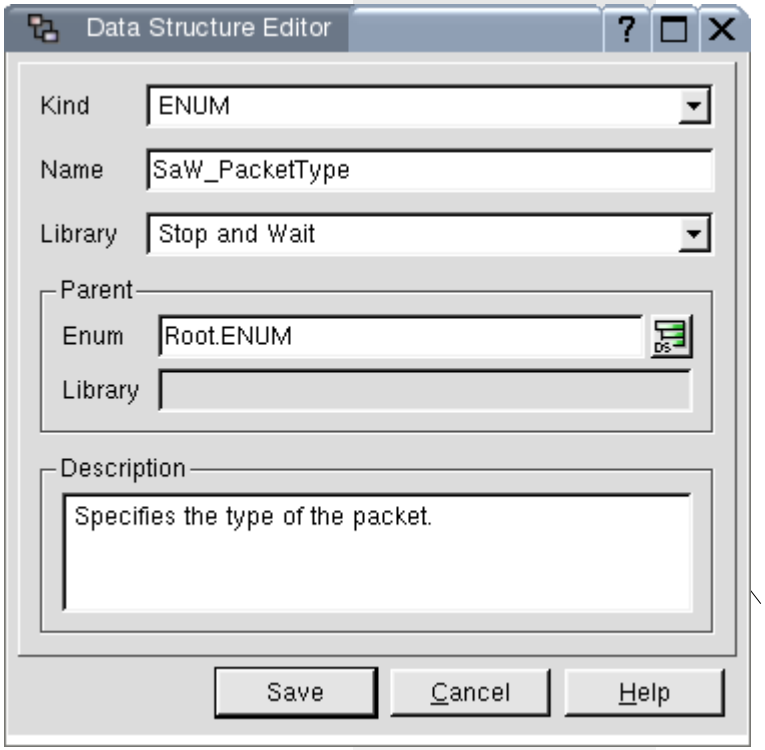
Stop and Wait: Data Structure for Packet (2)

- Creation of a Data Structure Member
  - Length of the packet
  - Time of packet creation
  - Time when packet was received
  - Label
  - Retransmission counter
- Creation of a Data Structure Member
  - Tool Button: New Member / Element 
  - Name: CreationTime
  - Type Name: Root.Float
  - Subrange: [0, Inf)
  - Description: ...
  - Save-Button



Stop and Wait: Data Structure for Packet (3)

- Creation einer Enumeration
  - open New Data Structure Dialog
    - Tool Button: New Data Structure 
    - Context Menu in Data Structure Editor
  - Kind: ENUM
  - Name: SaW_PacketType
  - Tarent Type: Root.ENUM
  - Description: ...
  - Save-Button



- important: Library must be open
  - Model contained in Library is opened
  - Library itself is opened within a Model Editor

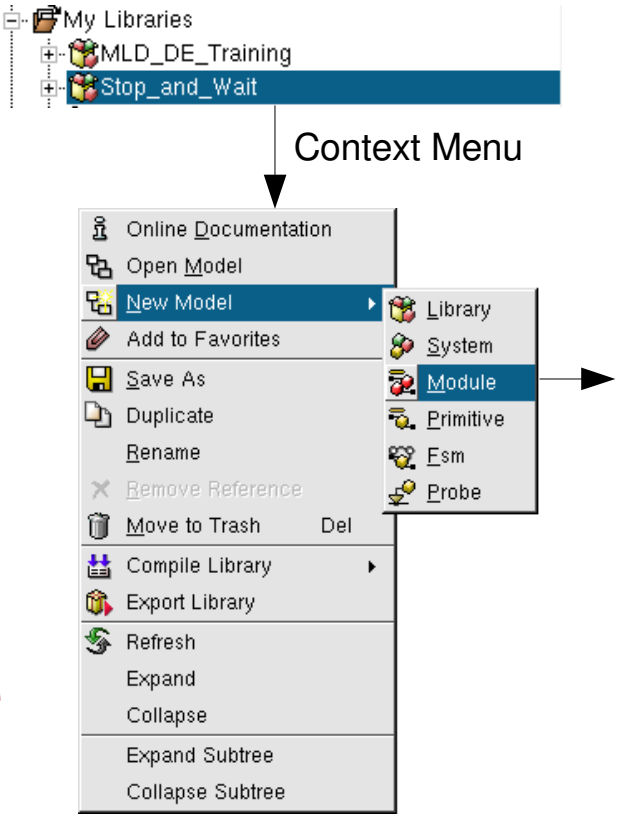
Stop and Wait: Data Structure for Packet (4)

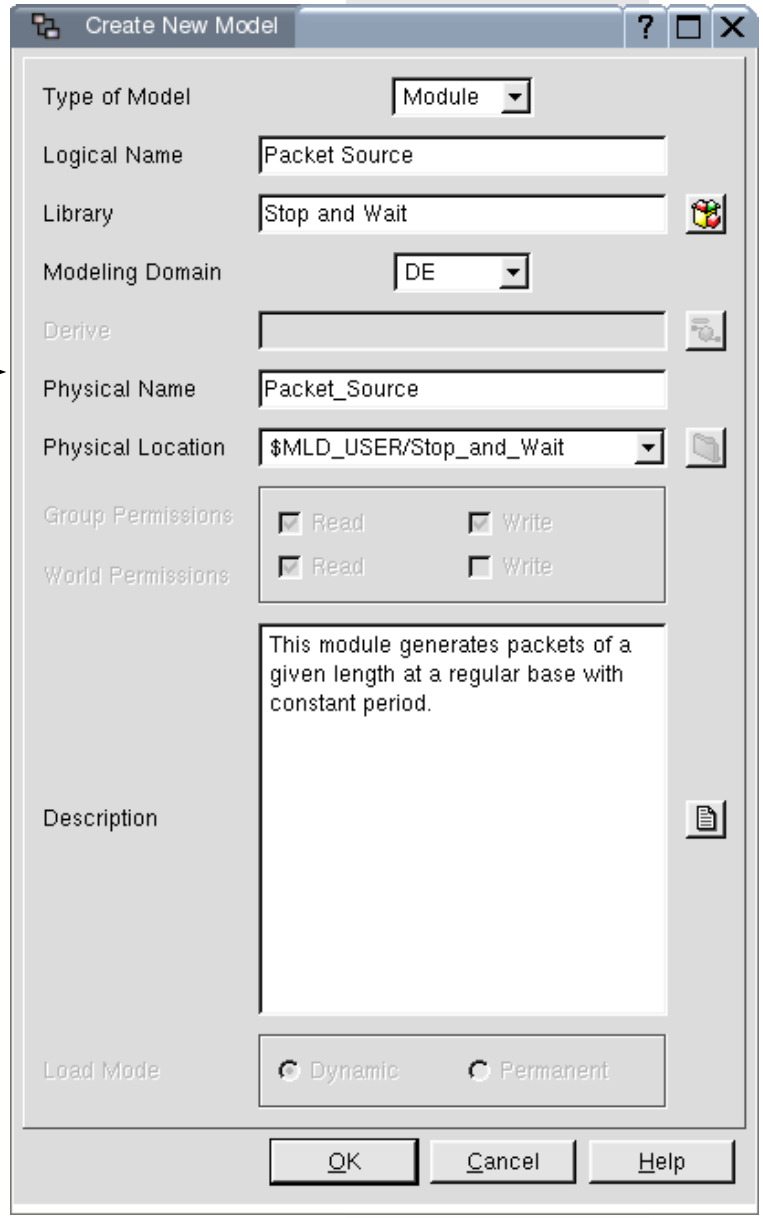
- Creation of all necessary Data Structure Members

Member	Type	Default	Subrange	Description
CreationTime	Root.Float	0	[0,Inf)	Specifies the time of creation (the arrival for sending) in seconds.
SentTime	Root.Float	0	[0,Inf)	Contains the time in seconds when the packet transmission was finished.
Type	Root.ENUM. SaW_PacketType	Data	-	Specifies the type of the packet.
Label	Root.Integer	0	[0,1]	Indicates which packet is sent by the transmitter and expected by the receiver, respectively.
Transmissions	Root.Integer	0	[0,Inf)	Contains the number of transmission attempts.
Length	Root.Integer	0	[0,Inf)	Contains the packet length in Byte
PacketNumber	Root.Integer	1	[1,Inf)	Sequence number used for statistical purpose only.

Stop and Wait: Module (1)

- Neccessary Modules
  - Packet Source
  - Transmitter
  - Channel
  - Receiver
  - Packet Sink
- Creation of a Modules
  - Type of Model: Module
  - Library: Stop and Wait
  - Modeling Domain: DE
  - Logical Name: Packet Source
  - Description: ...
  - OK-Button





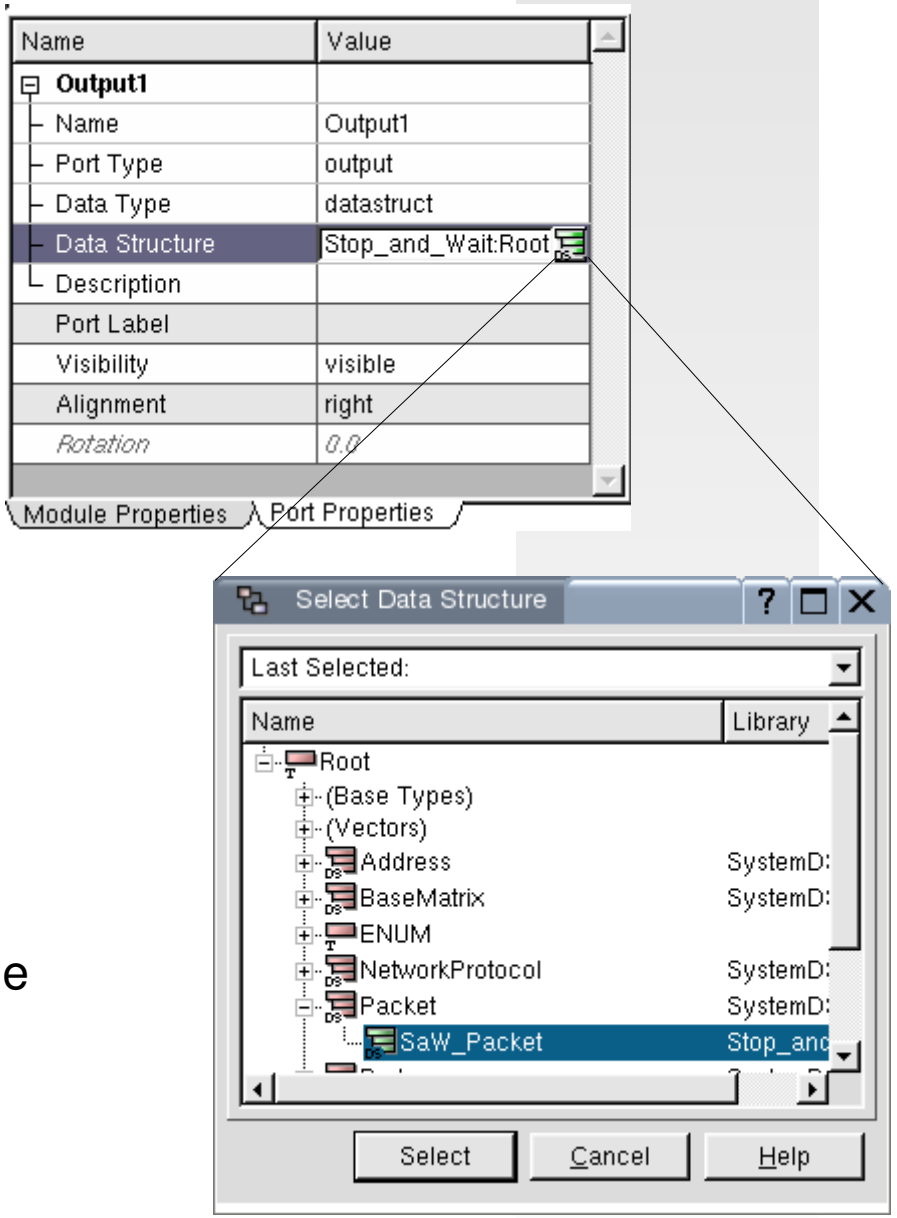
Stop and Wait: Module (2)

- Creation of all necessary moduley

Module	Description
Packet Source	This module generates packets of a given length at a regular base with constant period.
Transmitter	The module models the transmission of an arrived packet
Channel	The module injects bit errors and simulates an errorneous (lossy) channel. The bit error rate is determined by a given distribution and its parameters. Packets with single bit errors are lost.
Receiver	This module receipts the packets and sends acknowledgements. It uses the label field in acknolegdments to indicated which packet is expected next.
Packet Sink	This module uses the packet received by the receiver module to gain the data of interest.

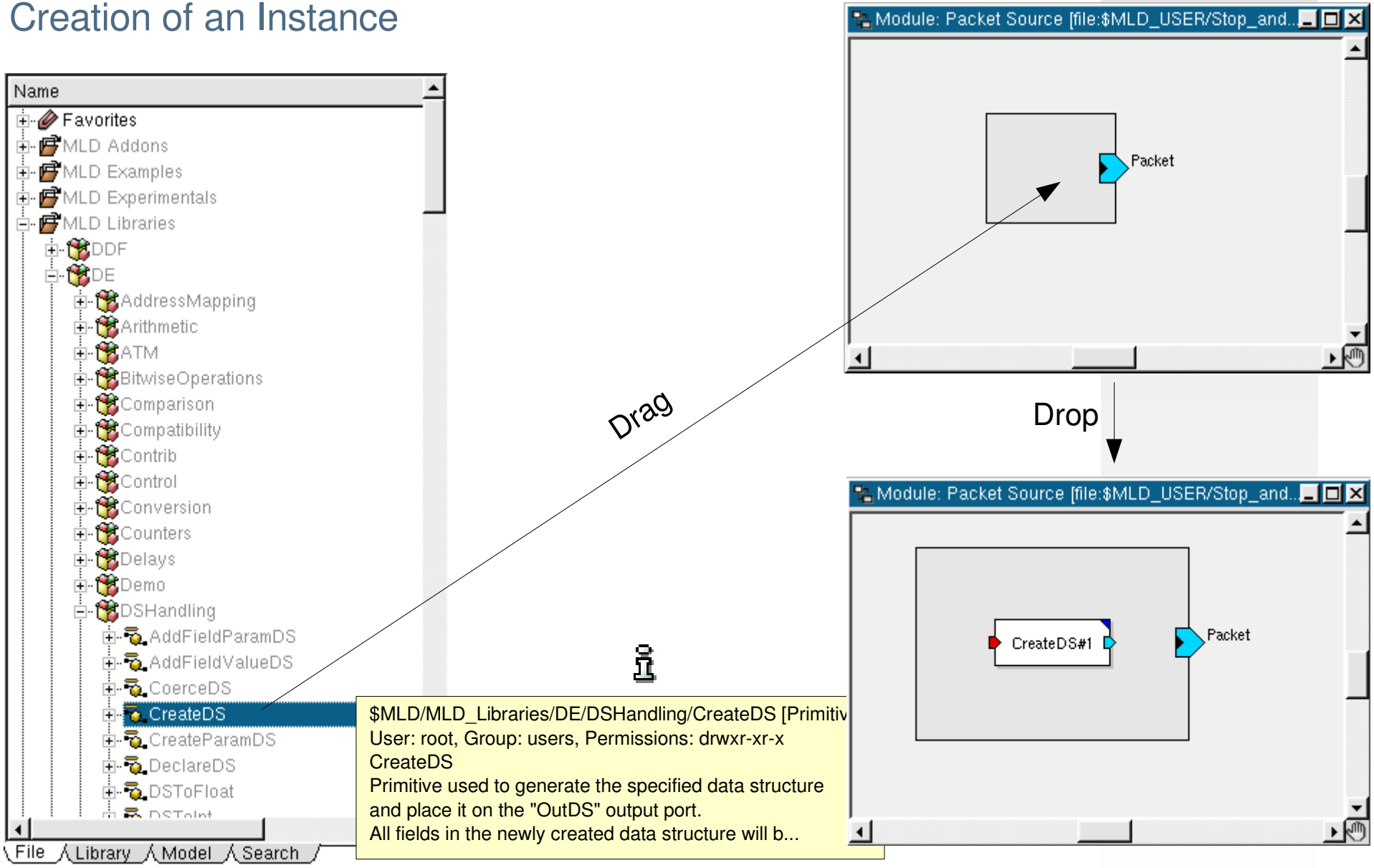
Stop and Wait: Packet Source (1)

- Creation of an Output Ports
  - Tool Button: Add Output Port 
    - changes to mode Port Creation
    - cancel with ESC or right-click
  - Context Menu: Add Output Port
  - select Port
  - Data Type: datastruct
  - Data Structure: Root.Packet.SAW_Packet
  - Description: ...
  - Name: Packet
    - inline in Model Editor
    - optional separate Instance Label possible



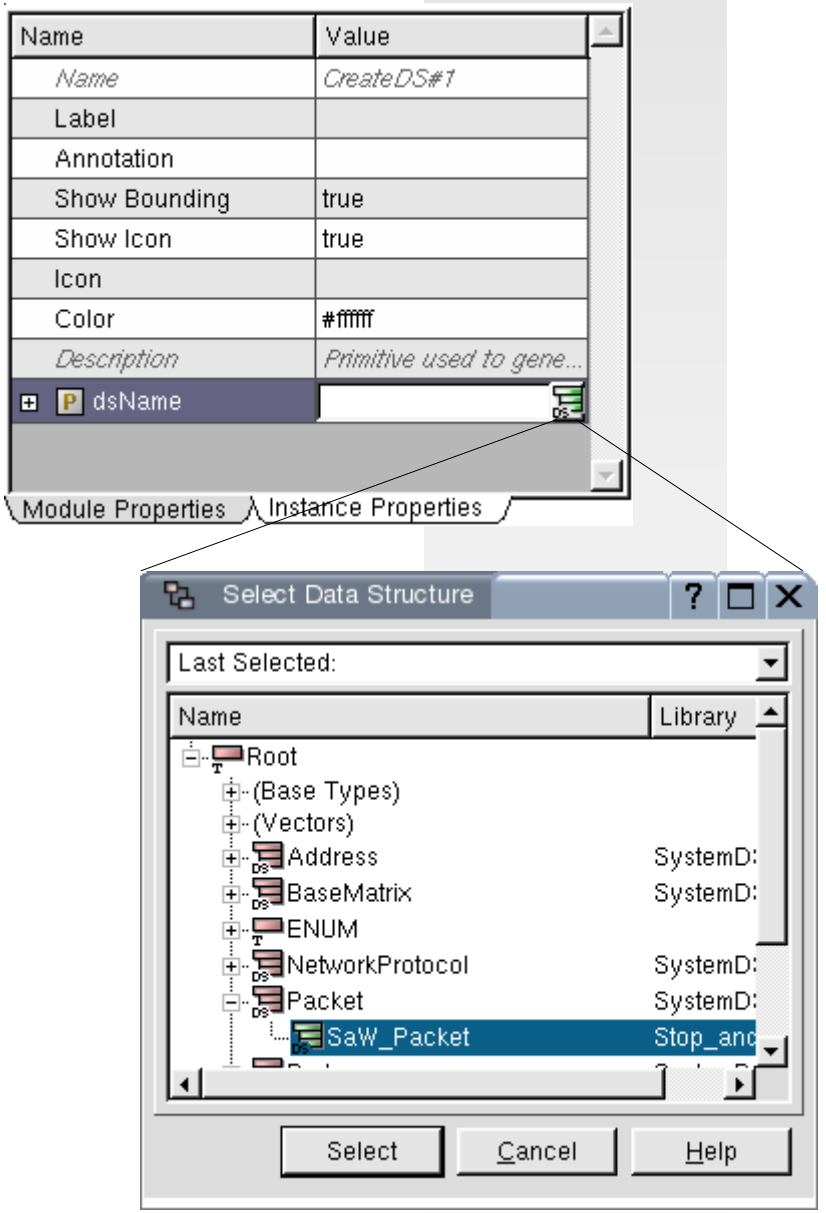
Stop and Wait: Packet Source (2)

- Creation of an Instance



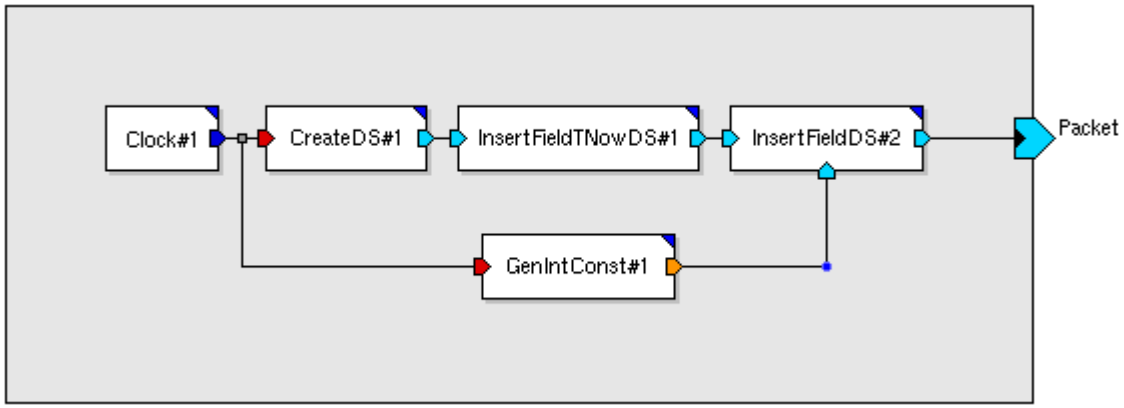
Stop and Wait: Packet Source (3)

- Setting Parameters
  - CreateDS has a parameter: dsName
  - specifies the created data structure
  - Usage of context sensitive dialog Select Data Structure
- Properties of Parameters
  - can be expanded
  - Tool Tip shows short summary
    - name and type
    - current value
    - description



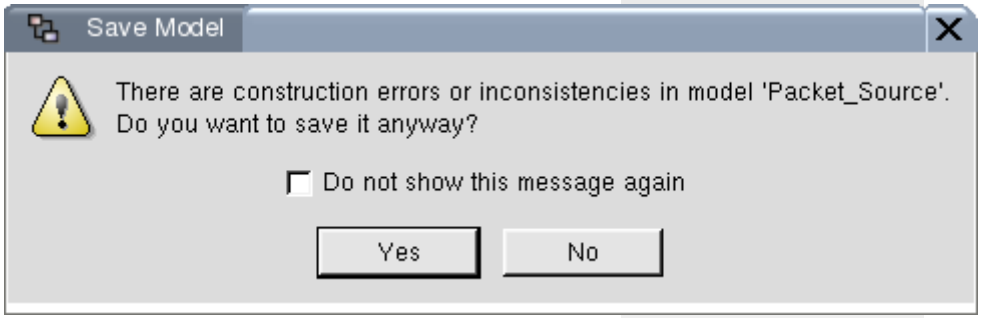
Stop and Wait: Packet Source (4)

- Creation of instances
  - Clock (Sources): periodical generation of events
  - CreateDS (DSHandling) : creation of a packet
  - InsertFieldTNowDS (DSHandling): setting CreationTime
  - GenIntConst (NumberGenerators): setting CreationTime
  - InsertFieldDS (DSHandling): setting packet length
- Create relations
  - start: double click on Port
  - finish: click on Port
  - free intermediate points
- Hint:
  - all Ports have to be allocated
  - Ports can be terminated

