GSLib Test vectors and test schematics
Table of Contents

1. Introduction

In the equations for processors the variables has the following notation

- xi[j] means : from input i we take only the value nr j, this is the current value
- yi[j] means : from output i we take only the value nr j, this is the old value from previous transition.
- ti[j] means: from input i we take only the value nr j, this is the old value from previous transition.

All the wire and port in the tests are considered to have dimension 1 except where are special indications.

Realization of models and simulation of one system is composed from three base objects: the real system which is seen like a source of data; the model, which is a collection of instructions to generate the comparable dates with the real system; the simulator, which is an algorithm programmed into one language which execute the models instructions.

1.1 Formal definition of base models

Any simulation model could be considered like a real time dynamic system, its components are described by three types of variables: input variables, state variables and output variables. The dynamic behavioral of the model is described by the DEVS (Discrete Event System Specification) object, this is define:

$$DEVS=(X,S,Y,ta,\delta_{int},\delta_{ext},)$$

Where

· X is the "input" collection, Cartesian product of the input variables.

· Y is the "output" collection, Cartesian product of the output variables.

· S is the "state" collection, Cartesian product of the state variables.

· $ta:S\rightarrow R\ (0,\infty)$ is the time advance function, it's value $ta(s),s\in S$ represents the time interval after the system make a transition between state s and state $\delta_{int}(s)$.

· $\delta_{int}(s):S\rightarrow S$ is the internal transition function of the system, this function represent the transition between one and another state without the external influence.

· $\delta_{ext}(s):Q\times I\rightarrow Q$, where $Q=\{(s,e)|s\in S,e\in R,0\leq e\leq ta\}$ is the external transition function of the system, this function represent the transition between one and another state when an external event arrived at time e.

· $\delta:S\rightarrow Y$ is the output function, if the system is in state s then $\delta(s)$ will be the information sent to the output before the systems make a transition given by the transition function (internal or external).

1.2 Network with static structure

Coupling more models is done using a new model, the network model, which can be with static or dynamic structure.

The static structure model is define by

$DN=(D,\{M_i\},\{I_i\},\{Z_{i,j}\},select)$

Where

· D is identifications collection of system components.
· for each $i \in D$ we have
  - M_i is the base model associated with i
  - I_i is the collection of systems influence by i
· for each $j \in I_i$ we have the function $Z_{i,j}$ which is the message transfer function from the system i to the system j.
· select is the selection function: from the components who change their state in the same moment this function choose the component, which influence the state of the coupled model.

Any model defined using the DN structure is a base model.

1.3 Class Hierarchy and functionality

Here there are three different types of classes:

1. Generic classes for the simulator data and functions. Other generic classes inherit those classes.
2. Generic classes for the simulator. User classes inherit those classes.
3. User classes. These classes are the simulator specific application.

Type 1 classes are: CGData, CGRepartition, Queues, Cparser_func, Cparser_func_data, Cparser_func_memory.

Type 2 classes are: CGInput, CGOutput, CGNetworkS, CGProcessor, CGTransition, CGWire.

Type 3 classes are: CInput, COutput, CNetworkS, CProcessor, CTransition, CCoordonator.

1.3.1 Class CGData

This class inherits the CObject class and is the base for the CGInput and CGOutput classes.

This class basic functionality is to hold the data and time for transfer (ports) and internal processor Queues.

Principal functions:

· CGData is the constructor who can take one or two arguments: first argument is the name of the port; which held the data and the second argument is optionally and is the initial value of the data (the second argument is used for quells of the processor if we want only the value).
· GetDim is the function that returns the dimension of the data port.
· GetName is the function that returns the name of the port.
· SetDim is the function that set the dimension of the data port.
· SetWTime is the function who set working time of the data; which is hold by the port.
· SetTime is the function that set the time when the data was created.
· SetParent is the function that set the parent and which held this port.
· GetParent is the function that returns the parent and which held this port.
· SetParentW is the function that set the wire and which held this port into connection with other ports.
· GetParentW is the function that returns the wire and which held this port.
· GetClientExt is the function that returns the external client.
· Set is the function that set the data value.
· Get is the function that gets the data value.
· Reset is the function that resets the data value.
· SetSource is the function that set the pointer to the parent of the data.
· GetSource is the function that returns the pointer to the parent of data.

Functions Set,Get and Reset are virtual functions who are replaced by function in CGInput and CGOutput classes.

1.3.2 Class CGInput

This class inherits the CGData and it must be derived by class CInput to obtain the correct input class.

This class is the input port for all systems: processor and network.

This class also held the pointer to the generator and file if the user desired to have a log file of the generator.

If the dimension of port is multidimensional because this wires are synchronized it can hold a generator for each element of data.