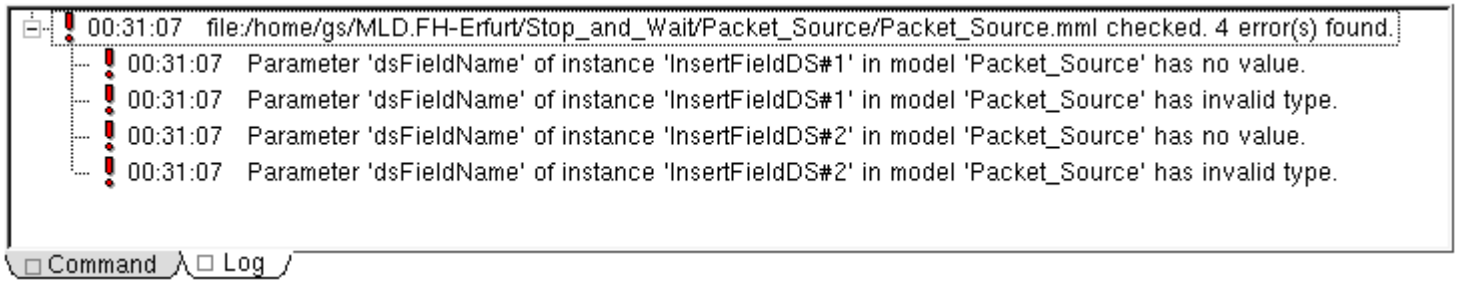


Stop and Wait: Packet Source (5)

- Save Model
  - Tool Button: Save oder Save All 
  - Menu: File – Save / Save All
  - Key Shortcut: Ctrl-S

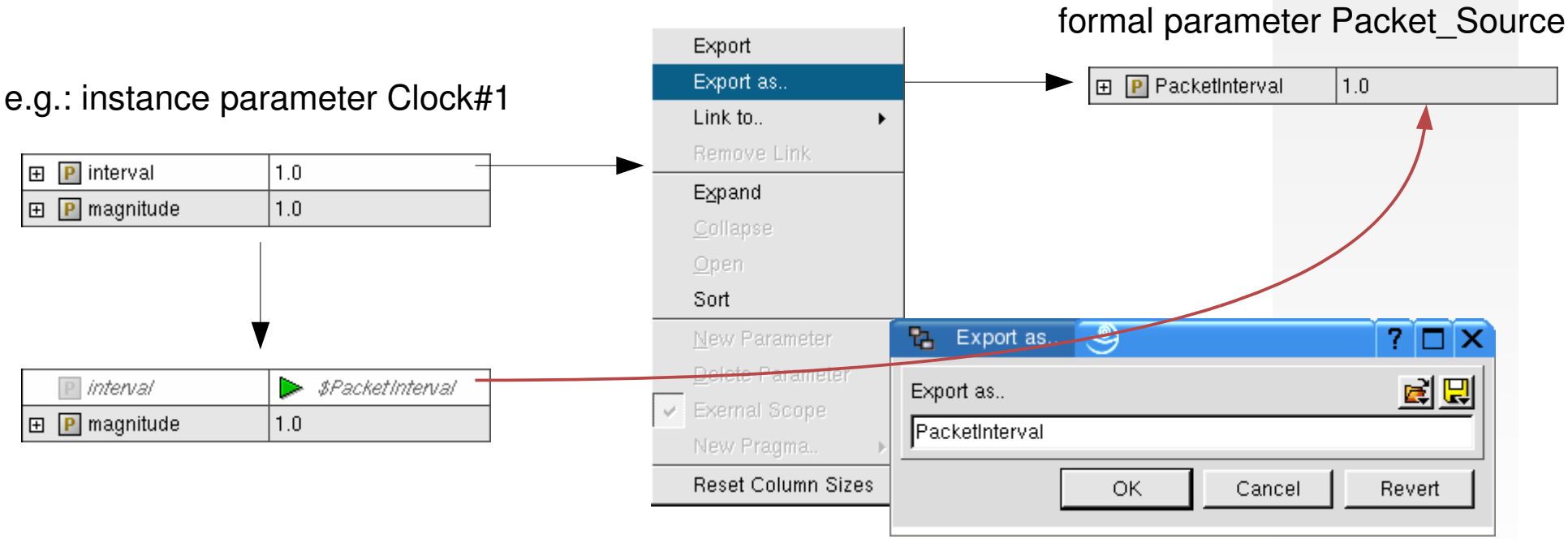


- Model Check
  - automatically performed during model save
  - Tool Button: CheckDesign



Stop and Wait: Packet Source (6)

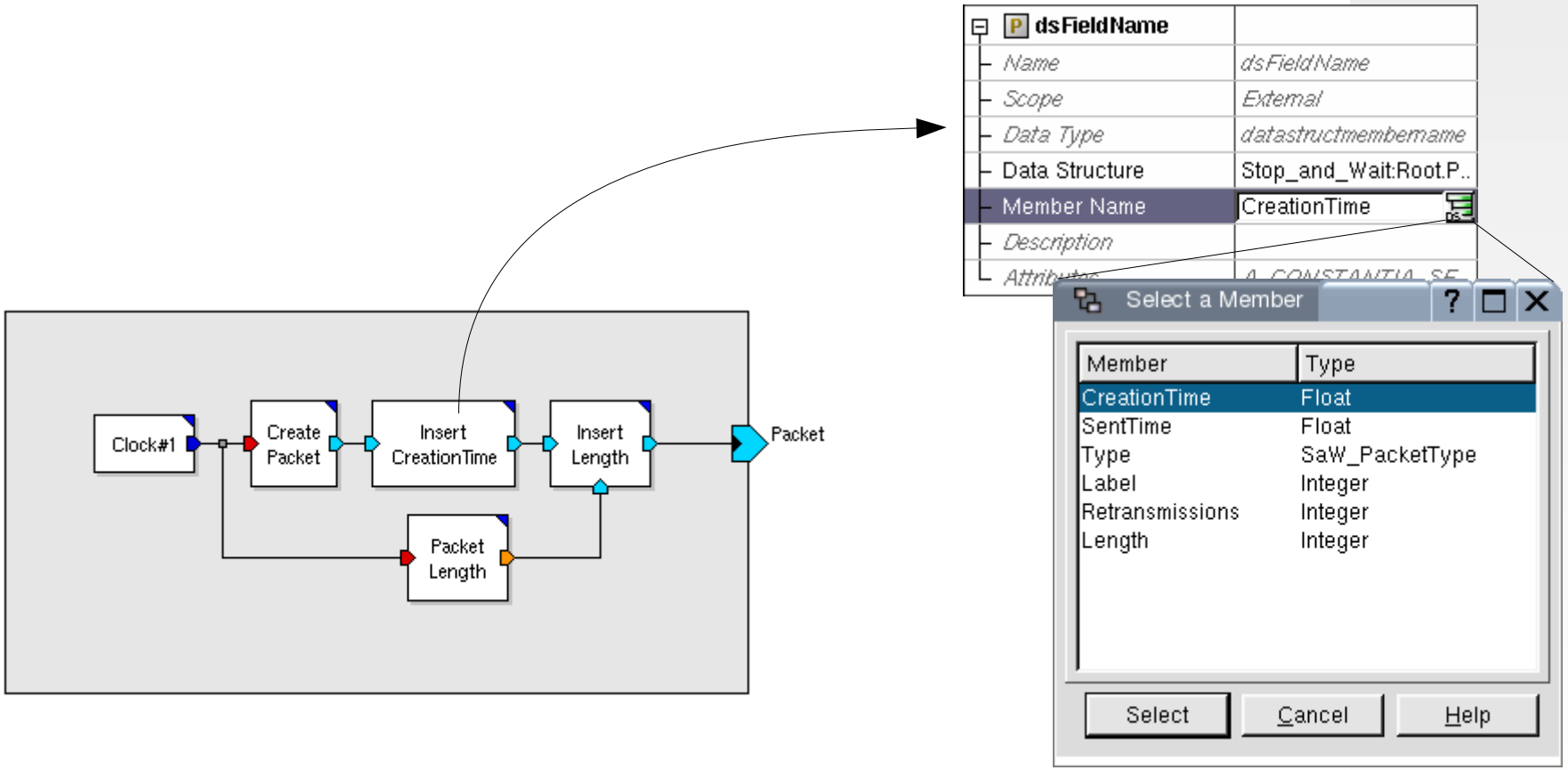
- linking Parameter
  - instance parameters are linked to formal parameters of the model
  - value of the instance parameter results from value of the parameter of the model
- exporting Parameter
  - instance parameter is cloned to the model and automatically linked to its clone
  - name of the clone is configurable during export



Stop and Wait: Packet Source (7)

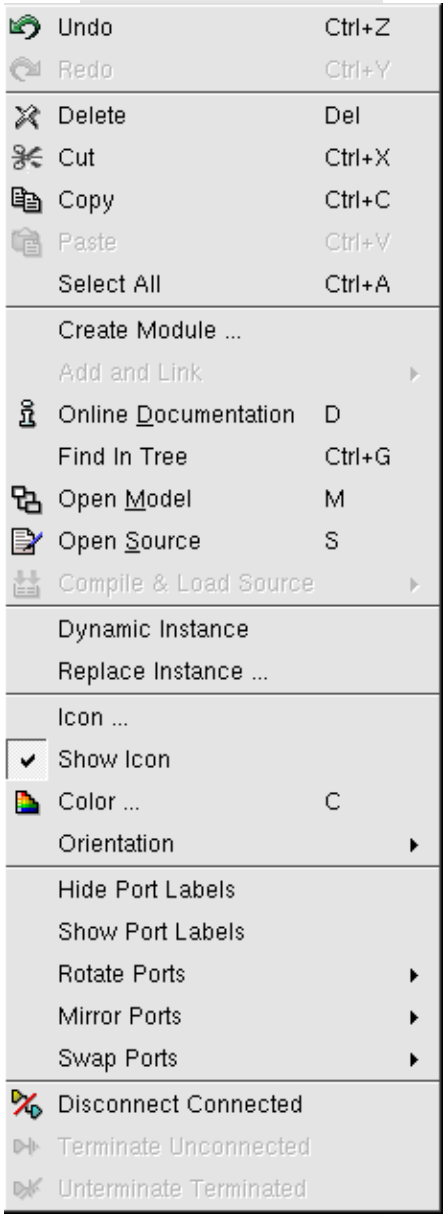
Finishing the Module

- setting of instance labels (more intuitive labels)
- export of the parameter Value of GenIntConst#1 as parameter PacketLength
- setting of parameters dsFieldName of InsertFieldDS#1 and InsertFieldDS#2



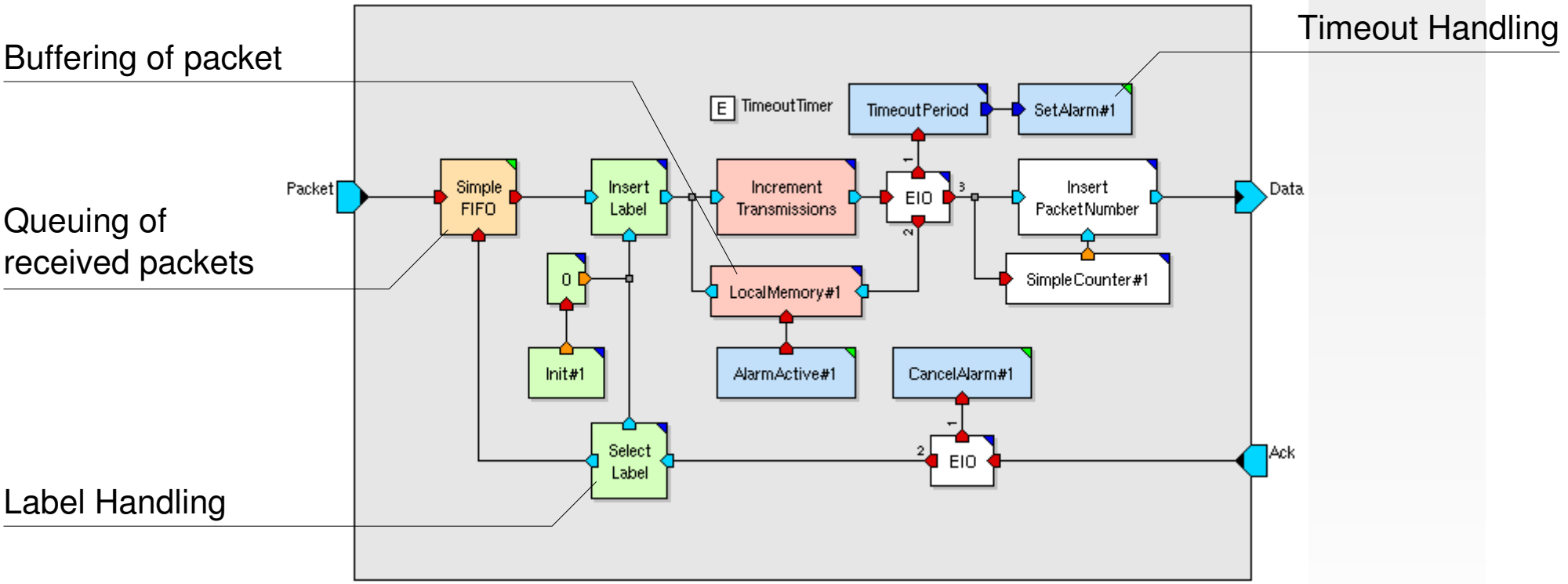
MLDesigner: Useful Editor Functions

- Instances
  - Resizing of the instance block
  - double-click to open model
  - definition of color or icon
  - replacing instances via Drag & Drop
  - rotate and swap Ports
  - move Ports
  - highlighting of instances with linked model elements
  - Create Module
- general functions in Context-Menu
  - Find in Tree for models and instances
  - Online Documentation
- Verbindungen
  - creation and deletion of intermediate points at any time



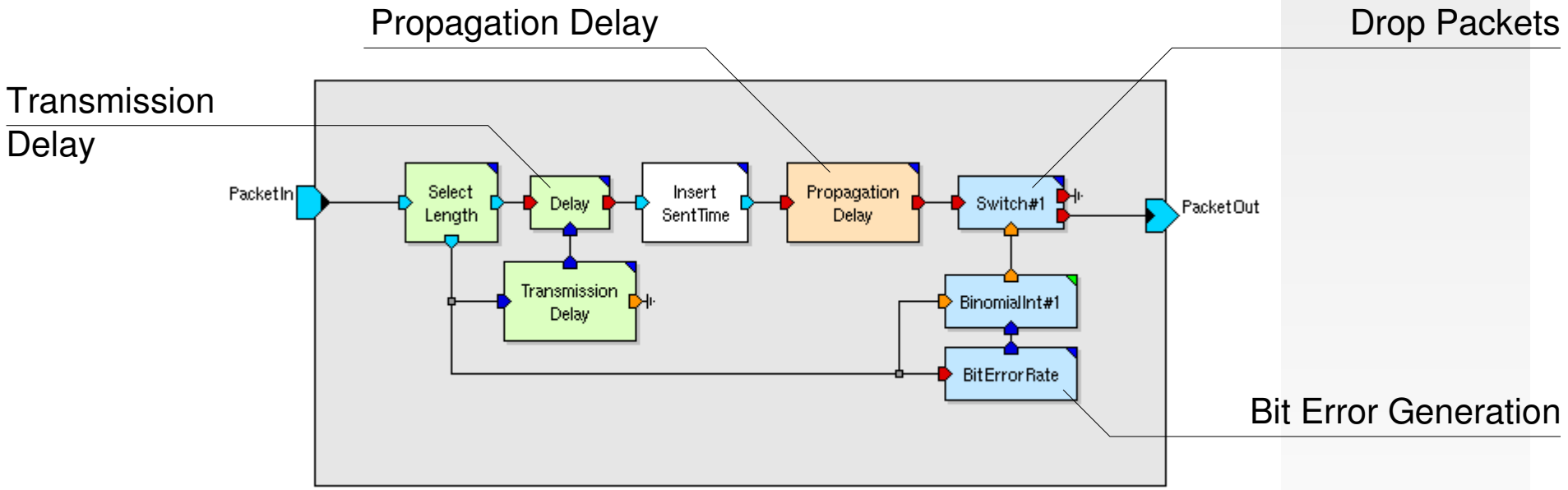
Stop and Wait: Transmitter

- Formal parameters
  - MaximumQueueSize [int] = 5
  - TimeoutValue [float] = 1.0



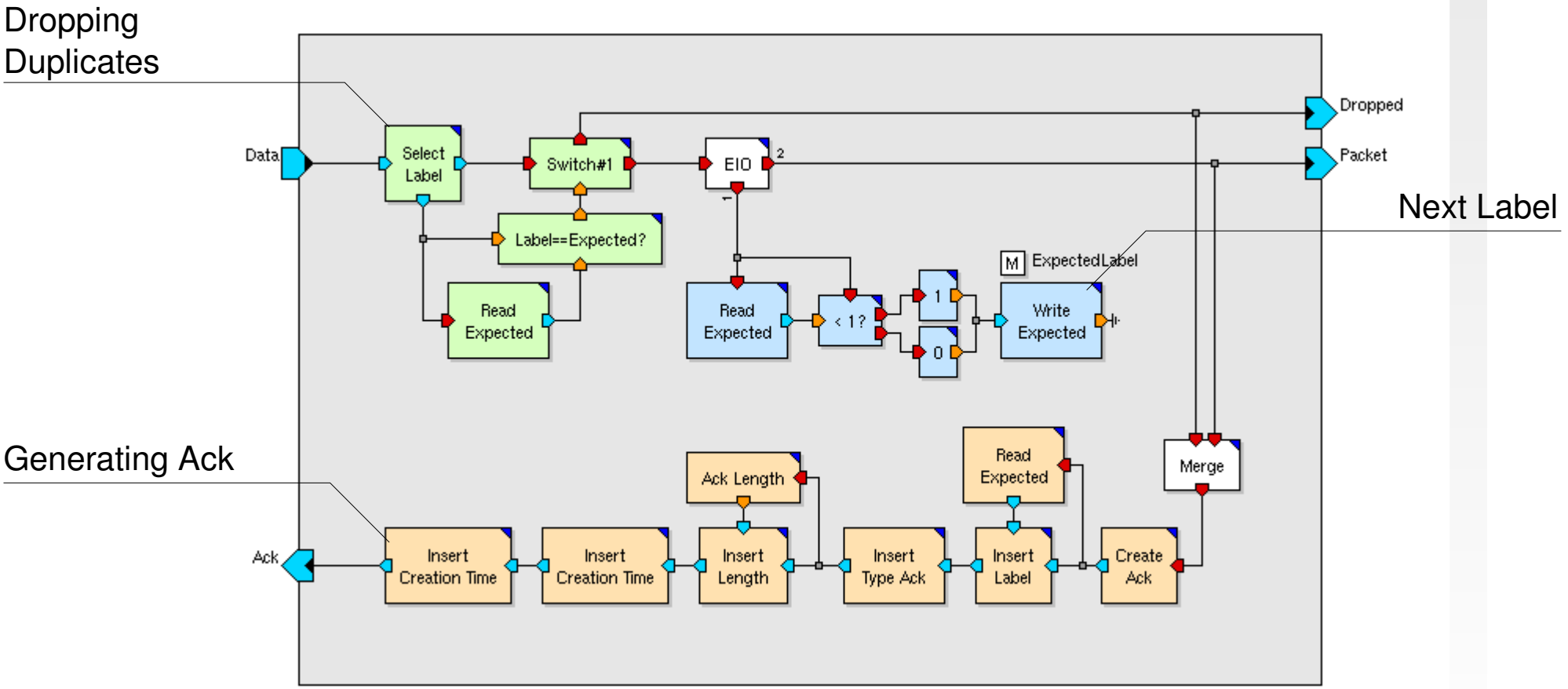
Stop and Wait: Channel

- Formal parameters
  - BitErrorRate [float] = 1e-6
  - Propagation Delay [float] = 0
  - ChannelSpeed [int] = 10000



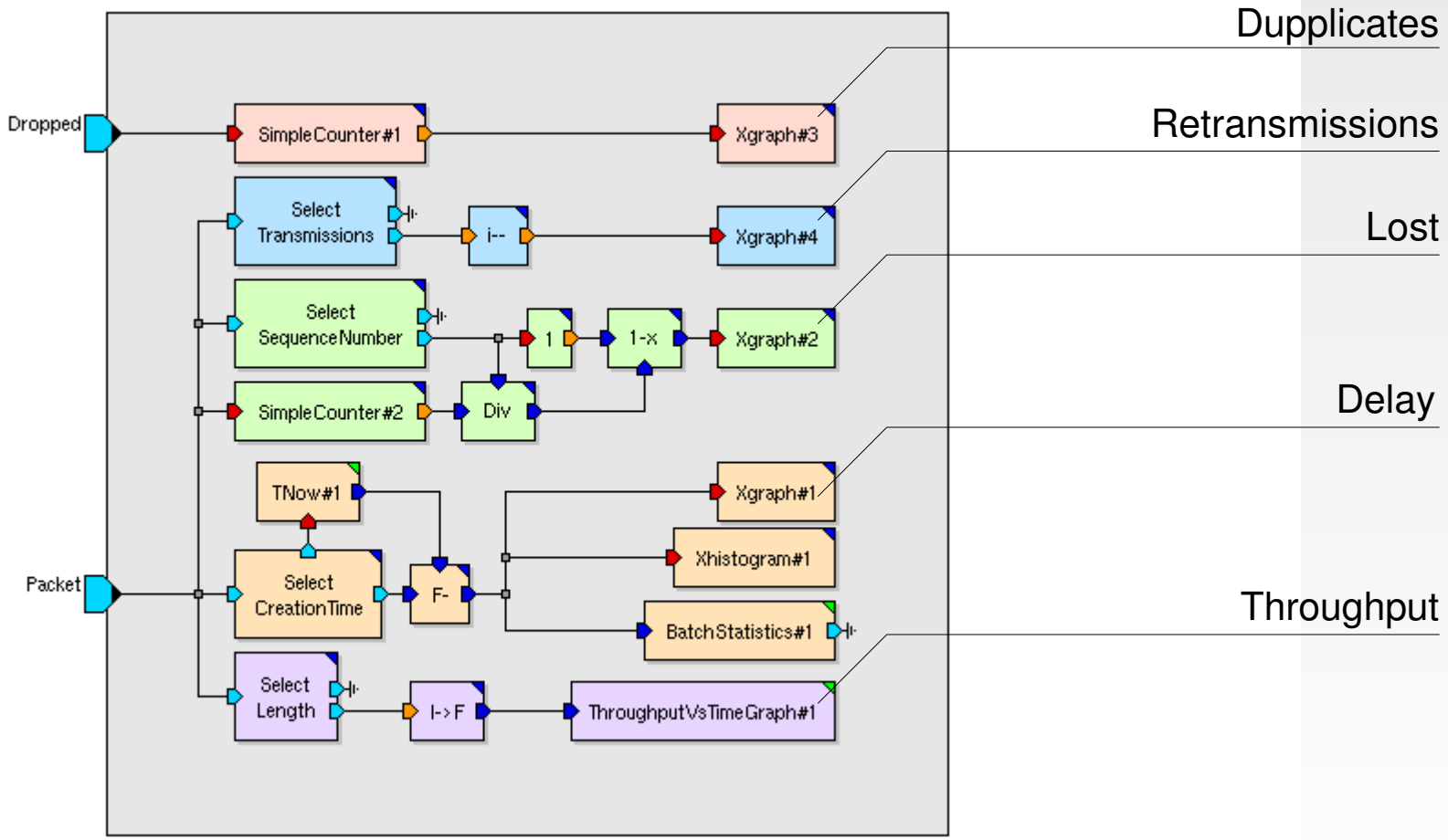
Stop and Wait: Receiver

- Formal parameters
  - AckLength [int] = 2



Stop and Wait: Packet Sink

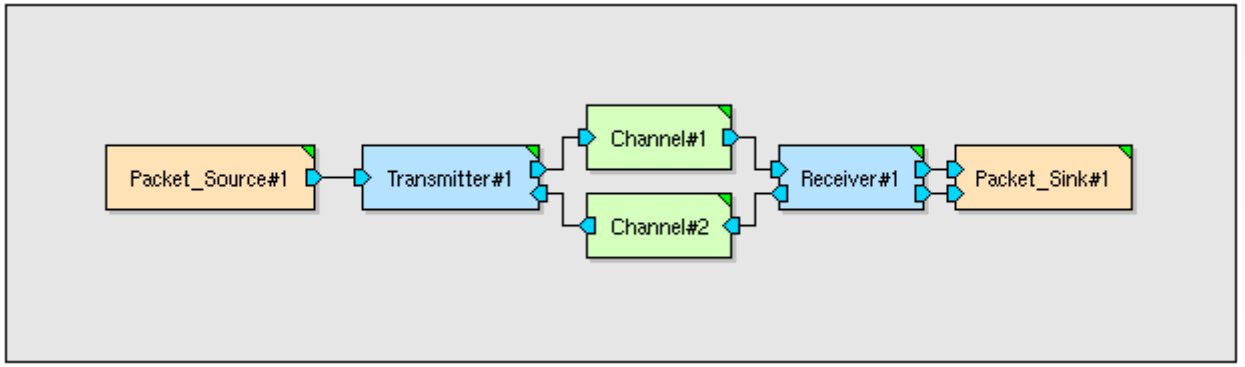
- Formal parameters
  - ChannelSpeed [int] = 10000



Stop and Wait: System

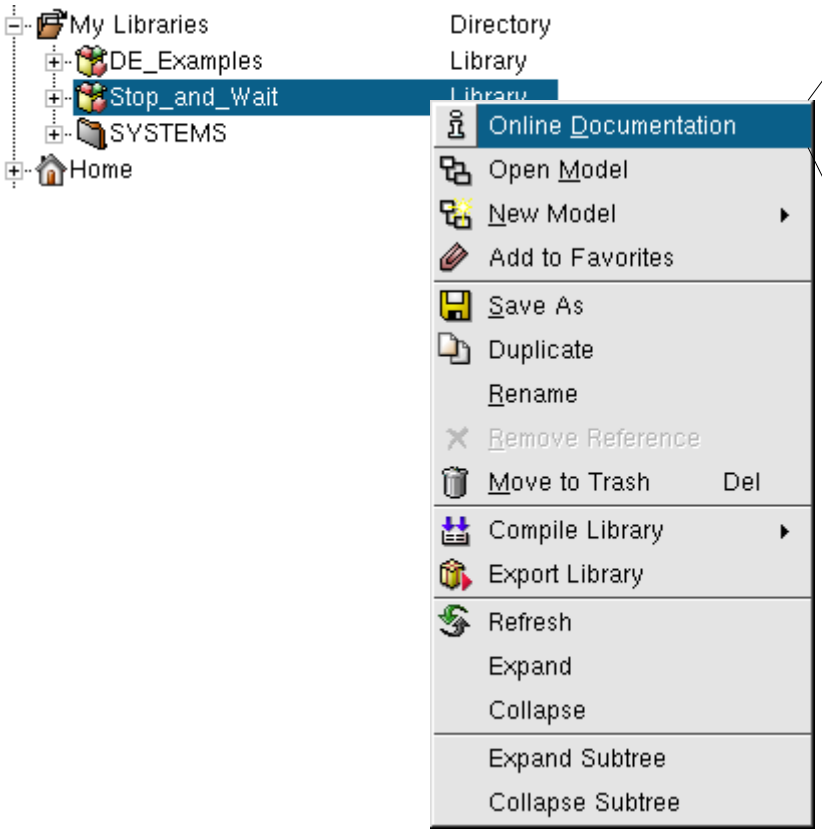
System parameters

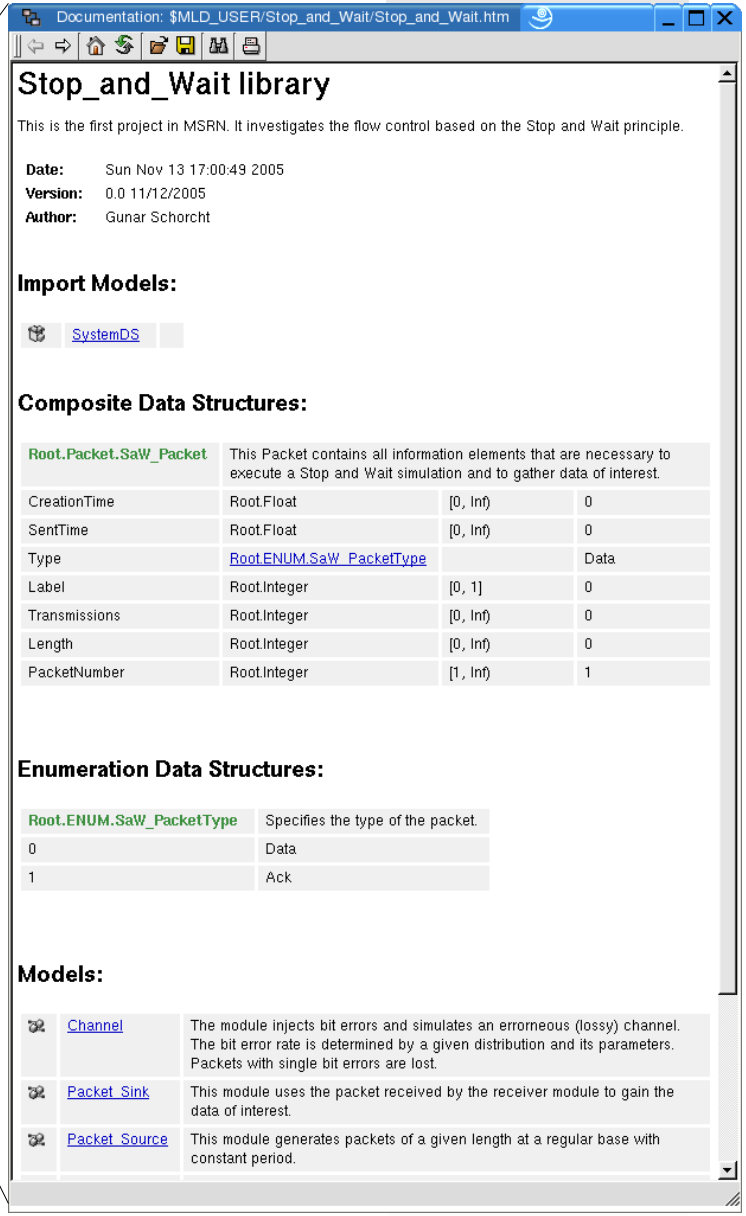
Parameter	Typ	Wert
PacketInterval	float	$PacketLength*8/$ChannelSpeed*0.6
PacketLength	int	512
MaximumQueueSize	int	500
TimeoutValue	float	$PacketLength*8/$ChannelSpeed*10.0
ChannelBitErrorRate	float	1e-4
PropagationDelay	float	0
ChannelSpeed	int	56000
AckLength	int	2



Stop and Wait: Online Documentation

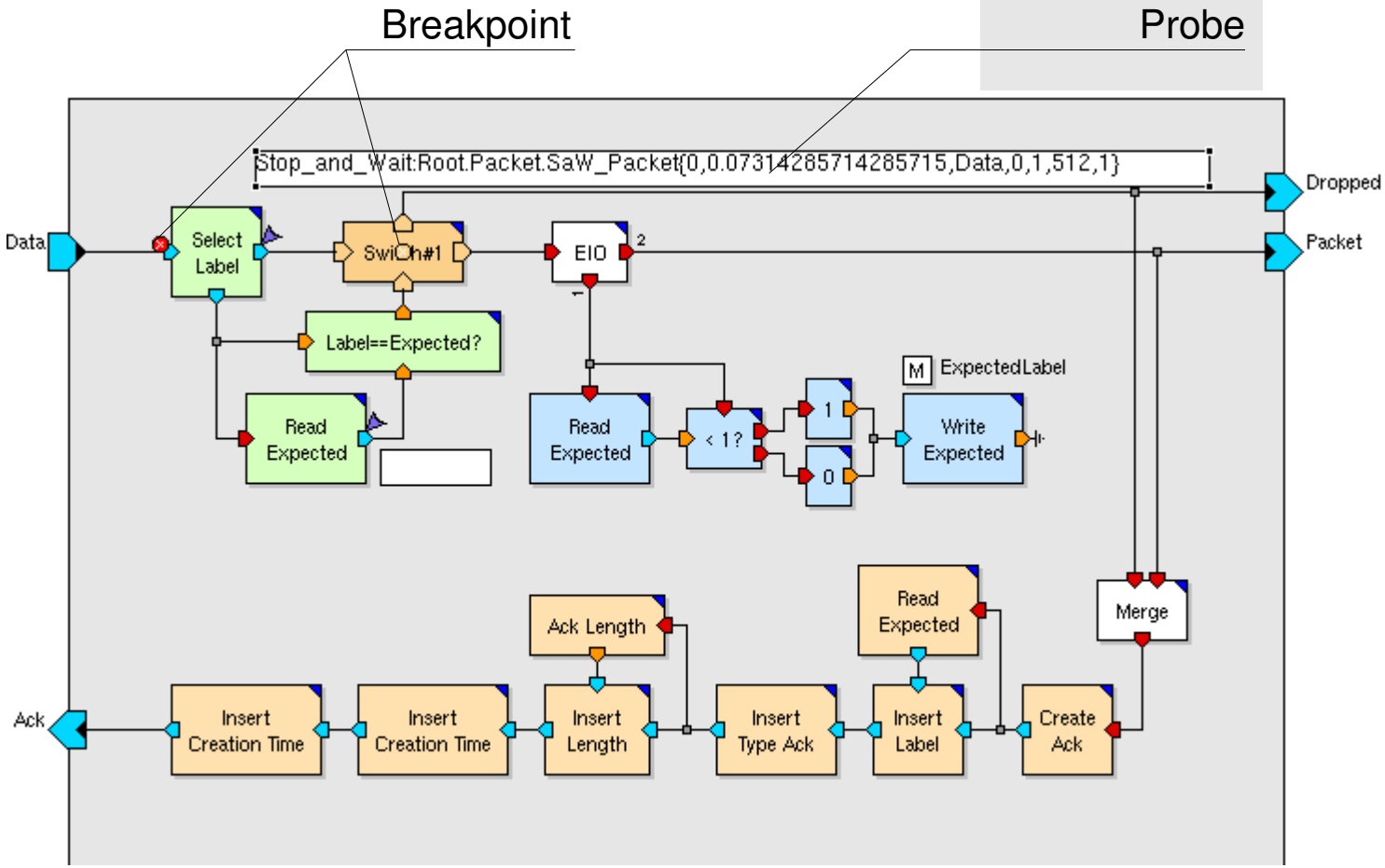
- Hypertext-Online-Documentation
  - automatically generated
  - completely linked





Stop and Wait: Simulation (1)

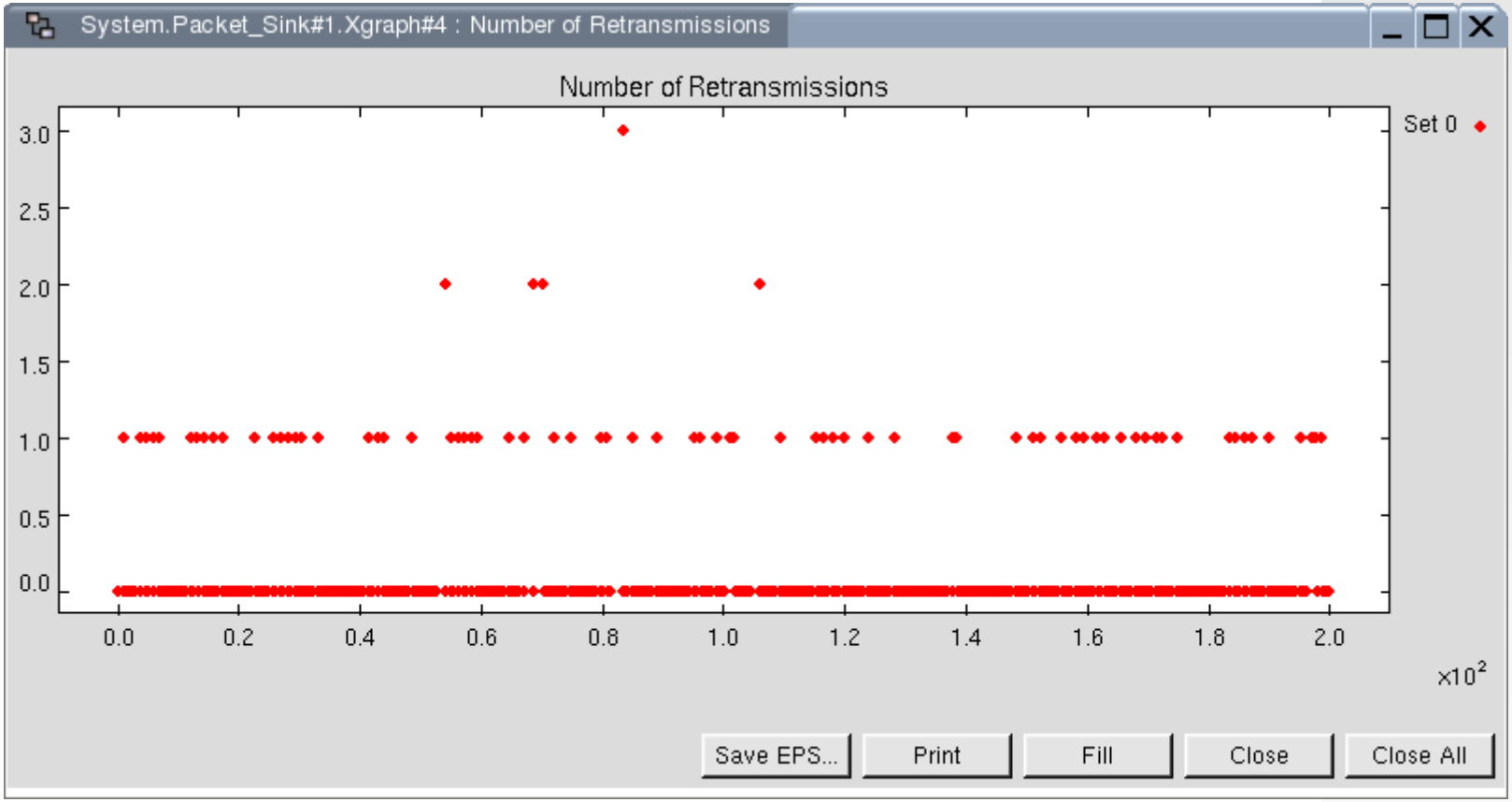
- Simulation Mode
  - interactiv
  - Breakpoints
  - Animation
  - Probes
  - external C++/PTcl
  - distributed
- Probes
  - data mining
  - internal display



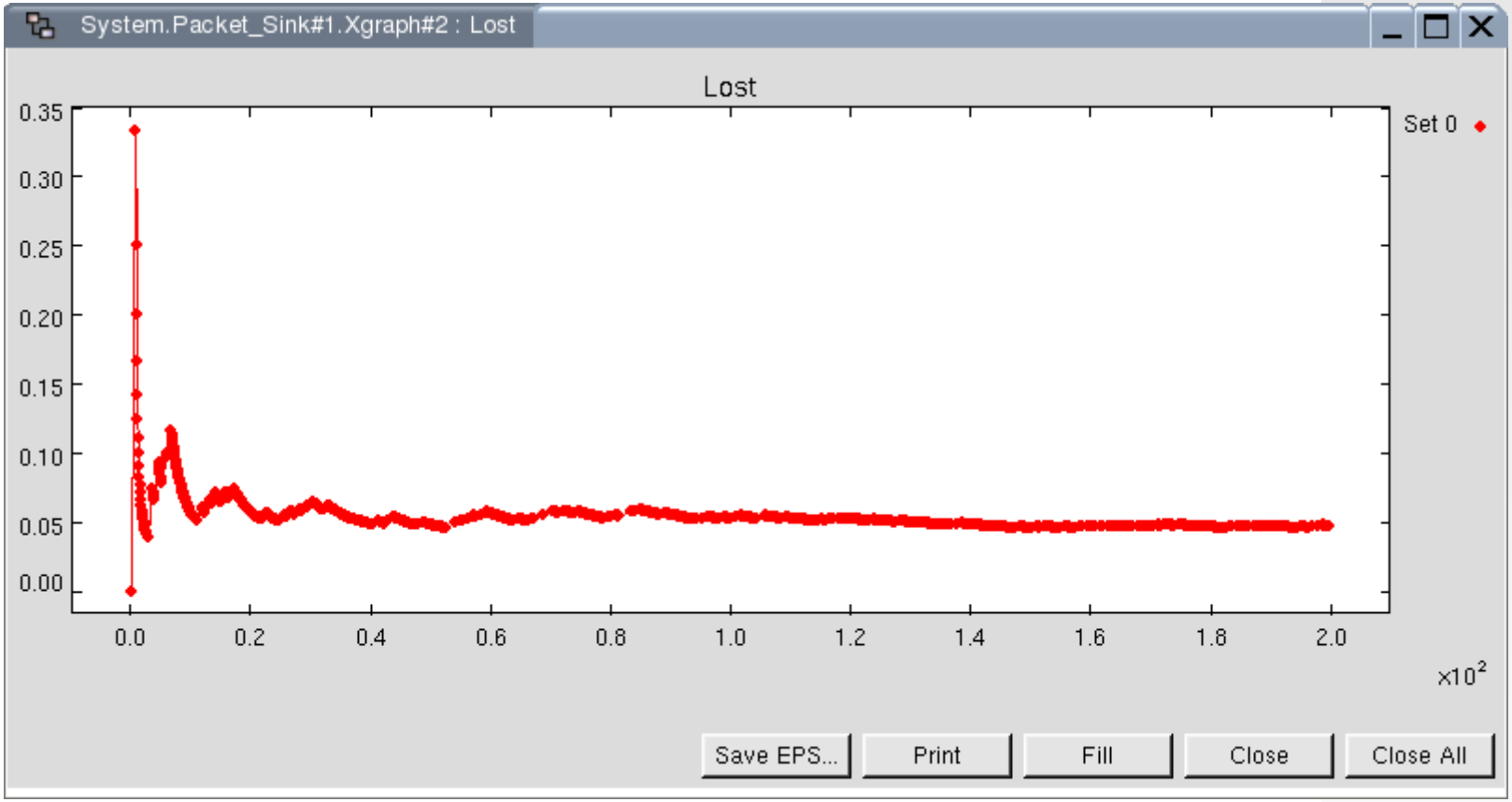
Iteration	Time	Entity	Name	Action	Value
1	0.926476190476191	Instance	System.Channel#1.Switch#1	will now execute	
1	0.926476190476191	Port	System.Channel#1.Switch#1.falseOut	send	Stop_and_Wait:Root.Packet.SaW_Packet{0.121904761904
1	0.926476190476191	Port	System.Receiver#1.SelectFieldDS#1.InDS	received	Stop_and_Wait:Root.Packet.SaW_Packet{0.121904761904
1	0.926476190476191	Instance	System.Receiver#1.SelectFieldDS#1	will now execute	
1	0.926476190476191	Port	System.Receiver#1.SelectFieldDS#1.Field	send	Root.Integer{1}
1	0.926476190476191	Port	System.Receiver#1.SelectFieldDS#1.OutDS	send	Stop_and_Wait:Root.Packet.SaW_Packet{0.121904761904
1	0.926476190476191	Port	System.Receiver#1.Switch#1.input	received	Stop_and_Wait:Root.Packet.SaW_Packet{0.121904761904

Command | Log | Progress | Animation | Breakpoints

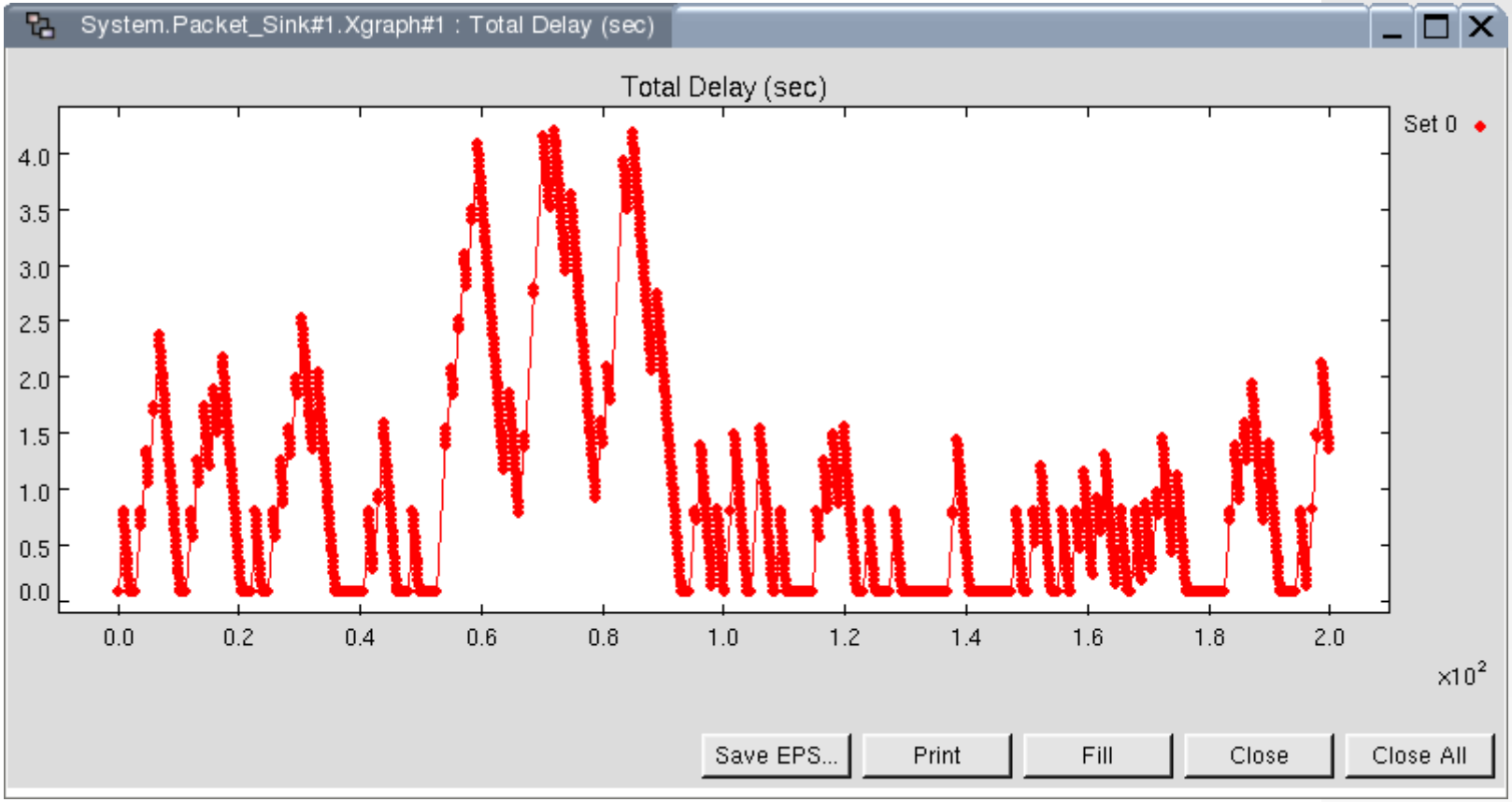
Stop and Wait: Results (1)