Functions:

· CGInput is the constructor for the input port. It has two arguments: one is the name of the port and the second argument, which is optionally, is the name of the file in which we put the log.
· SetGenerator is the function who set the type of the generator for each element of the data if the dimension is greater than 1.
· Set is the function that set the value of data; this function overrides the function Set defined in CGData class. First version who takes no argument and who generate a value given by the generator. Second version takes one argument and who put the value given by the argument into data; if the port has a generator the argument is added to the generated by the generator. If a file is defined the value is also put in the log file.

1.3.3 Class CGOutput

This class inherits the CGData and it have to be derived by class COutput to obtain the correct input class.

This class is the output port for all systems: processor and network.

Functions:

· CGOutput is the constructor. It has two arguments: first argument is the name of the port and the second argument, which is optionally, is the name of the log file.
· Set is the function that set the value of the port. If a file is defined the value is also put in the log file.

Classes Cparser_func, Cparser_func_data, Cparser_func_memory

These classes are for parser for functions. These classes are called from CGTransition class.

1.3.4 Class CInput

This class inherits CGInput. This class has to be derived to accommodate the specific function of the port.

1.3.5 Class CGWire

This class is the base of the communication of the networks. It has several base functions: send the message from an input (output) port to the connected output (input) port from diverse networks and processors.

Functions:

· CGWire is the constructor, which takes three arguments: first argument is the pointer to the parent port (input/output) from which we take the data and commands; second argument is the type of the wire: input, output, internal; third argument is the pointer to the parent network who has this wire.
· Add is the function that adds a new port to the wire. This port is connected to the parent port.
· receivemsg sends the message thruout the connection. It has two arguments: first argument is the type of message (init, internal, external or output); second argument is the time when the message has arrived.
· select selects the closest end of network or processor.
· sigma returns the minimum of end time or the element with the lowest queue from the connected elements.
· IsHere returns true is the port given in the argument is connected in this wire.
· GetSource returns the source of the connected port.

1.3.6 Class CGNetworkS

This is the base class who contain all the information. This class held the connection between the processors and internal networks. This class is also the front-end of the simulation.

Functions:

· CGNetworkS is the constructor for the class. It has two parameters: first which is optional hold the parent of the network, if this is NULL or isn't given, this is consider that that network is the front end of the simulation; the second argument is the name of the network.

· GetName returns the name of the network.

· SetParent sets the parent of the network, it overwrite the parameter given by the constructor.

· GetParent returns the parent of the network.

· GetPort returns the pointer and the type in the second argument to the port characterized by the name.

· AddPort adds a port to a network and the type of the port (input/output).

· AddNetworkNetwork adds a wire between ports of two internal networks.

· AddProcNetwork adds a wire between ports of one processor and one internal network.

· AddNetworkProc adds a wire between ports of one internal network and one processor.

· AddParentNet adds a wire between a port from the network and a port from an internal network.

· AddParentProc adds a wire between a port from the network and a port from an internal processor.

· AddProcProc adds a wire between a port from one internal processor to an proc from another internal processor.

· AddNetwork adds the affiliation of a network to this network.

· AddProc adds the affiliation of an active processor to this network

· AddProcP adds the affiliation of a passive processor to this network

· GetTypeWire returns the type of the wire (true for unique signal).

· SetTypeWire sets the type of the wire (true for unique signal).

· GetInType returns the type of input data.

· SetInType sets the type of input data (false for data is not a fraction and true the data is divided thru-out the processors and networks).

· SetTypeQueue sets the administration type of queue (false for des-centralized and true for centralized).

· GetTypeQueue returns the administration type of queue.

· SetClockPrecision sets the precision of the clock ± the value when a process is considered out or in.

· PrintData prints the current data on stdout.

· sigma returns the value of the input port (when the process is end).

· SetProc sets the processor, given by his pointer, has executed the transition at a specific time.

· SetNet sets the network, given by his pointer, has executed the transition at a specific time.
· receivemsg is the function of transfer messages: 0-init,1-internal,2-external,3-output.

1.3.7 Class CGProcessor

This class is the base class of the simulator. It contains the active processor.