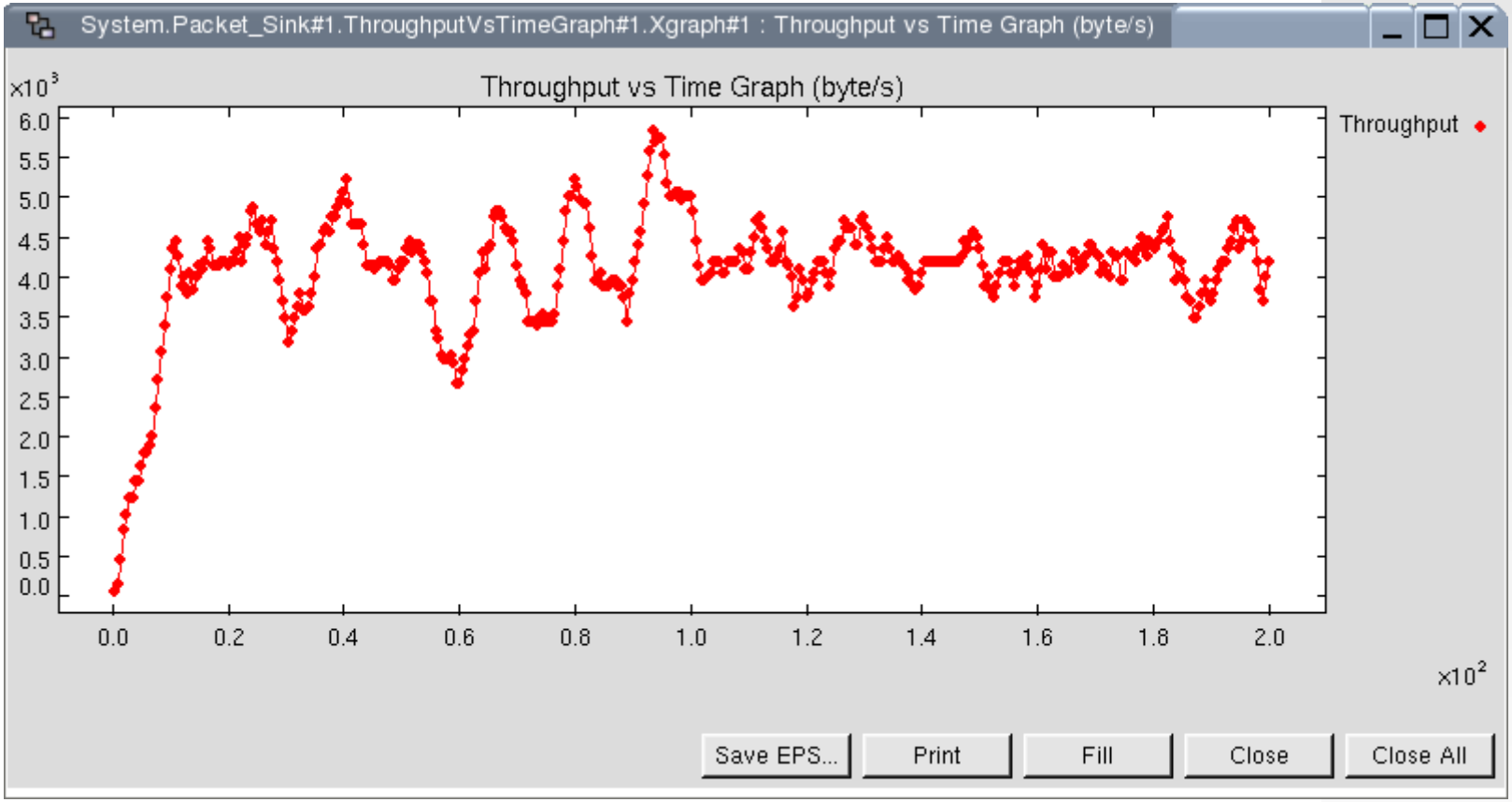
Stop and Wait: Results (2)



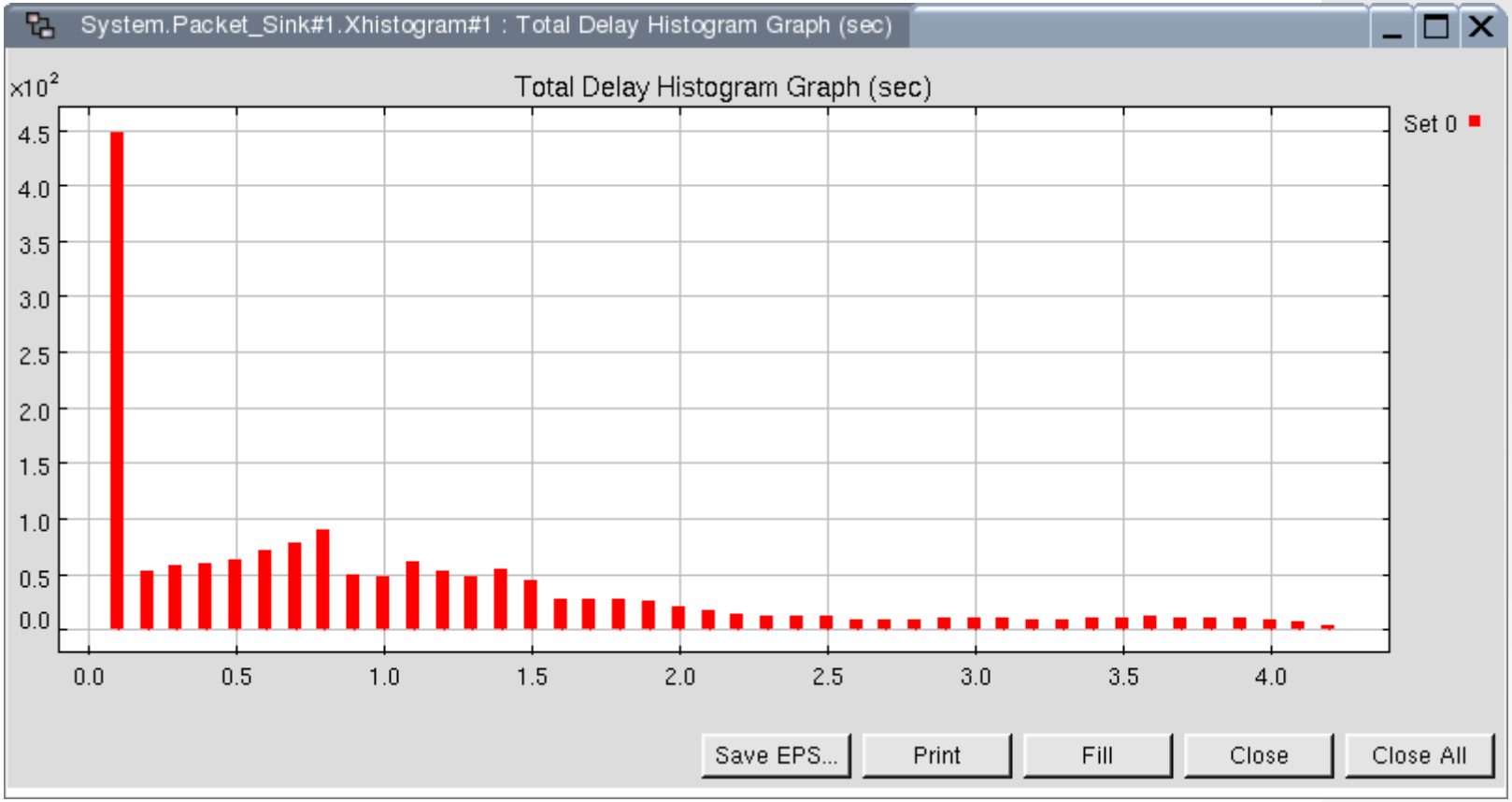
Stop and Wait: Results (3)



Stop and Wait: Results (4)

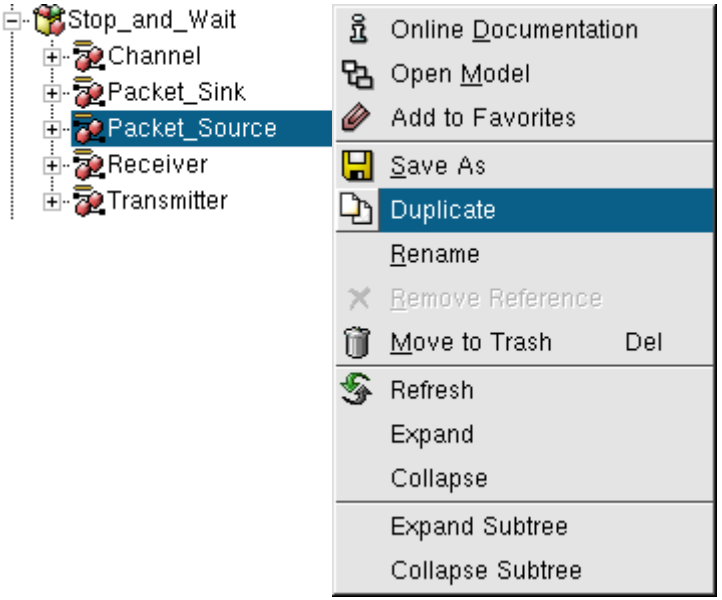


Stop and Wait: Results (5)

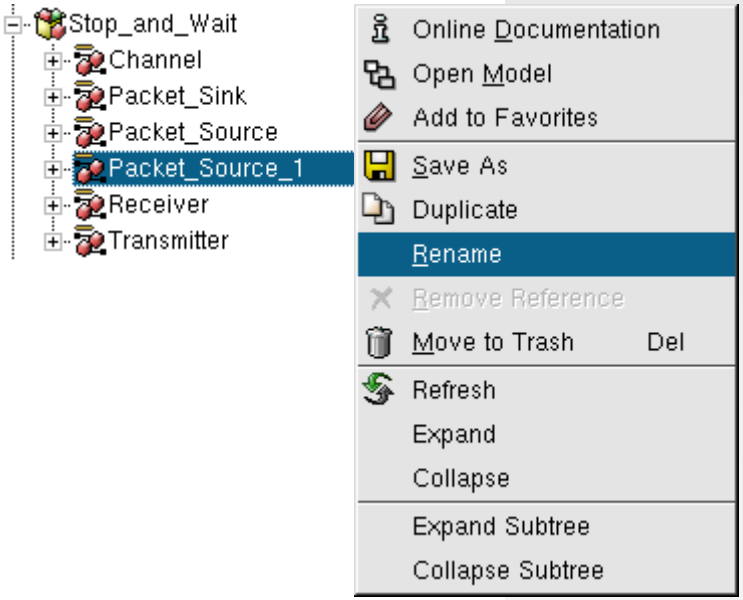


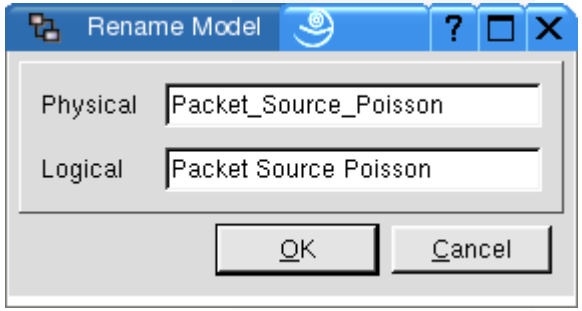
Stop and Wait: Packet Source Poisson (1)

- Creating a copy of Packet Source



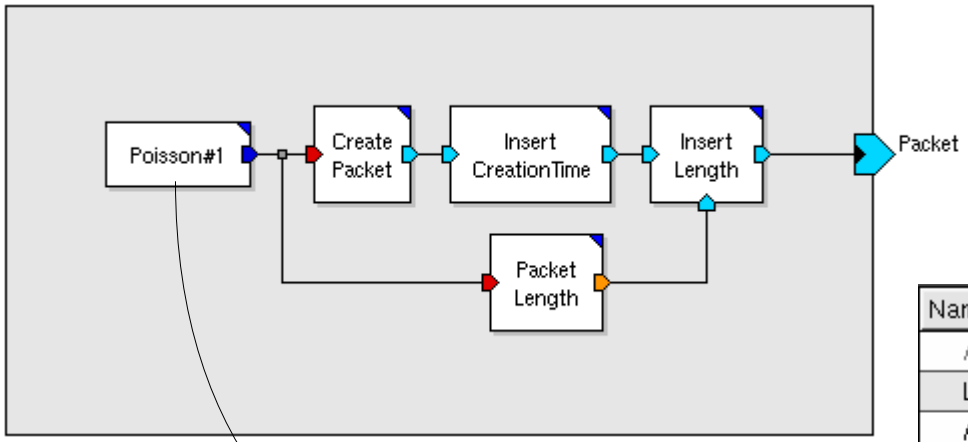
- Renaming the copy



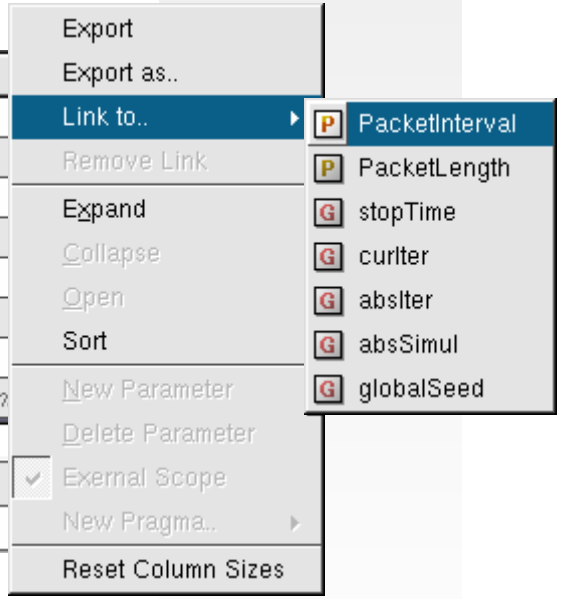


Stop and Wait: Packet Source Poisson (2)

- Model modifications
  - replace instance Clock#1 with Poisson#1
  - link parameter MeanTime to PacketIntervall

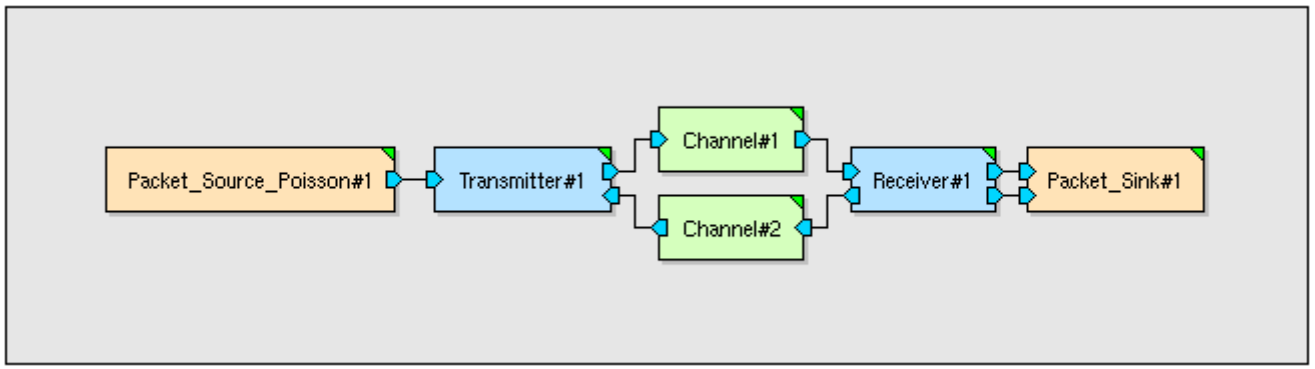


Name	Value
Name	*Poisson#1*
Label	
Annotation	
Show Bounding	true
Show Icon	true
Icon	
Color	#ffffff
Description	*Generate even*
⊞ P MeanTime	1.0
⊞ P Magnitude	1.0
⊞ P Seed	-1



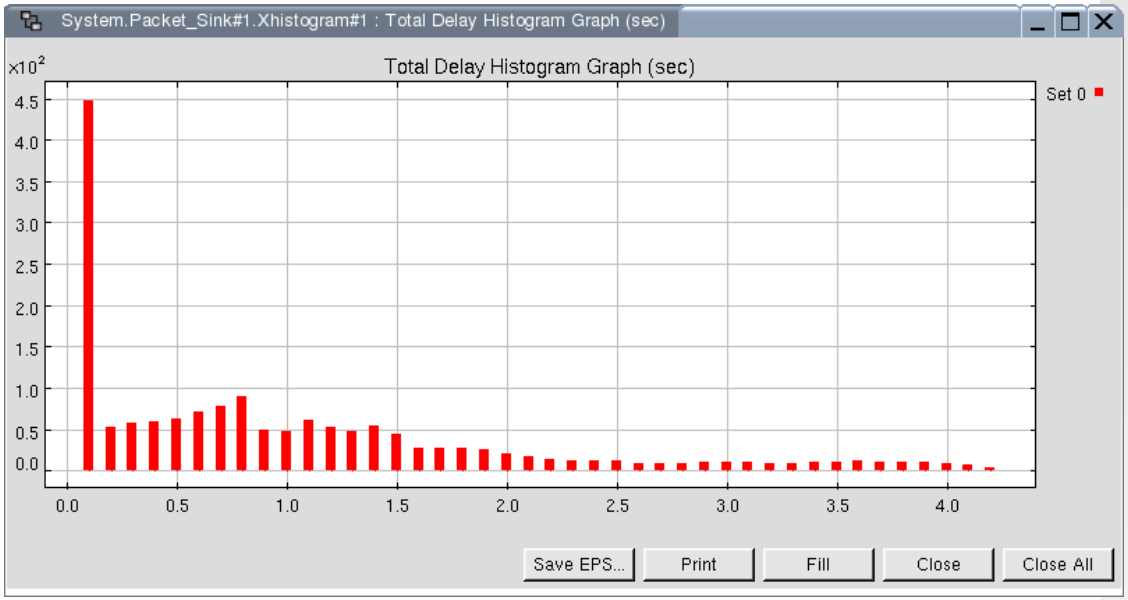
Stop and Wait: System Poisson (1)

- Duplicate System model
  - using Duplicate in Tree View on System model
  - rename to System_Poisson
- Modify System Poisson
  - replace instance Packet_Source#1 with Packet_Source_Poisson#1

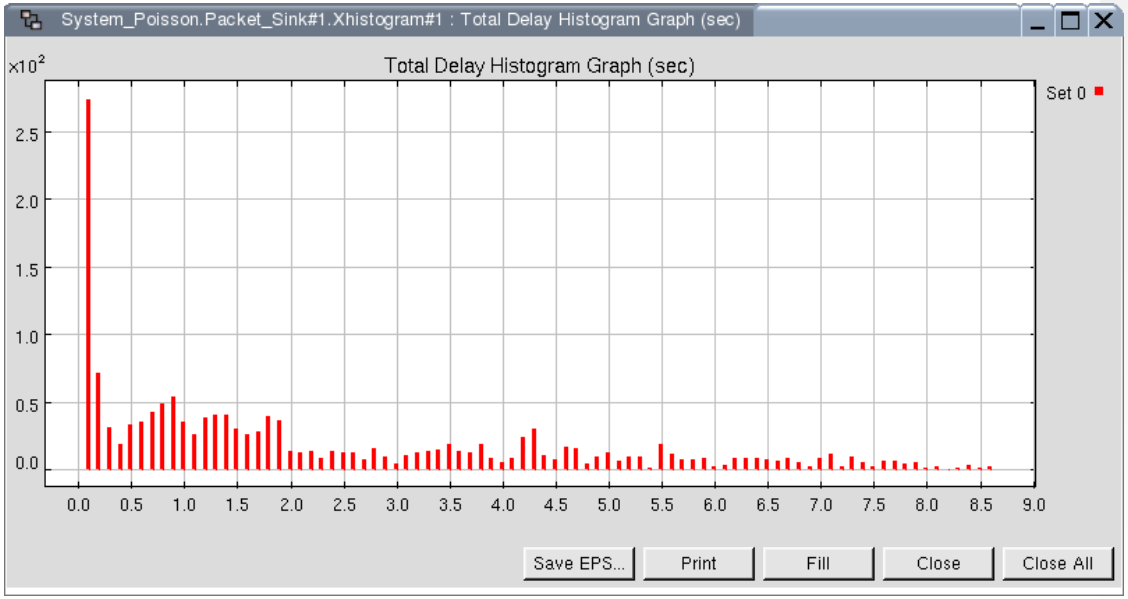


Stop and Wait: Periodische vs. Poisson-Quelle (1)

Total Delay periodical source

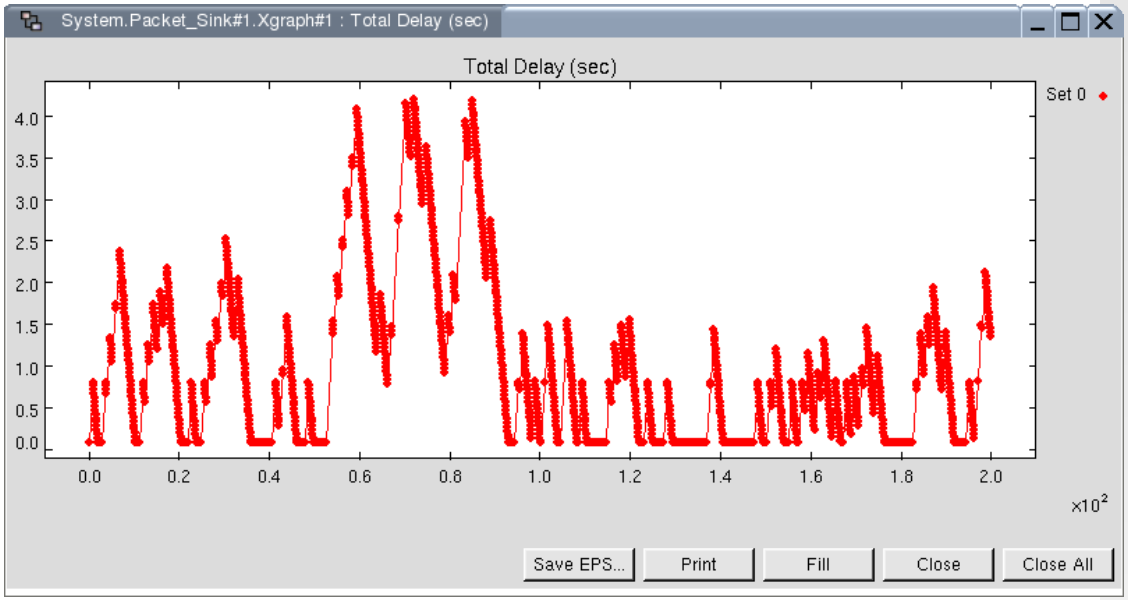


Total Delay Poisson source

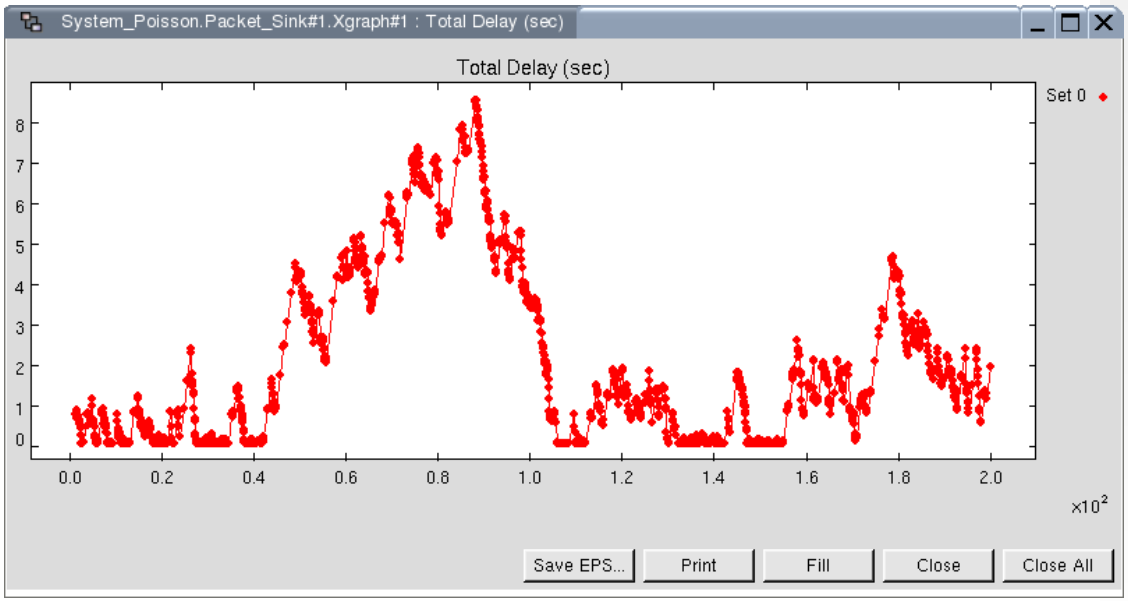


Stop and Wait: Periodische vs. Poisson-Quelle (2)

Total Delay
periodical source

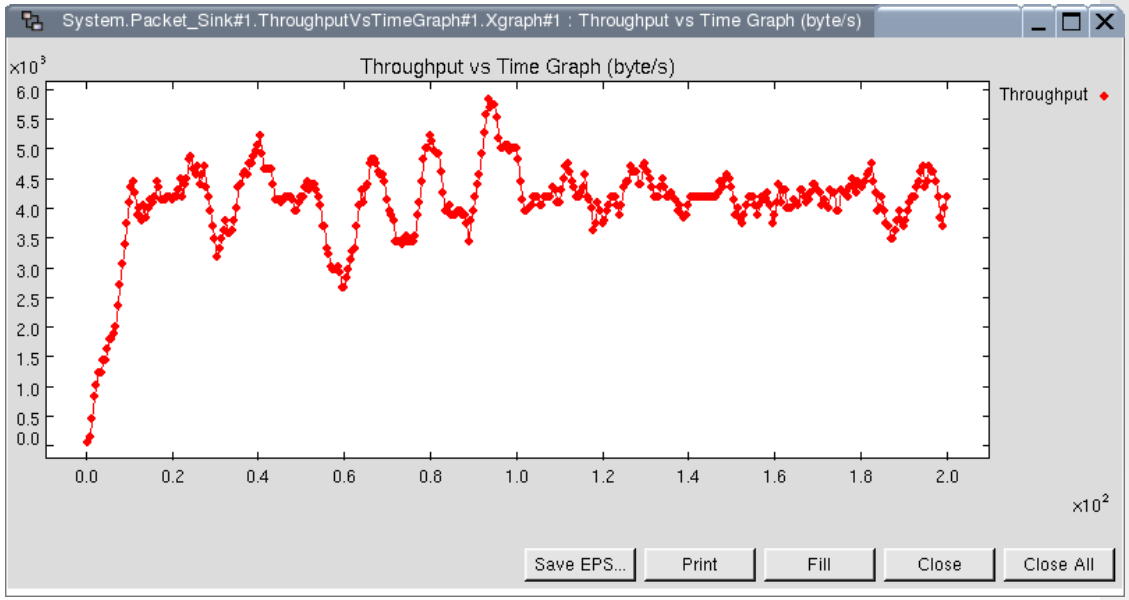


Total Delay
Poisson source

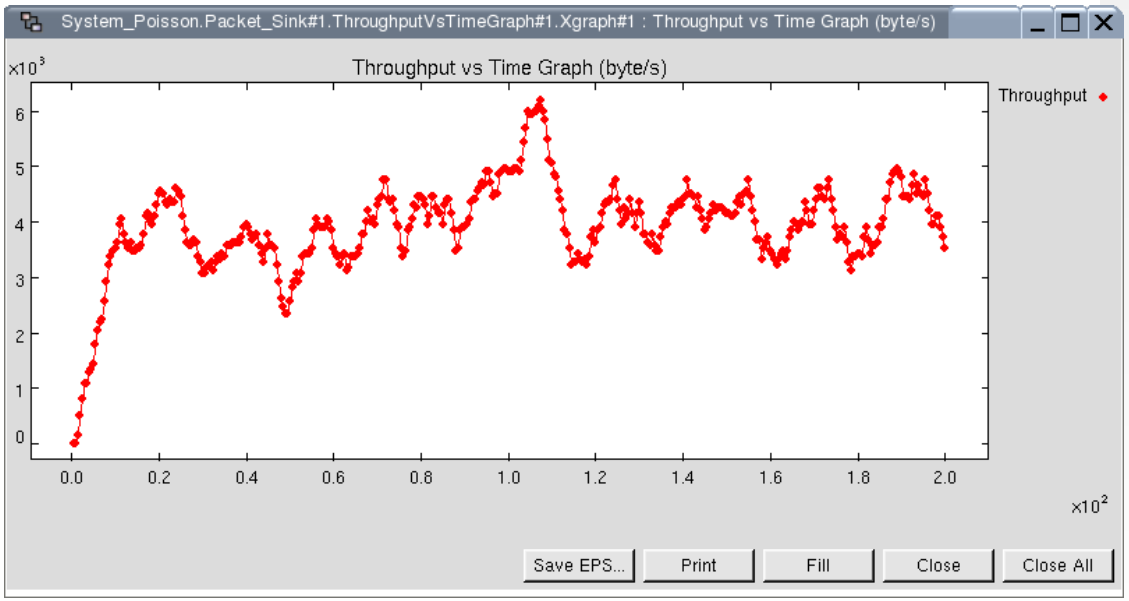


Stop and Wait: Periodische vs. Poisson-Quelle (3)

Throughput periodical source

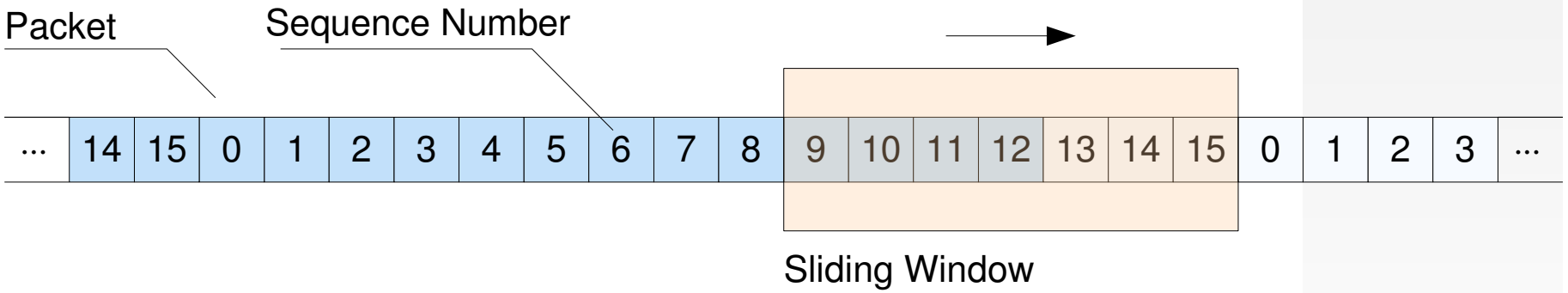


Throughput Poisson source



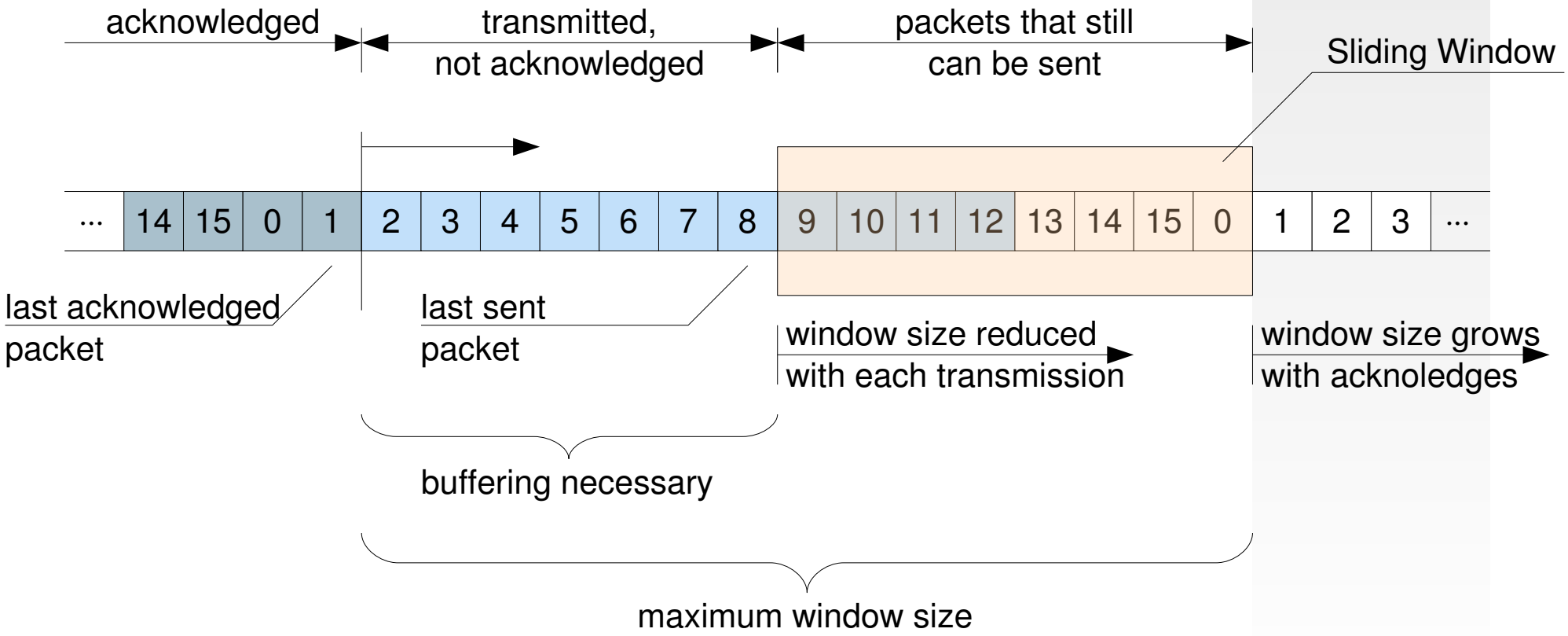
Pipelined Protocols: Sliding Window Mechanism

- Principle
  - transmission of multiple packets before acknowledge is necessary
  - positive ACK for flow control
  - negative ACK for error control
  - usage of sequence numbers: packets are numbered with n Bits continuously
  - maximum window size 2^n-1: sequence numbers must be unique



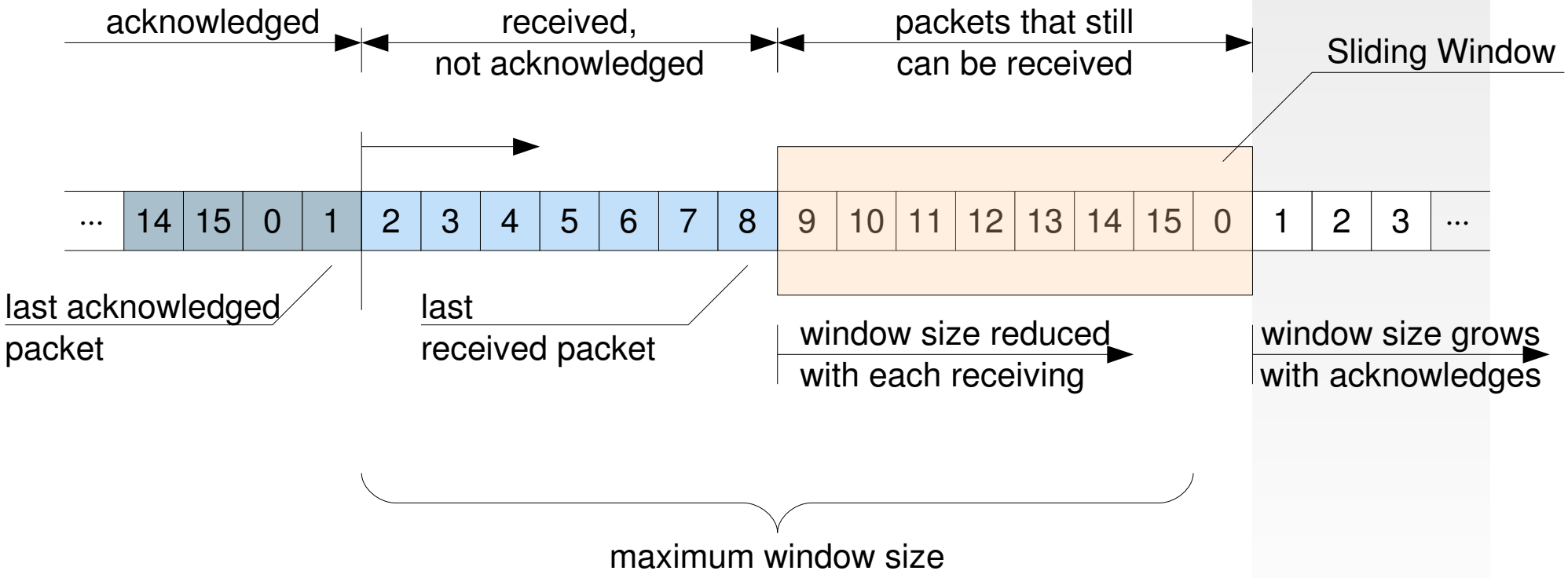
Sliding Window: Transmitter

- Window with transmitted but not acknowledged packets
  - size is reduced with each sent packet
  - size is growing with each acknowledged packet
  - maximum size reached if all packets are acknowledged

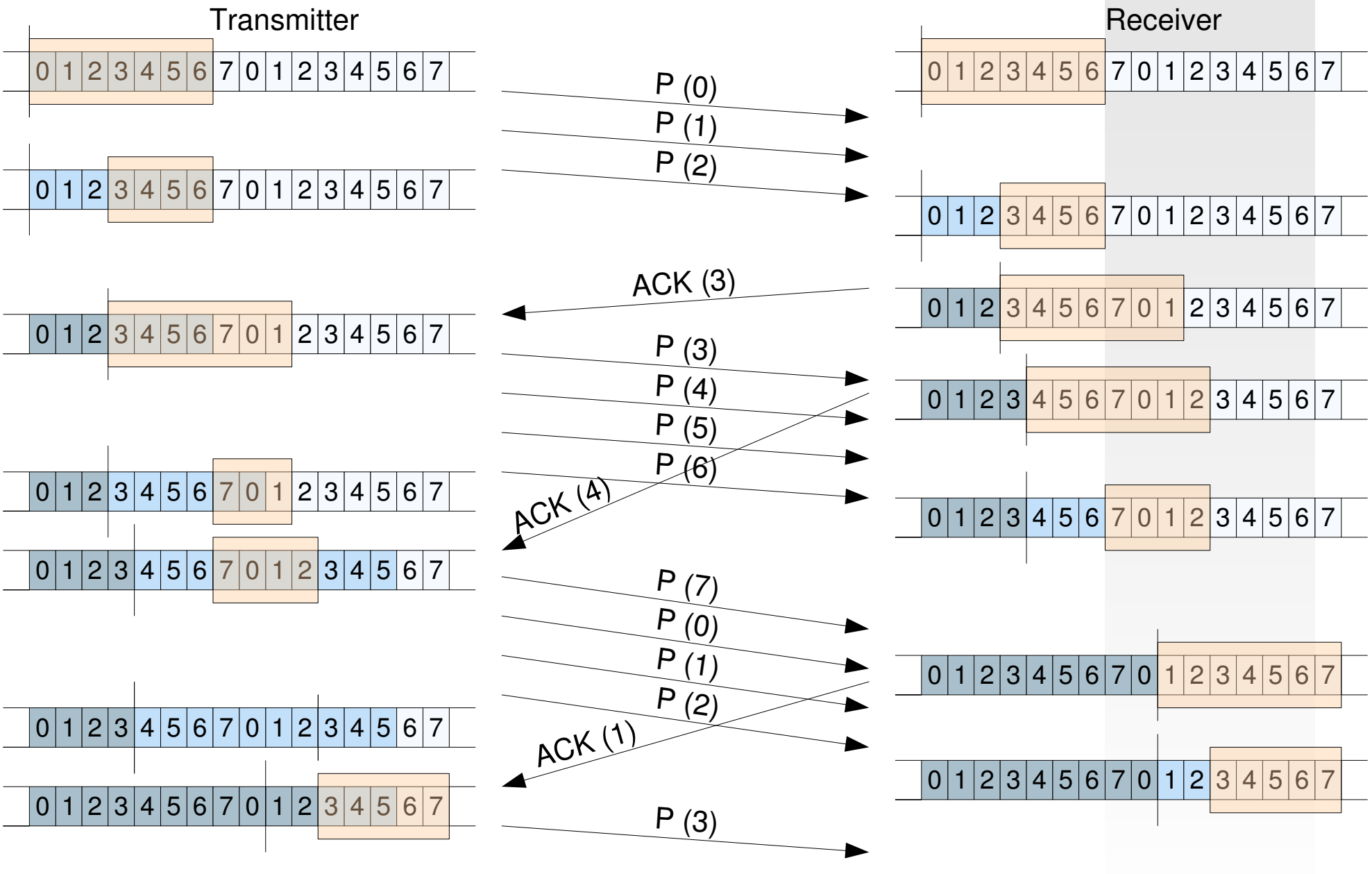


Sliding Window: Receiver

- Window with received but not acknowledged packets
  - size is reduced with each received packet
  - size is growing with each acknowledged packet
  - maximum size, if all received packets were acknowledged

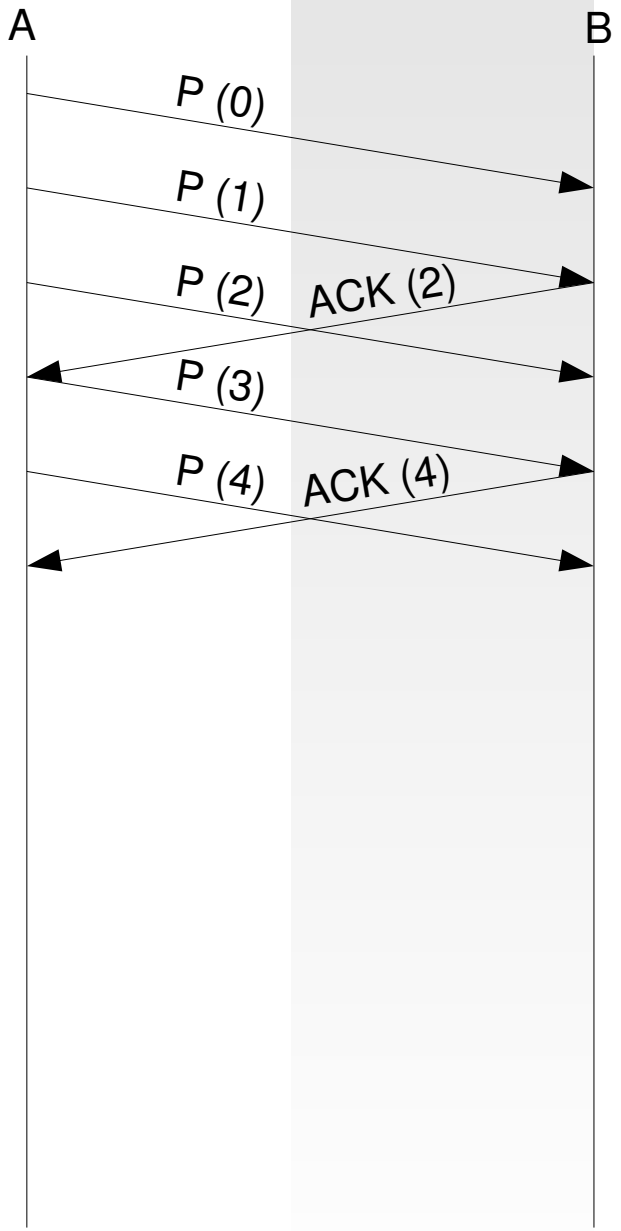


Sliding Window: Example



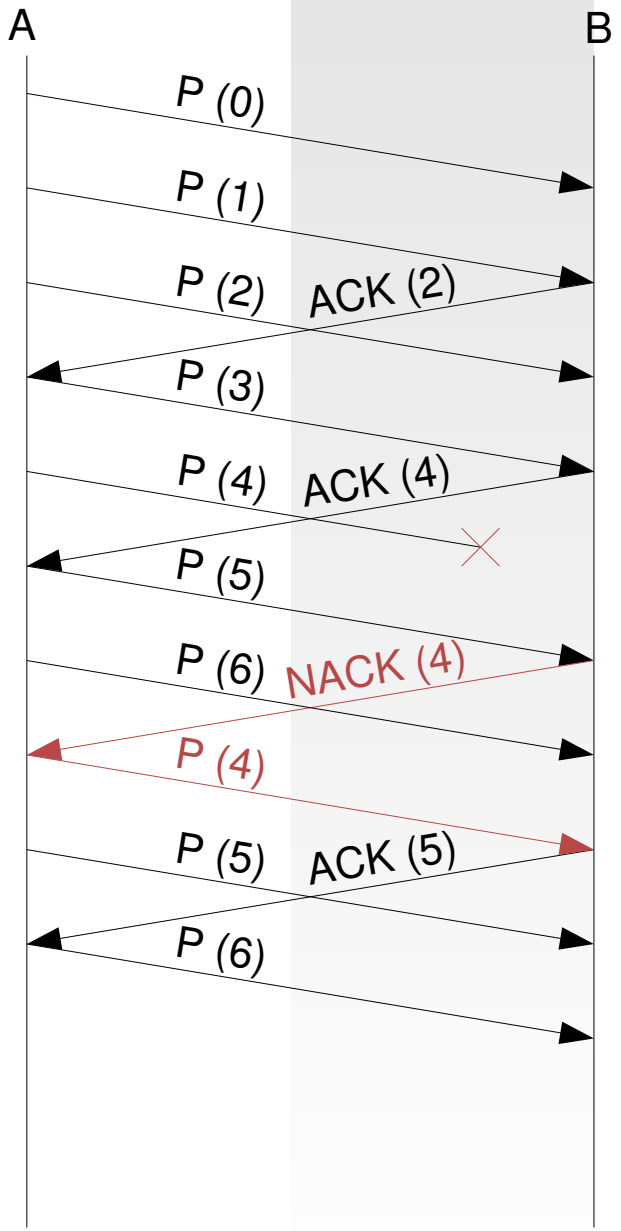
Go-Back-N: Flow Control

- Transmitter
  - transmission of multiple packets
    - reduce size of window until exhausted
    - waiting for acknowledge (ACK)
  - after acknowledge was received
    - increase size of window
    - transmission of further packets
  - acknowledges are cumulative
- Receiver
  - receiving of multiple packets
    - reduce size of window until exhausted
    - sending of acknowledge
  - is afterwards ready for further packet receiving



Go-Back-N: Error Control – Packet gets lost (1)

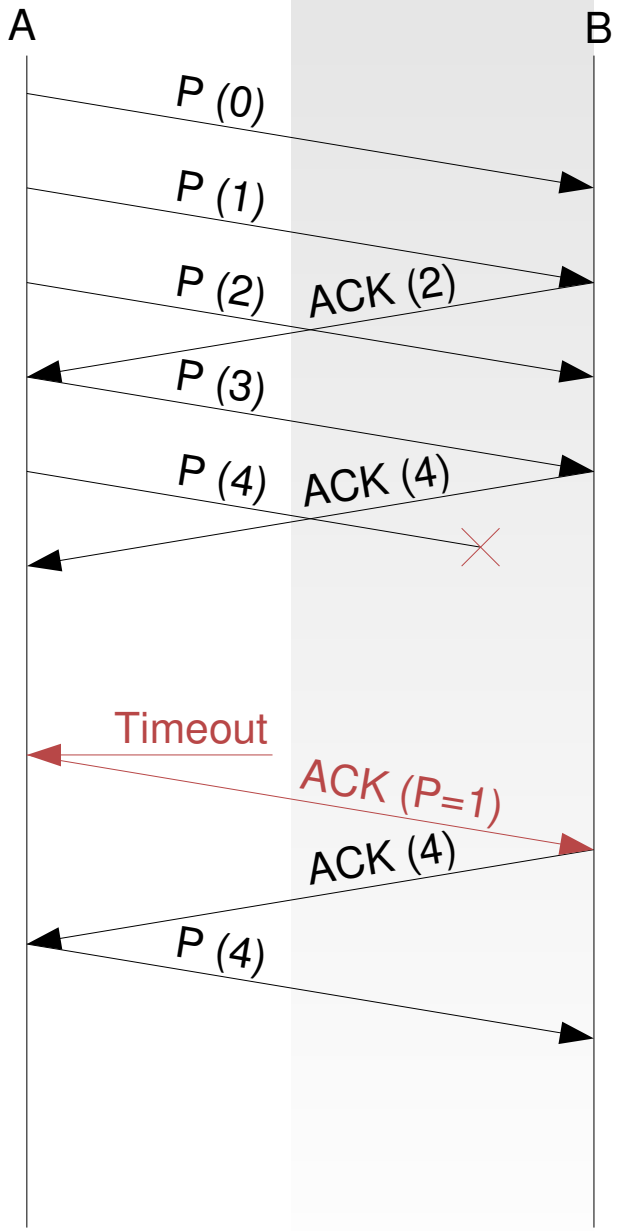
- Case 1: A sends further packets
  - B receives packets in a not expected order
  - B refuses packets with NACK



- Comments:
  - Piggybacking possible
  - acknowledge must not be perfomed using ACK
  - also possible using packets

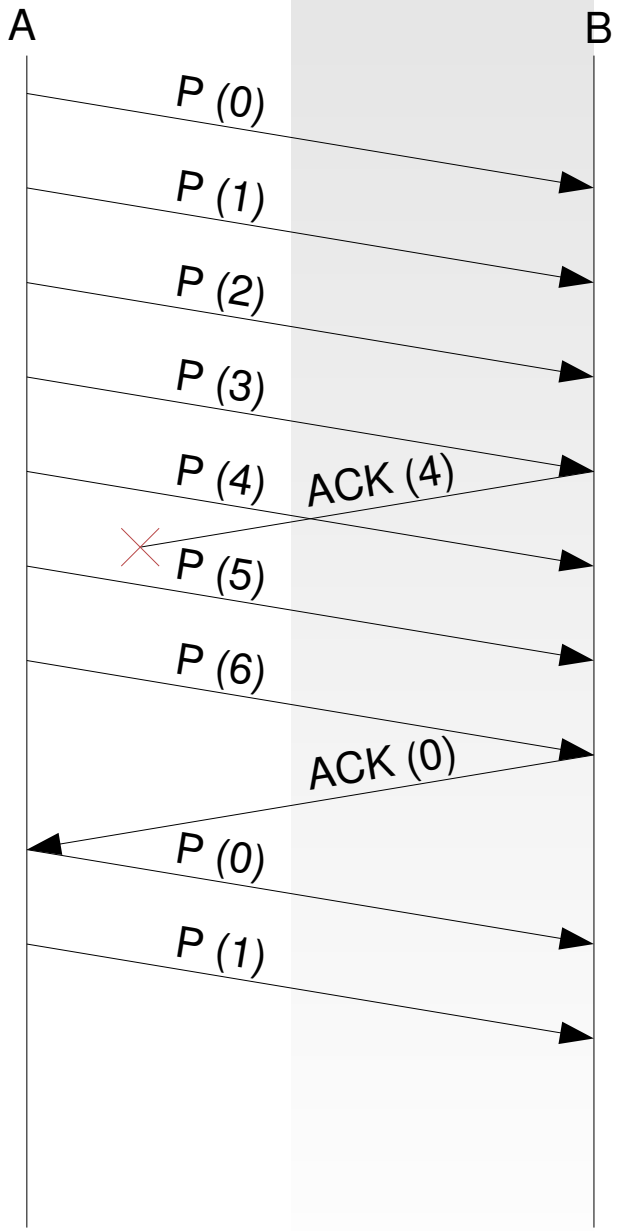
Go-Back-N: Error Control – Packet gets lost (2)

- Case 1: A sends further packets
  - B receives packets in a not expected order
  - B refuses packets with NACK
- Case 2: A sends no further packets
  - Timeout occurs
  - Option 1:
    - A sends ACK with Poll-Bit set
    - fores B to send an ACK
  - Option 2:
    - re-transmission of all packets from last acknowledged packet



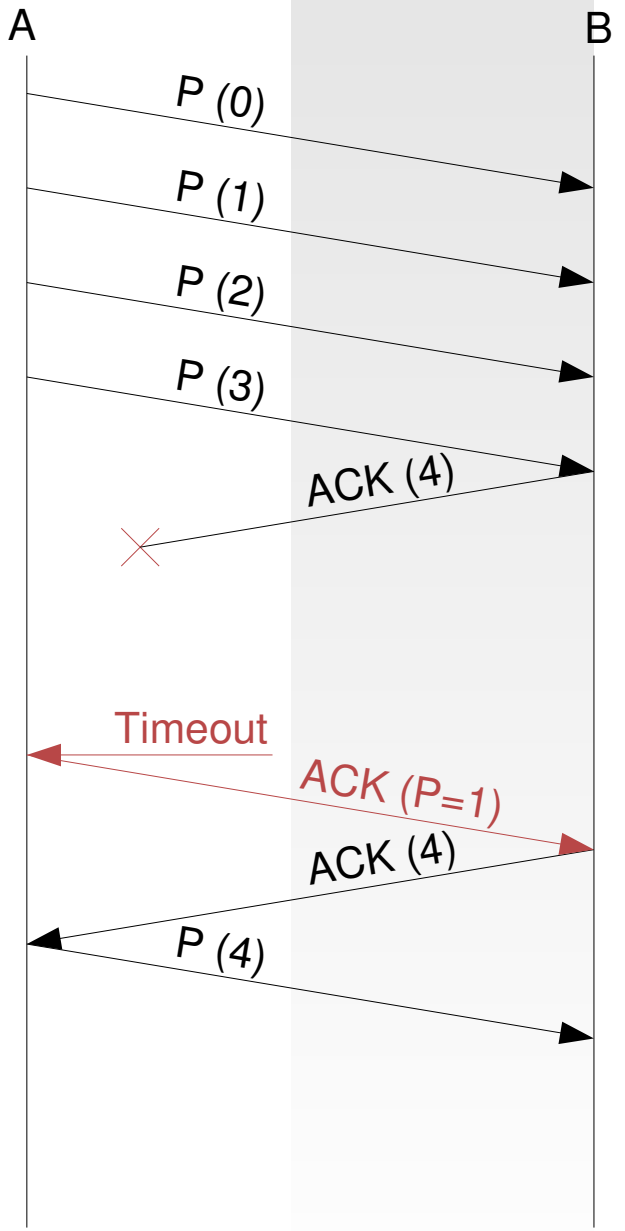
Go-Back-N: Error Control – Positiv-ACK gets lost (1)

- Case 1: A misses one acknowledge
  - acknowledges are cumulative
  - acknowledge all previous packets
  - include all previous acknowledges
  - no problem appears



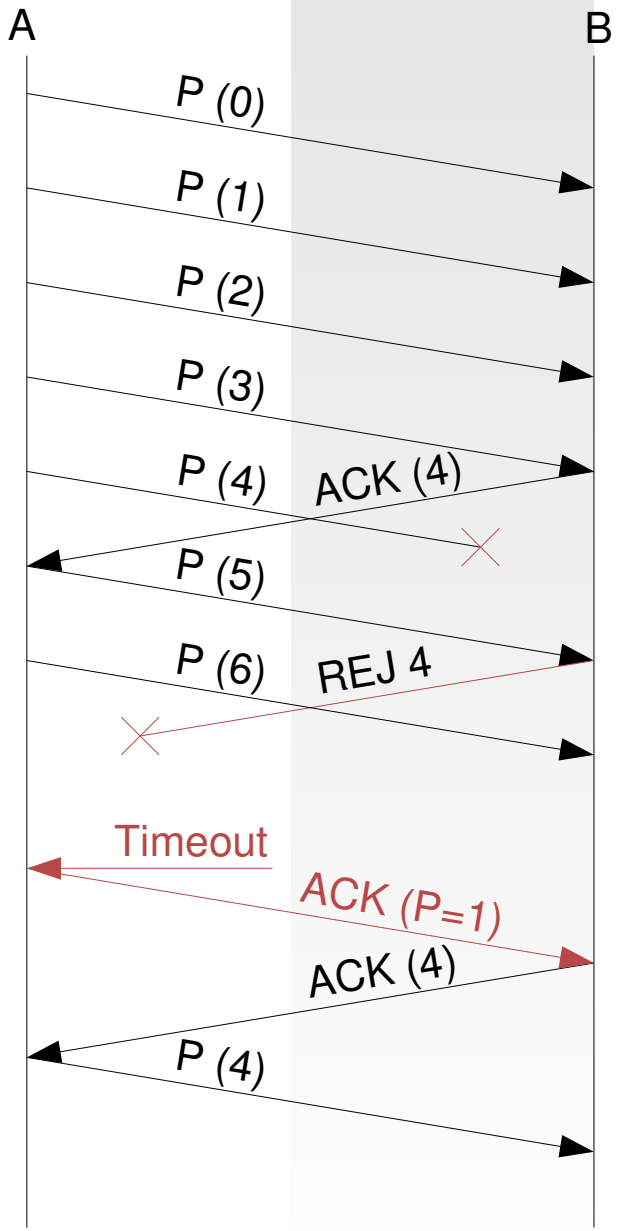
Go-Back-N: Error Control – Positiv-ACK gets lost (2)

- Case 1: A misses one acknowledge
  - acknowledges are cumulative
  - acknowledge all previous packets
  - include all previous acknowledges
  - no problem appears
- Case 2: A sends no further packets
  - Timeout occurs
  - Option 1:
    - A sends ACK with P-Bit set
    - forces B to send an ACK
  - Option 2:
    - re-transmission of all packets from last acknowledged packet



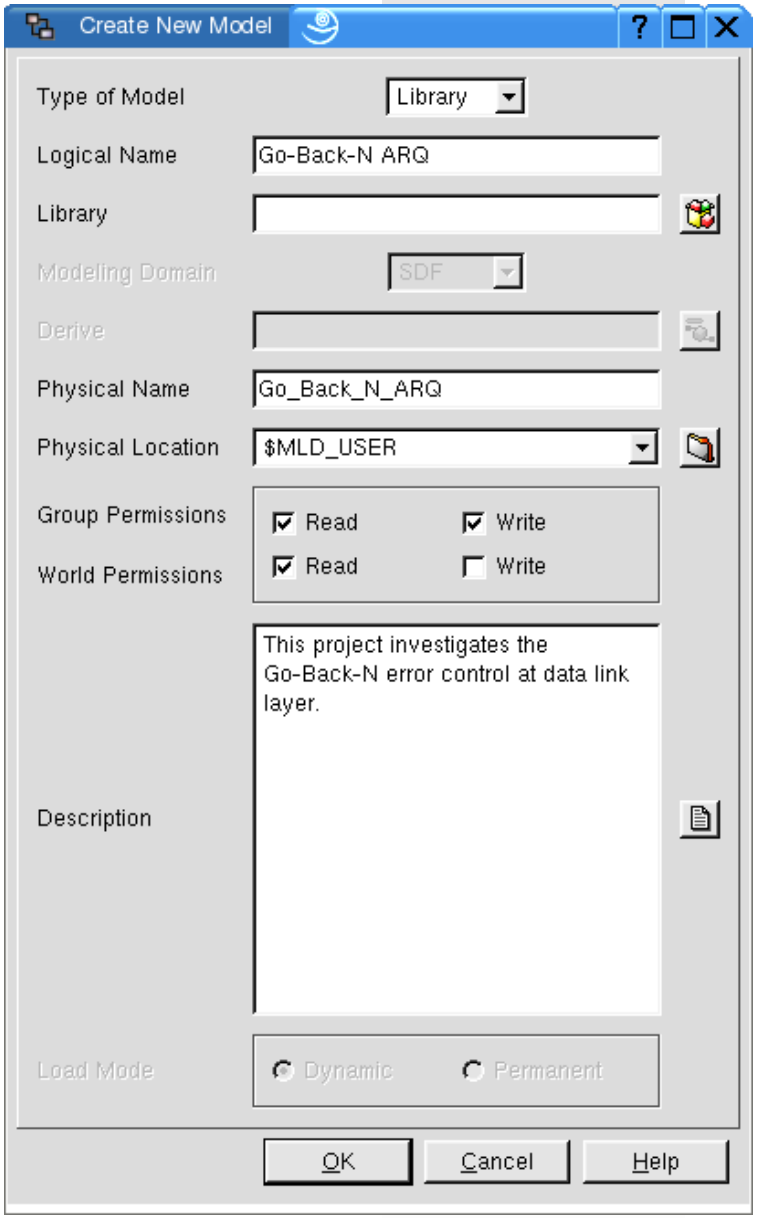
Go-Back-N: Error control – Negativ-ACK gets lost

- Timeout occurs
  - Option 1:
    - A sends ACK with P-Bit set
    - forces B to send ACK
  - Option 2:
    - re-transmission of all packets from last acknowledged packet

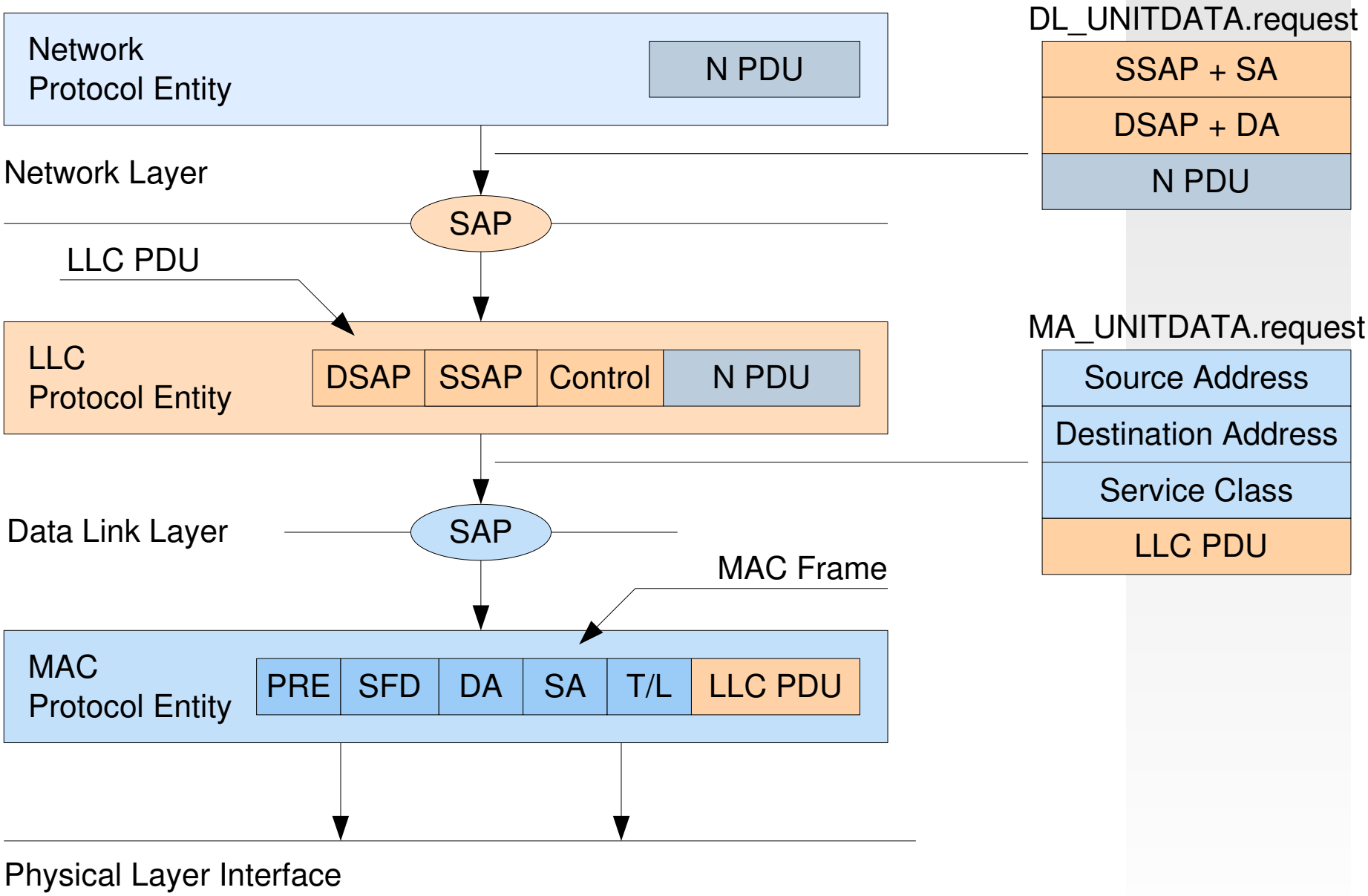


Go-Back-N: Creation of the Project

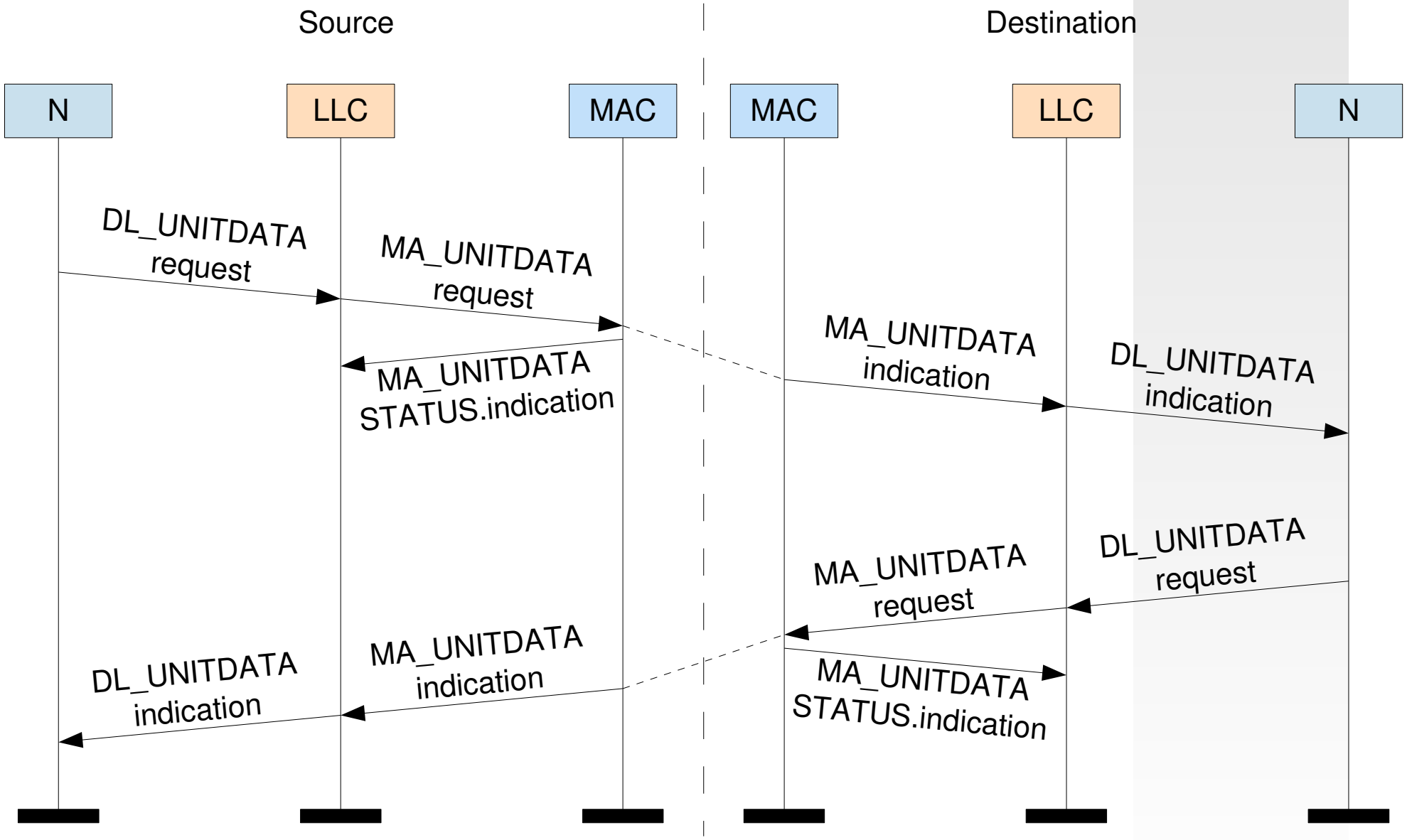
- Create Library Go-Back-N ARQ
- Copy models from Stop And Wait
  - Channel
  - Packet Source Poisson
  - Packet Sink
  - Transmitter
  - Receiver
- Modifications
  - all models use new Data Structure
  - new modules for Transmitter / Receiver



IEEE 802: General Workflow (1)



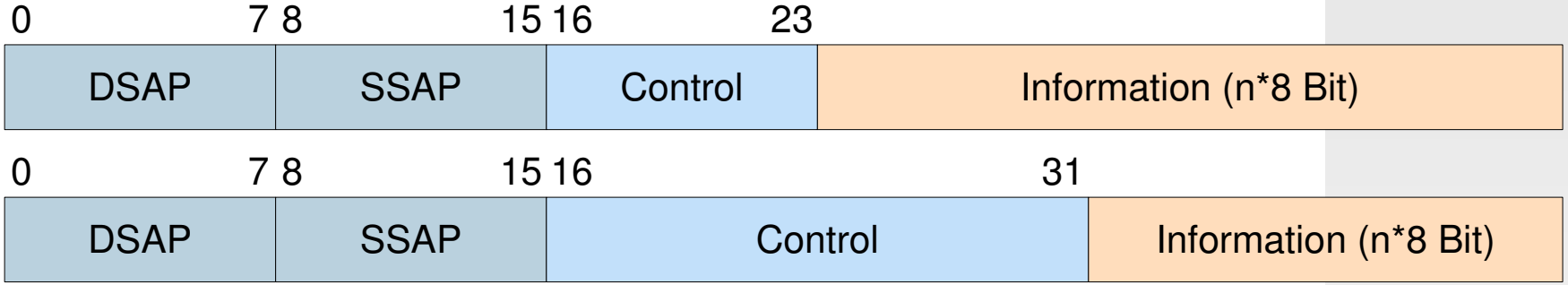
IEEE 802: General Workflow (2)



Go-Back-N: Data Structures Data Link Layer

IEEE 802.2 LLC-PDUs

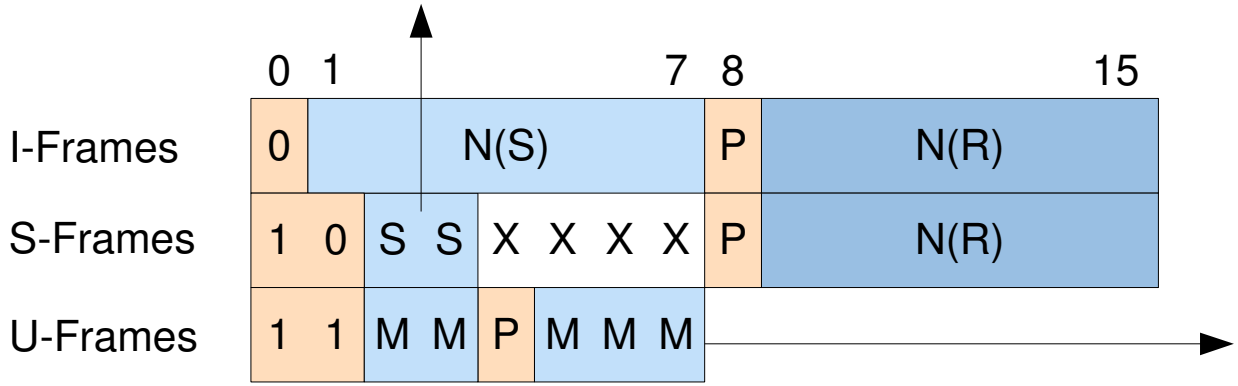
DSAP – Destination Service Access Point
SSAP – Source Service Access Point



Format Control Field

0 0 RR (Receive Ready)
0 1 REJ (Reject)
1 0 RNR (Receive Not Ready)

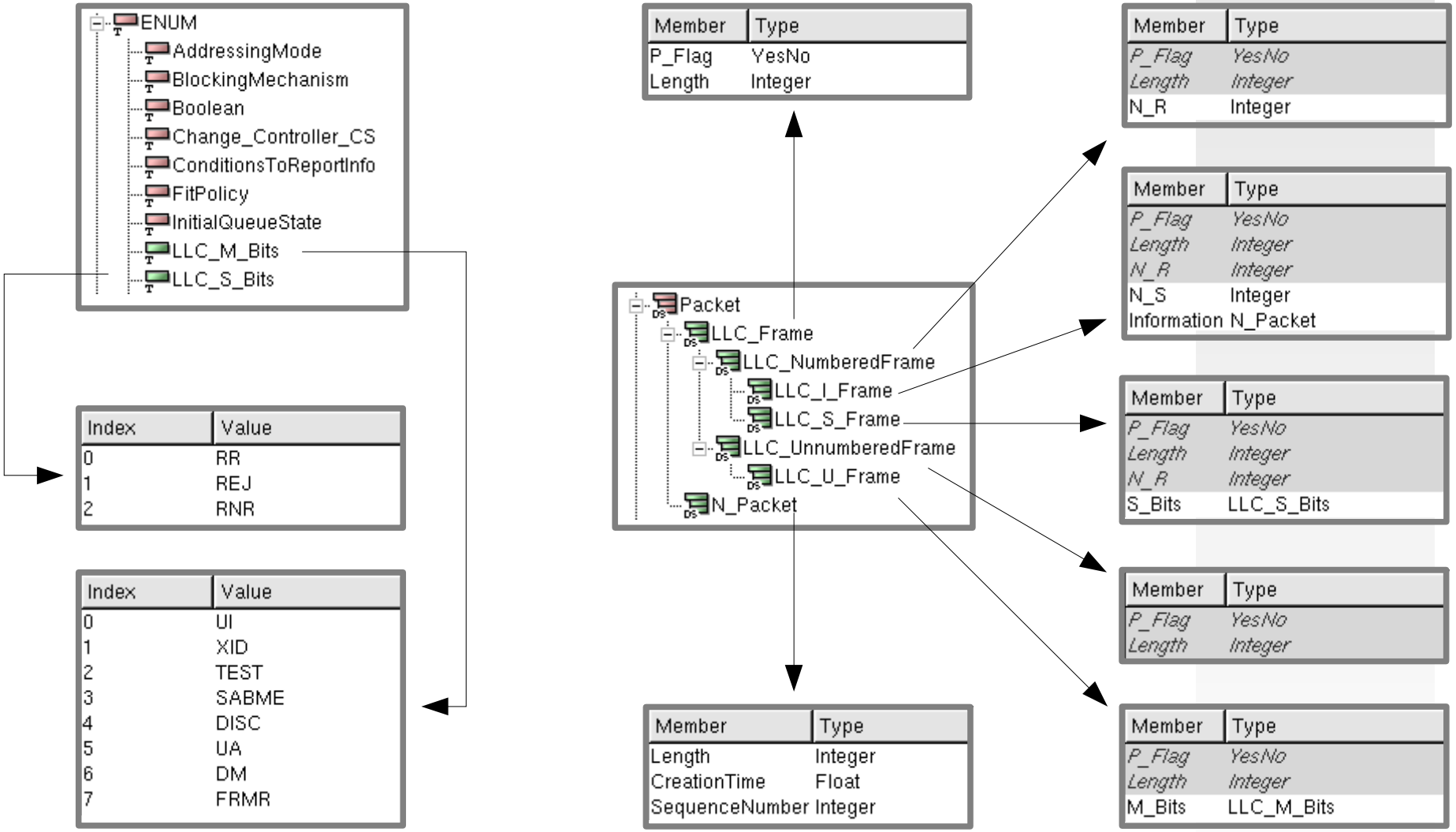
N(S) - Sender Send Sequence Number
N(R) - Sender Receive Sequence Number
S - Supervisory Function Bit (RR, REJ, RNR)
M - Modifier Function Bit



1 1 0 0 P 0 0 0	UI command
1 1 0 0 P 0 1 0	DISC command
1 1 0 0 P 1 1 1	TEST command
1 1 1 1 P 1 0 1	XID command
1 1 1 1 P 1 1 0	SABME command
1 1 0 0 F 1 0 1	UA response
1 1 1 1 F 0 0 0	DM response
1 1 1 0 F 0 0 1	FRMR response

Go-Back-N: Creation of Data Structures

Mission Level Design GmbH



Go-Back-N: Data structures of Network Layer

- packet to be transmitted: Root.Packet.N_Packet

Member	Type	Default	Subrange	Description
Length	Root.Integer	0	[0,Inf)	Contains the packet length in Byte.
CreationTime	Root.Float	0	[0,Inf)	Specifies the time of creation (the arrival for sending) in seconds.
PacketNumber	Root.Integer	1	[1,Inf)	Sequence number used for statistical purpose only.

Go-Back-N: Data Structures of LLC (1)

- LLC frame: Root.Packet.LLC_Frame

Member	Type	Default	Subrange	Description
P_Flag	Root.ENUM. YesNo	No	No, Yes	This flag is used a poll flag in commands and as final flag in responses.
Length	Root.Integer	0	[0,Inf)	Contains the frame length in Byte.

Go-Back-N: Data Structures of LLC (2)

- Numbered Frames Root.Packet.LLC_Frame.LLC_NumberedFrame

Member	Type	Default	Subrange	Description
P_Flag	Root.ENUM. YesNo	No	No, Yes	This flag is used a poll flag in commands and as final flag in responses.
Length	Root.Integer	0	[0,Inf)	Contains the frame length in Byte.
N_R	Root.Integer	0	[0, Inf)	Specifies the sequence number of the frame expected as next (the sequence number of the last frame received correctly +1).

Go-Back-N: Data Structures of LLC (3)

- I Frames Root.Packet.LLC_Frame.LLC_NumberedFrame.LLC_I_Frame

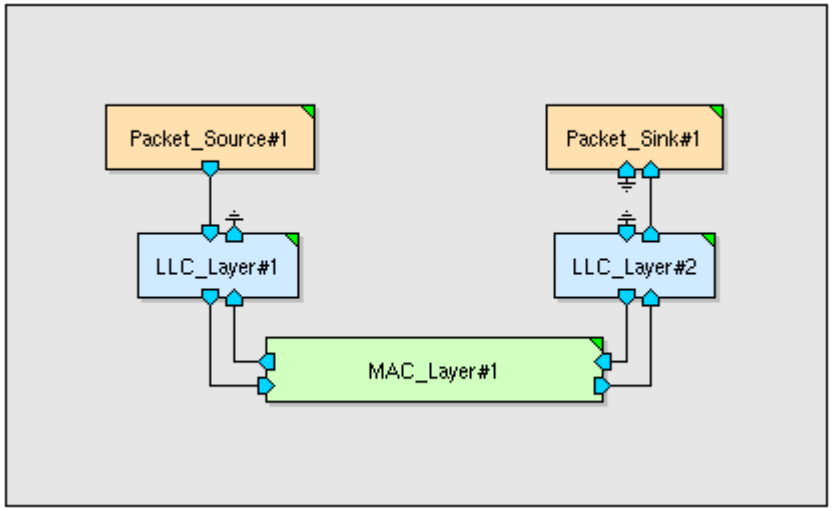
Member	Type	Default	Subrange	Description
P_Flag	Root.ENUM. YesNo	No	No, Yes	This flag is used a poll flag in commands and as final flag in responses.
Length	Root.Integer	0	[0,Inf)	Contains the frame length in Byte.
N_R	Root.Integer	0	[0, Inf)	Specifies the sequence number of the frame expected as next (the sequence number of the last frame received correctly +1).
N_S	Root.Integer	0	[0, Inf)	Sequence number of the packet.
Information	Root.Packet. N_Packet	-	-	The payload of information frames is a network layer packet.

Go-Back-N: Data Structures of LLC (4)

- S Frames Root.Packet.LLC_Frame.LLC_NumberedFrame.LLC_I_Frame

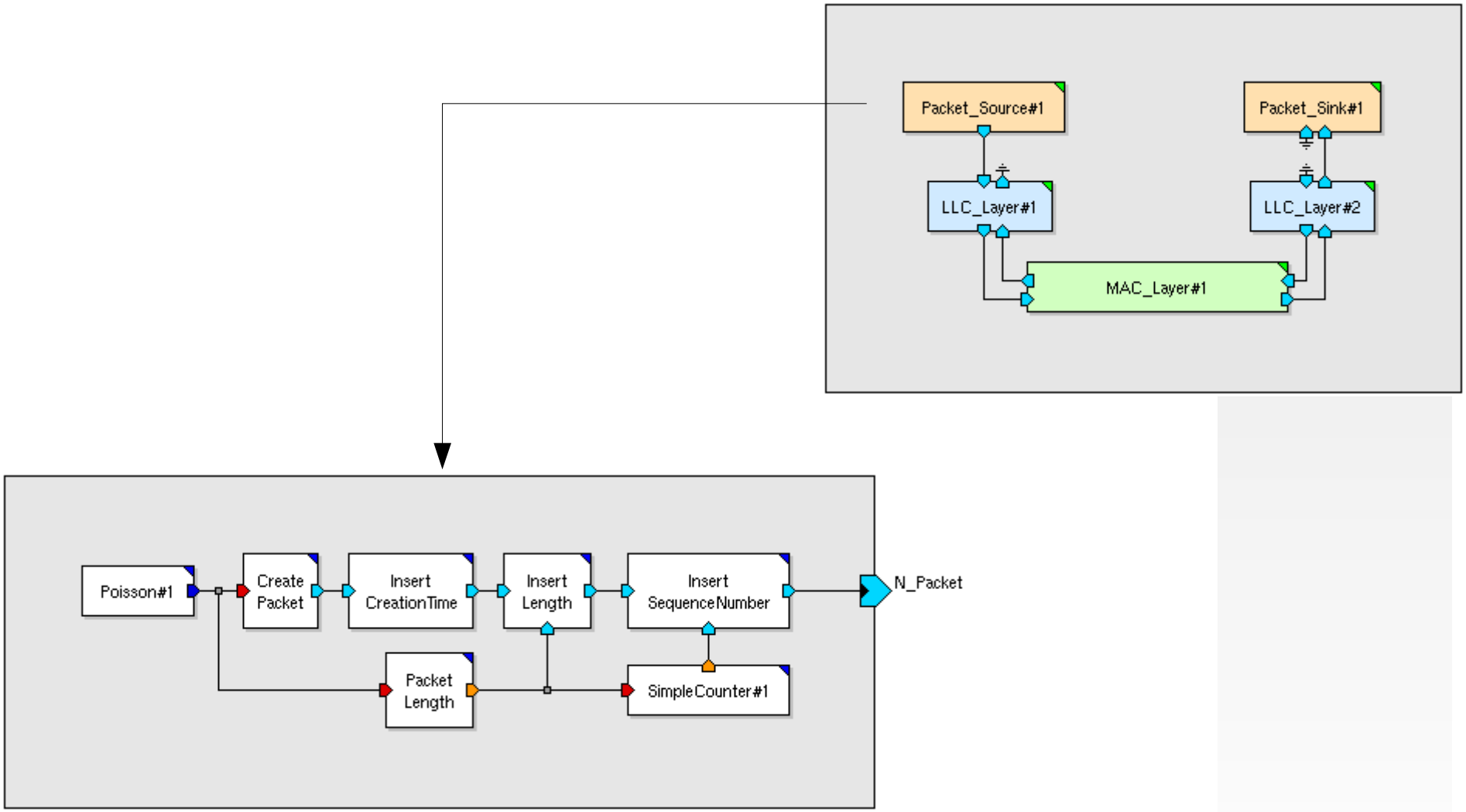
Member	Type	Default	Subrange	Description
P_Flag	Root.ENUM. YesNo	No	No, Yes	This flag is used a poll flag in commands and as final flag in responses.
Length	Root.Integer	0	[0,Inf)	Contains the frame length in Byte.
N_R	Root.Integer	0	[0, Inf)	Specifies the sequence number of the frame expected as next (the sequence number of the last frame received correctly +1).
S_Bits	Root.ENUM. LLC_S_Bits	RR	-	Specifies the type of the supervisory frame.

Go-Back-N: System

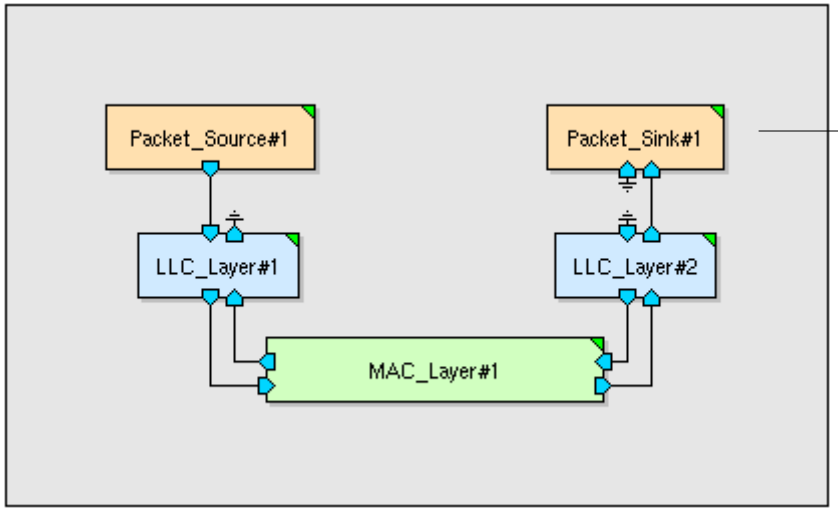


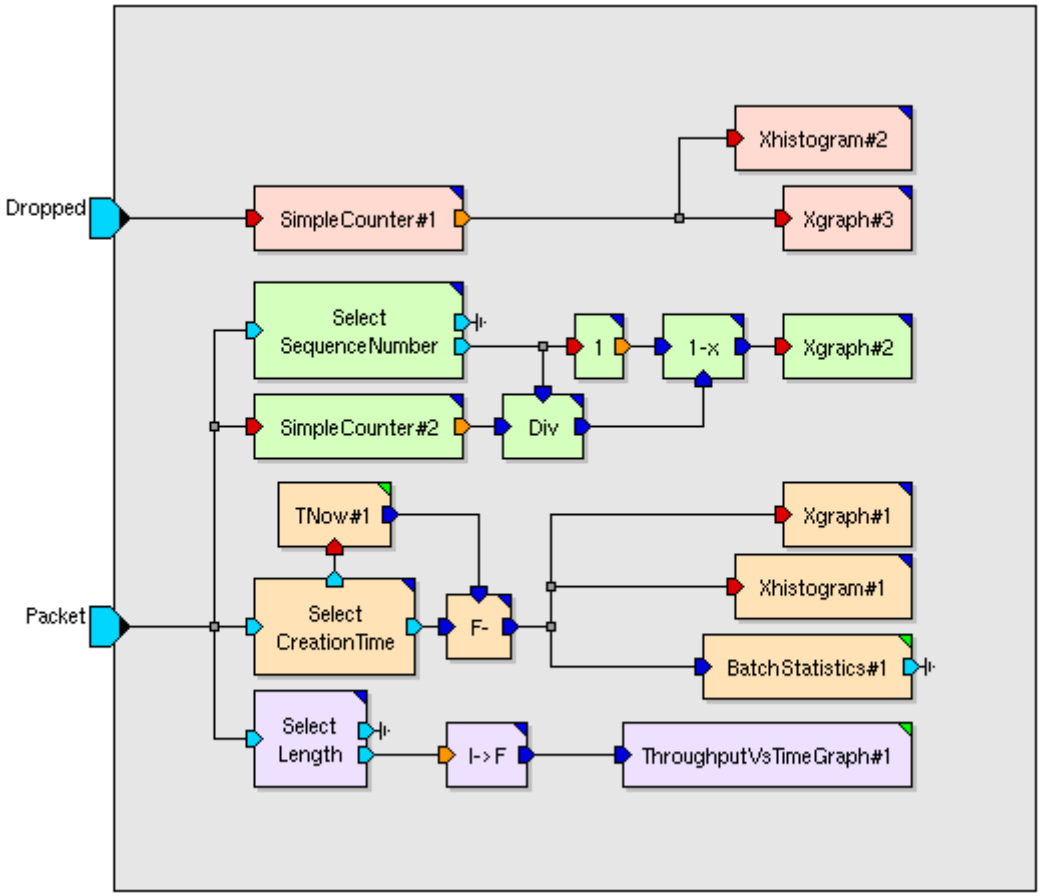
Parameter	Typ	Wert
PacketInterval	float	$PacketLength*8/$ChannelSpeed*0.6
PacketLength	int	512
MaximumQueueSize	int	500
TimeoutValue	float	$PacketLength*8/$ChannelSpeed*10.0
ChannelBitErrorRate	float	1e-4
PropagationDelay	float	0
ChannelSpeed	int	56000
AckLength	int	2
MaxSequenceNumber	int	7
MaxWindowSize	int	$MaxSequenceNumber

Go-Back-N: Network Layer – Packet Source

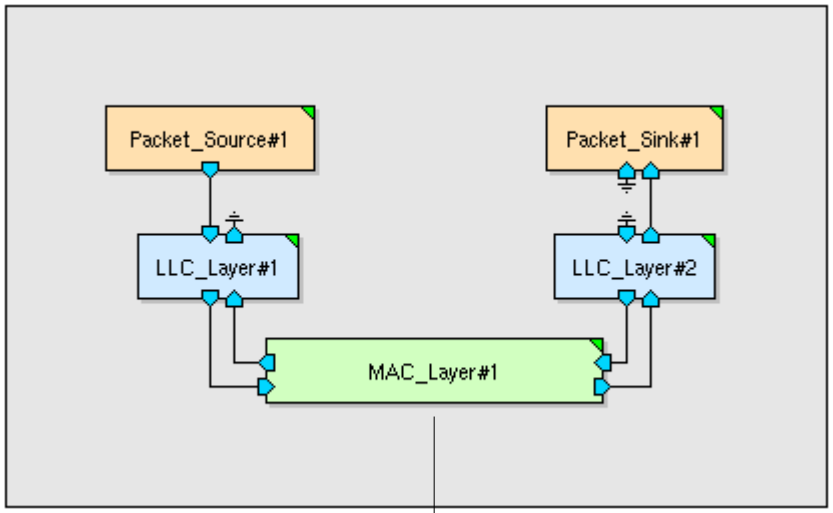


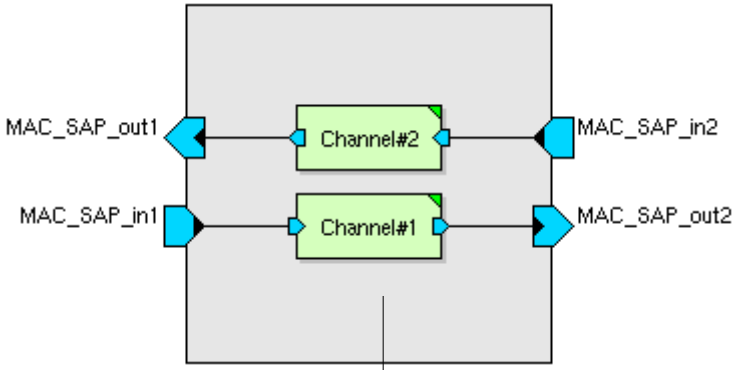
Go-Back-N: Network Layer – Packet Sink

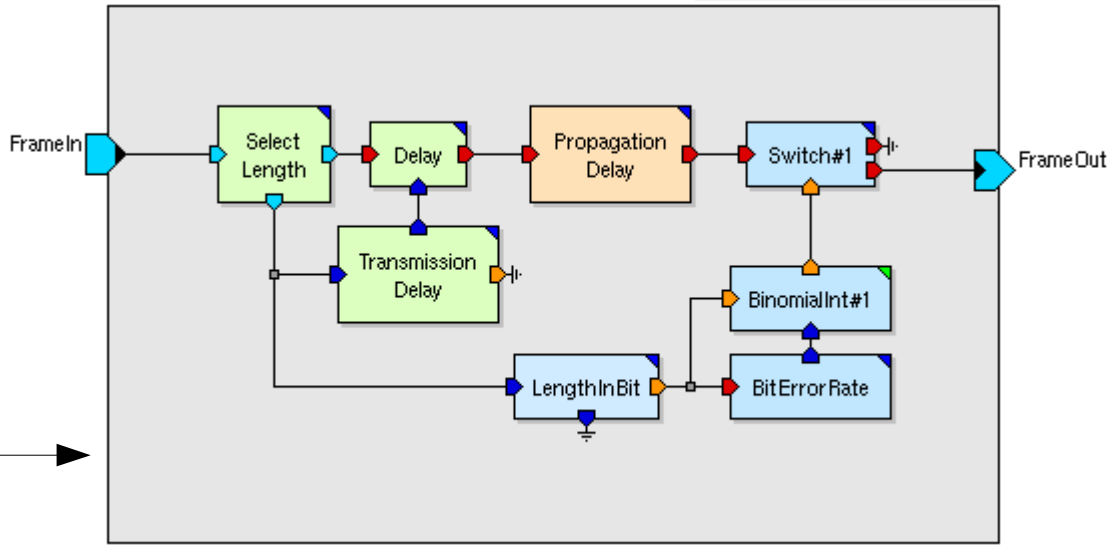




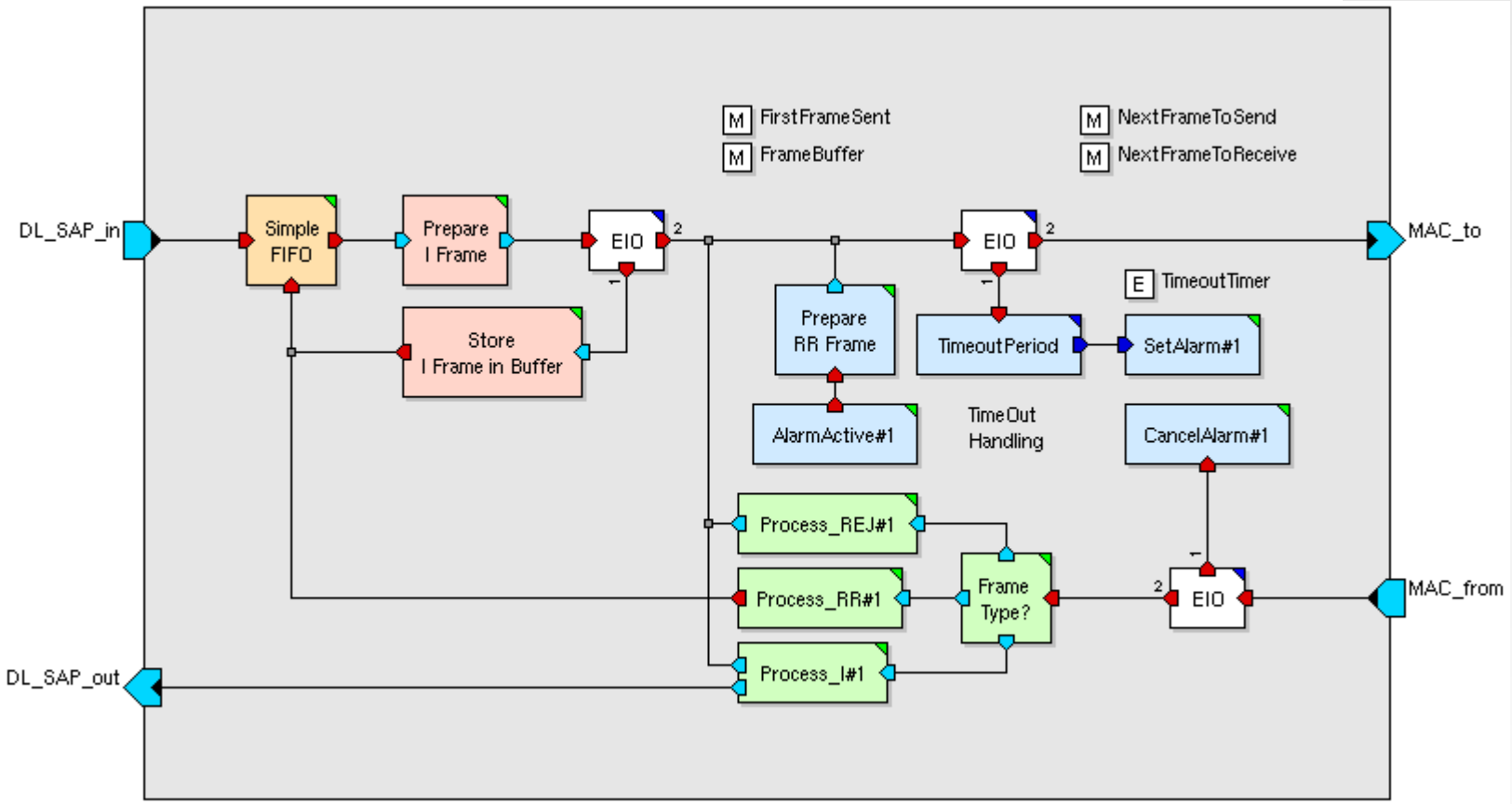
Go-Back-N: Data Link Layer – MAC Layer







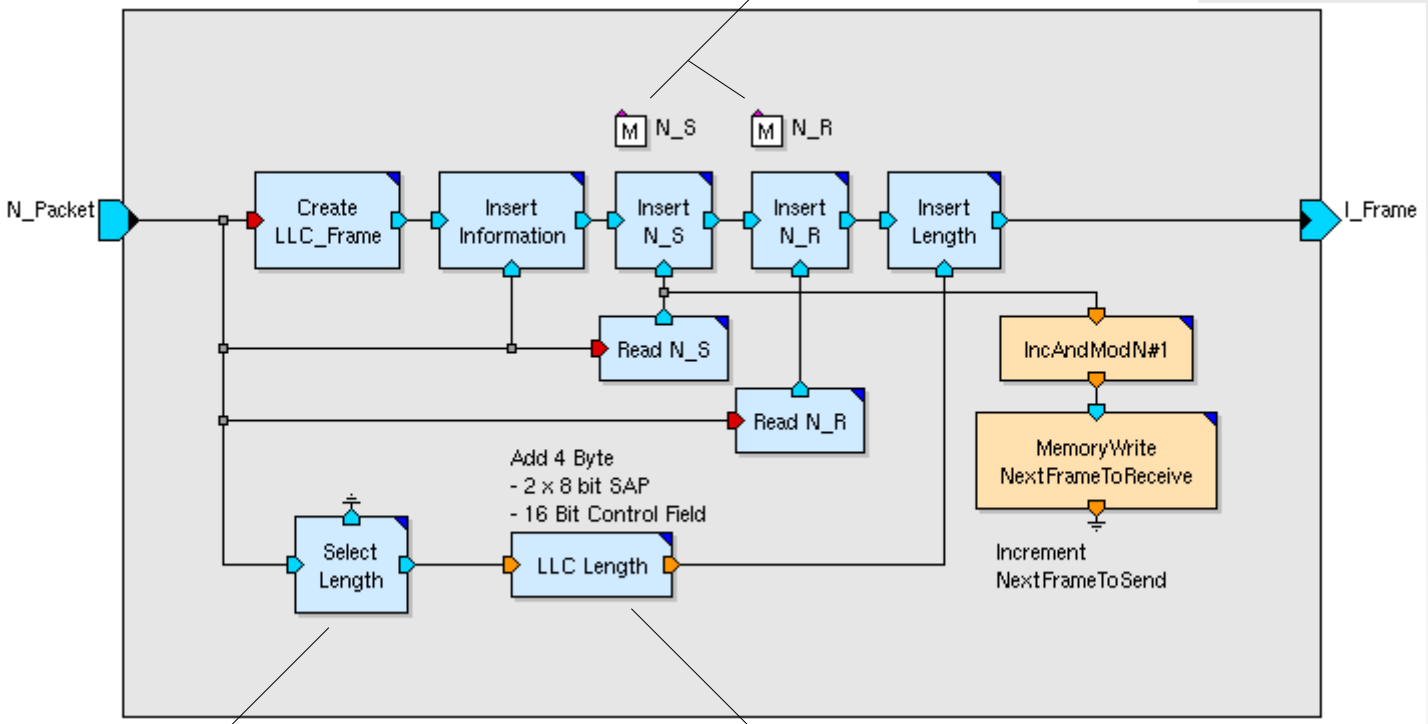
Go-Back-N: Data Link Layer – LLC Layer Typ2



Go-Back-N: LLC Layer Typ2 – Prepare I-Frame

Mission Level Design GmbH

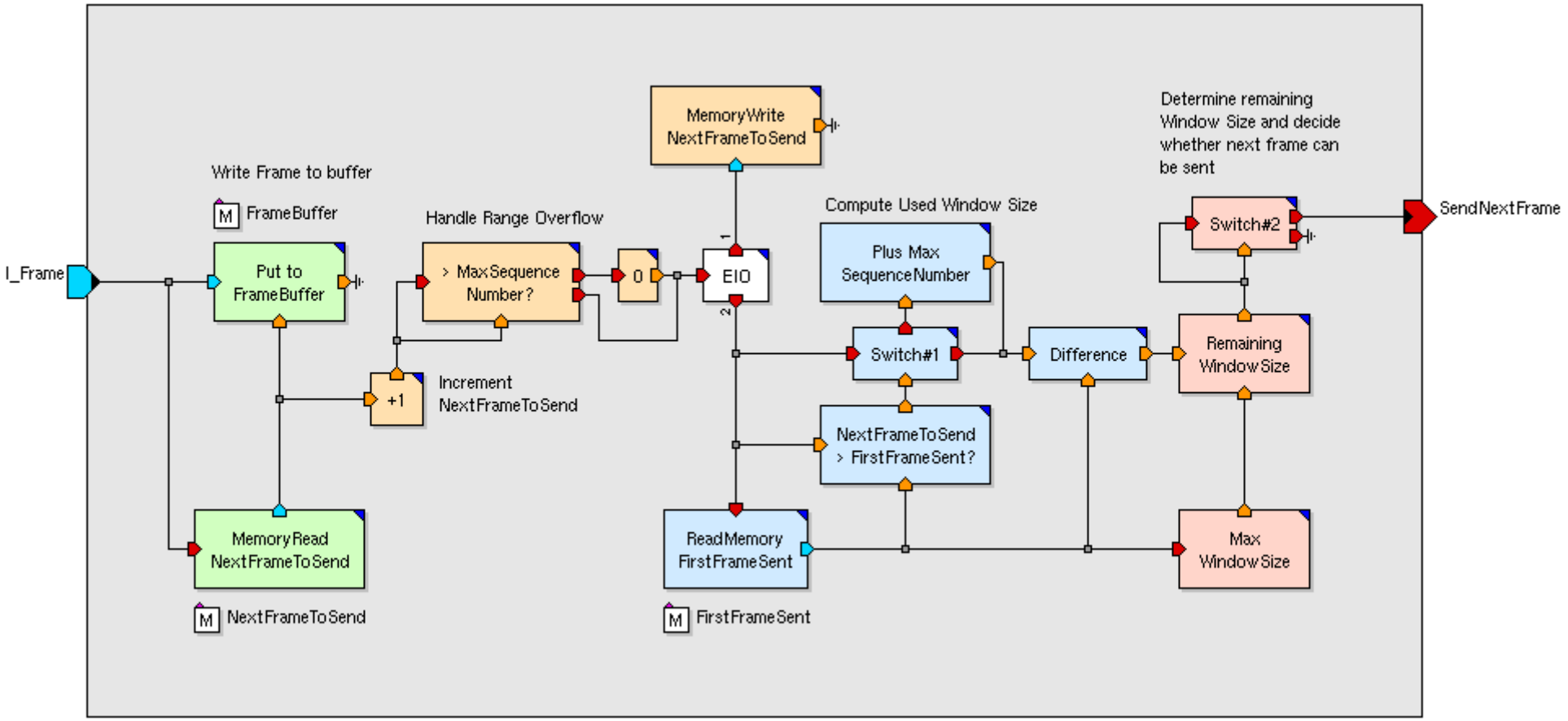
Memories are exported and linked in upper level



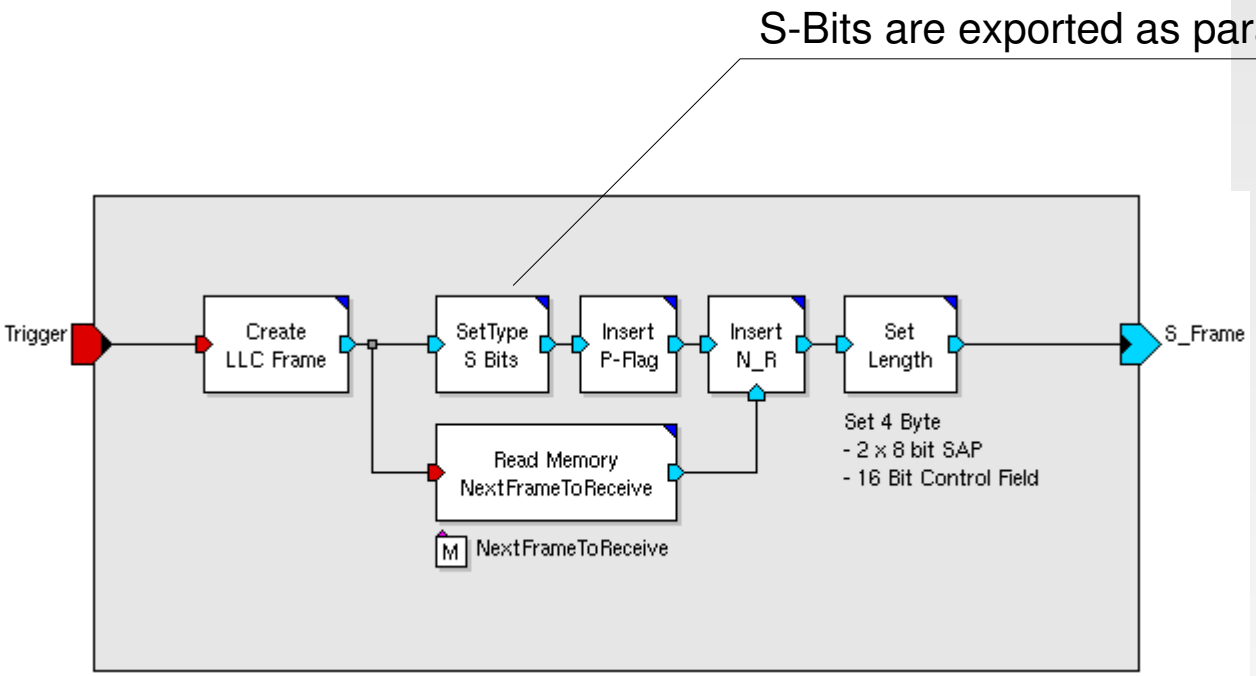
pickup from N_Packet

add a fix size of length

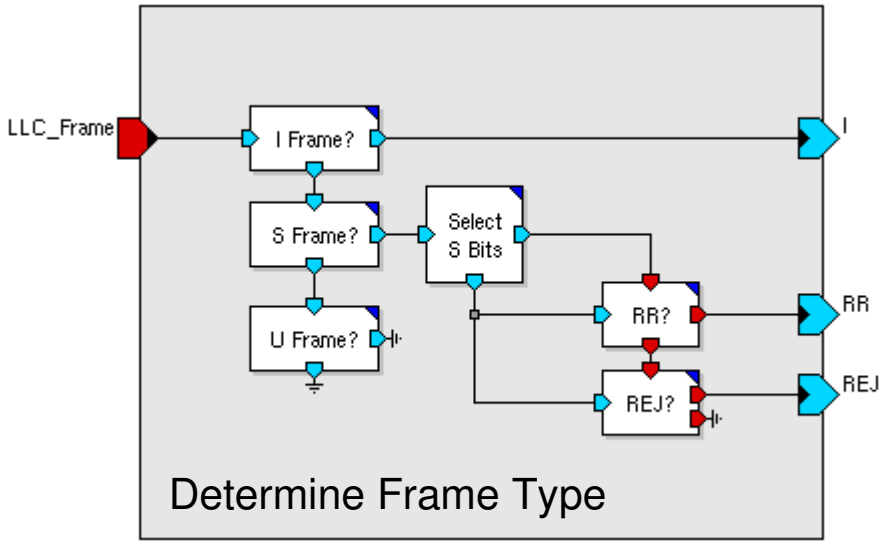
Go-Back-N: LLC Layer Typ2 – Store I-Frame In Buffer

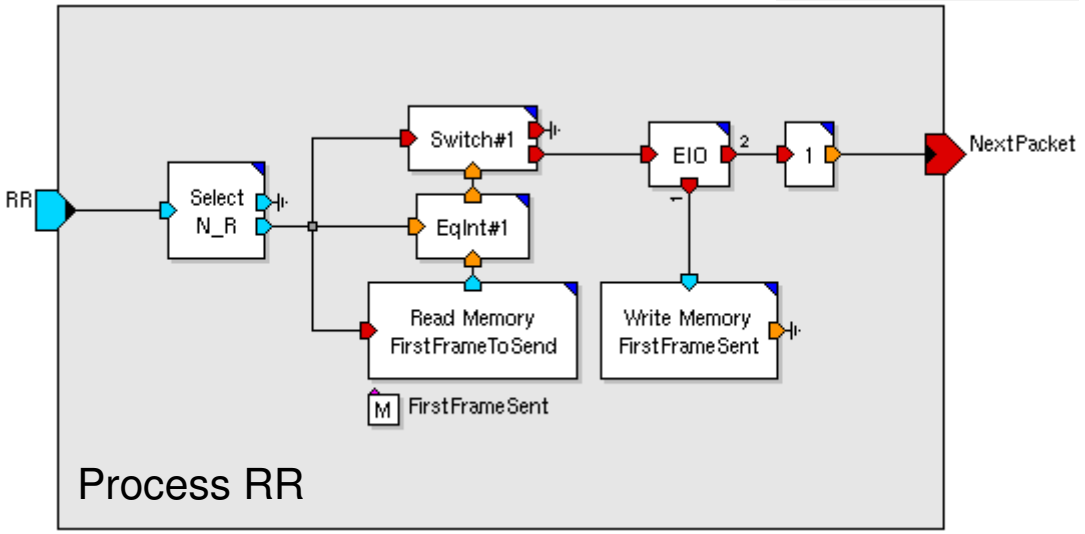


Go-Back-N: LLC Layer Typ2 – Prepare S-Frame

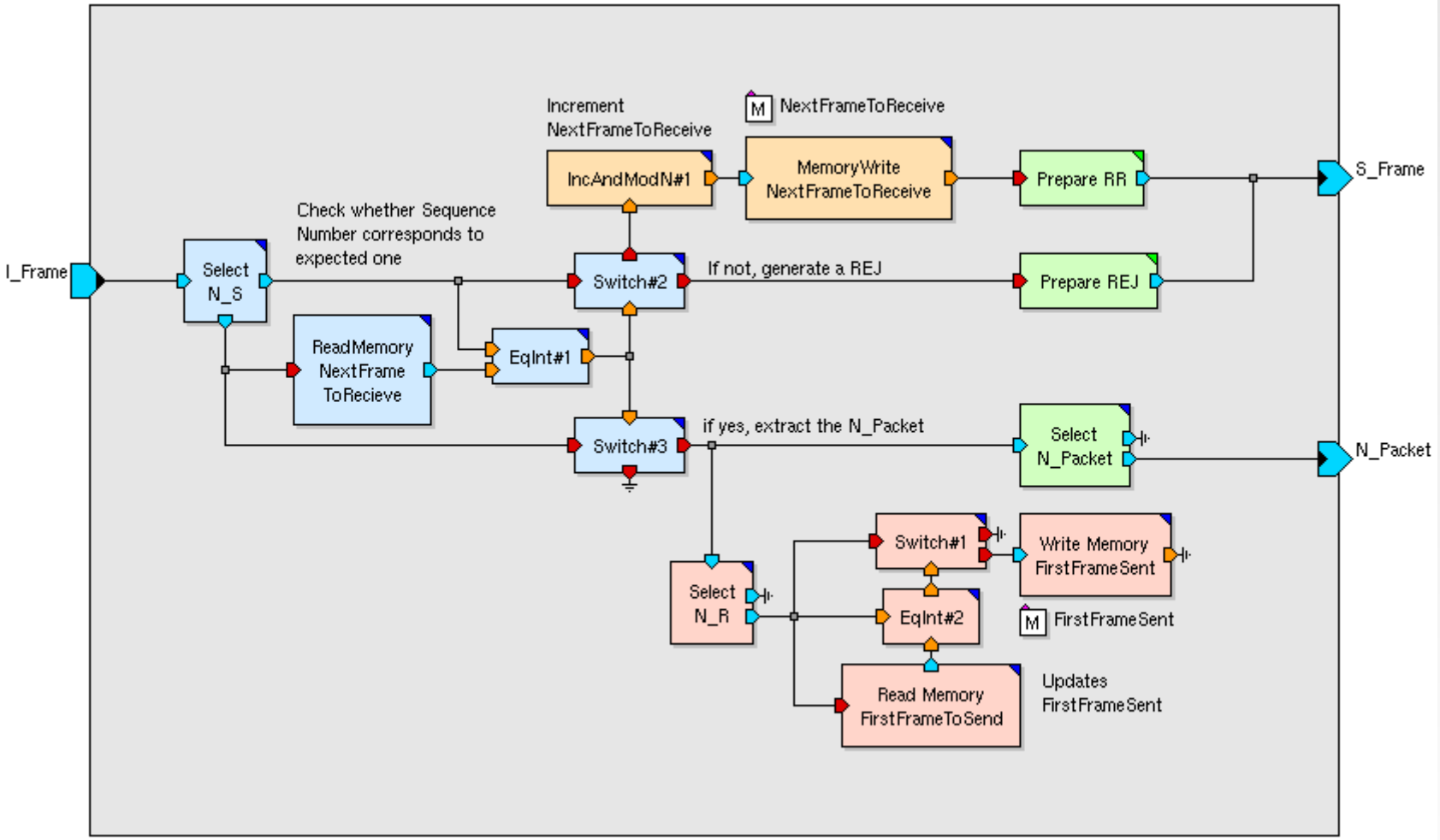


Go-Back-N: LLC Layer – Determine Frame Type / Process RR



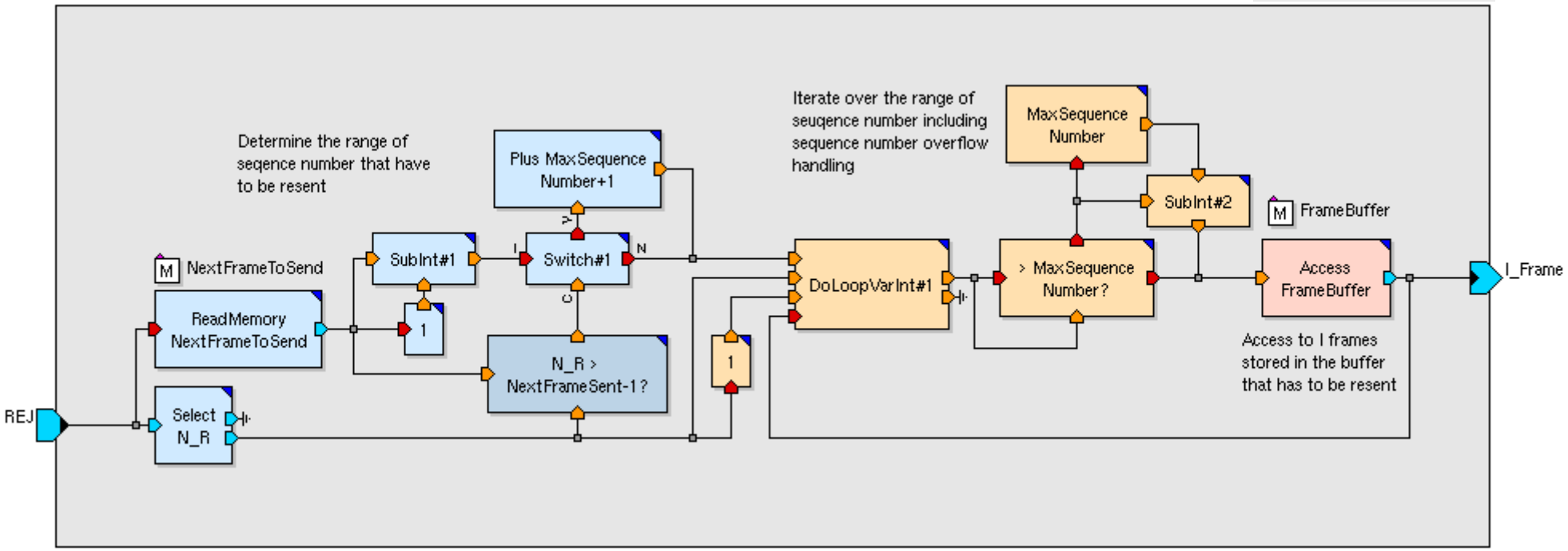


Go-Back-N: LLC Layer Typ2 – Process I

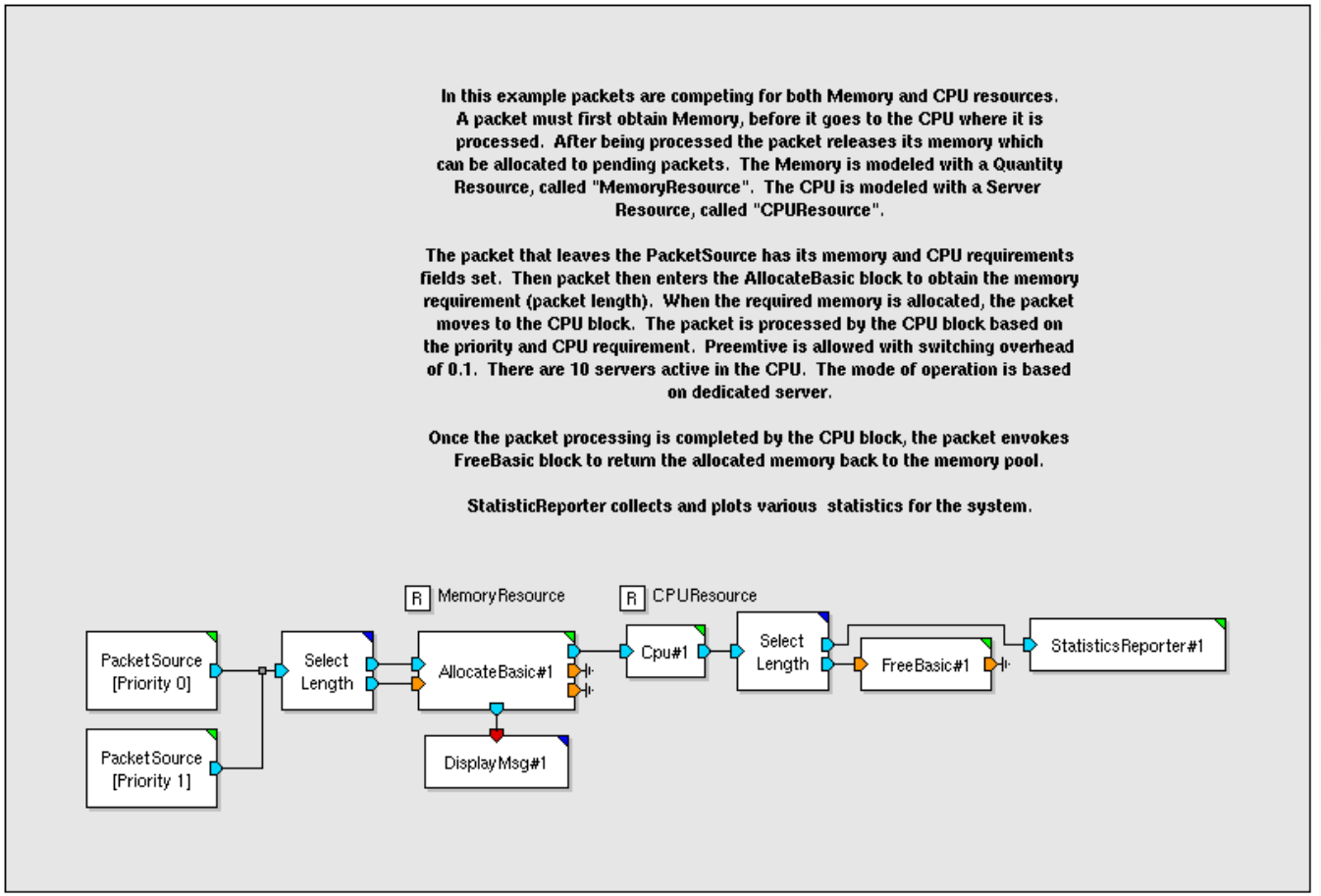


Go-Back-N: LLC Layer Typ2 – Process REJ

Mission Level Design GmbH



Explore CPU Demo



Agenda

- Introduction
- Modeling Concepts
- Application Examples
- Modeling Projects
- Discussion