· AddPort adds a port to the processor if type is true it is input port if it is false is output port.
· GetPort returns the pointer to the port with name port and return in type 0(is not found),1(is input port),2(is output port).
· SetParent sets the network parent of the processor.
· GetParent returns the pointer to the parent of the processor.
· EnableQ enables the queue of the processor.
· DisableQ disables the queue of the processor.
· sigma returns the done time and the dimension of the queue.
· SetClockPrecision sets the increment of clock, it must have the same value as the clock in the coordinator, the simulation clock.
· GetPrevTime returns the value of the time of the last event.
· GetTime returns the current time of simulation.
· GetName returns the name of the processor.
· CGProcessor is the constructor, which has some important arguments: type which said if in a period of working the processor hold the data or it reset the outputs port to zero; generator is the transition function of the processor.
· receivemsg is the interaction function of the processor. The type has 4 value: 0=initialization of the processor, 1=internal transition, 2=external transition and 3=get to the output the actual values.
· SetFirst sets if the first transition could be one internal of it must be one external.
· IsDelayed sets if the data is delayed or the value stored is the correct values.
· SettN sets the new done time.
· SettL sets the last event time.
· receive0 is executed in case of transition 0.
· receive1 is executed in case of transition 1.
· receive2 is executed in case of transition 2.
· receive3 is executed in case of transition 3.
· set_extfunc sets the external function, so the function for transition 2.
· set_intfunc sets the internal function, so the function for transition 1.

1.3.8 Class CGTransition

This is the class who held the real functionality of a processor. This class must be derivate in all cases to achieve the real functionality of the processor.

· SetRepartitionT sets the repartition table for an input repartition.

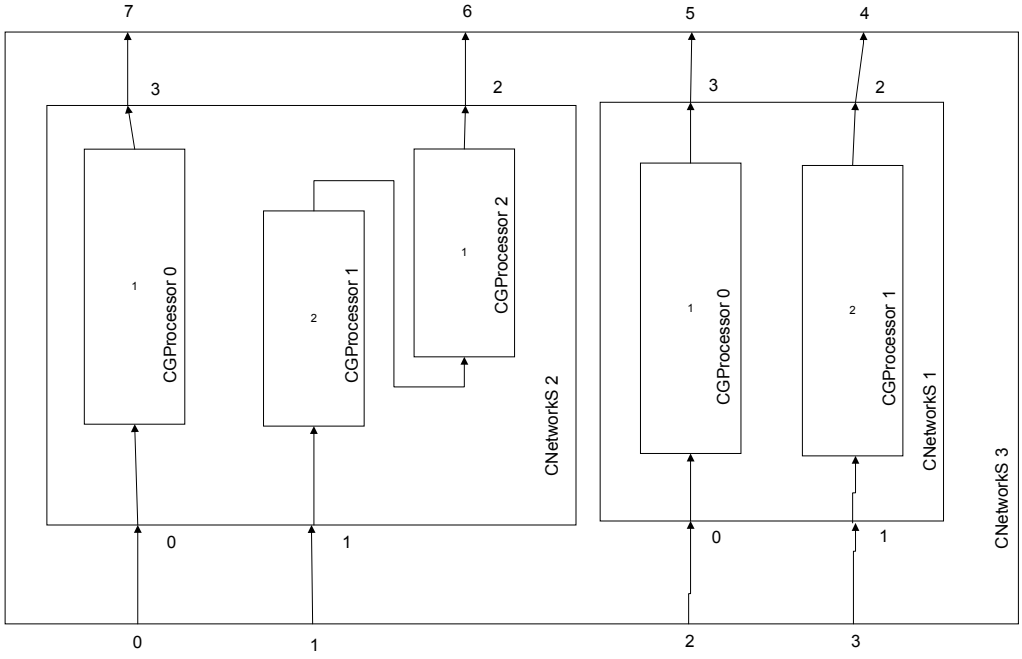
· GetIndexT returns the real index in the table of the input repartition for the argument.
· Set sets the seed for the random number generator.
· SetClk sets the clock of the simulation for a waiting process.
· SetParent sets the parent processor of the transition.
· CGTransition is the constructor, which creates the actual random repartition.
· proc_deltaext is executed when an external transition comes (it is virtual because real application have to derivate it).
· proc_delatint is executed when an internal transition comes (it is virtual because real application have to derivate it).
· proc_lambda is executed after each transition (it is virtual and some application must derivate it if they want to filter some output ports).
· assign_memory assigns the variable memory for the parser for all function.
· set_extfunc sets the parser for the external function for the specific output port.
· set_intfunc sets the parser for the internal function for the specific output port.
· is_parallel sets the parser in parallel mode, that mean all the wire from an output port with dimension great than 1 is consider to have the same internal and external function.

2. Normal Functionality

2.1 Testing the processors and networks (test1_1 and test1_2)

The equations for all processors are

- x0[0]+y1[0] in case of external function for port 1
- y1[0]+0.5 in case of internal function for port 1



In first network processors have the following delays: 1,2,1.
In second network processors have the following delays: 1,2.

All processors have queue, non-interruptible and reset for the data.

We have the following behavioral for the master network (3) for first test vectors

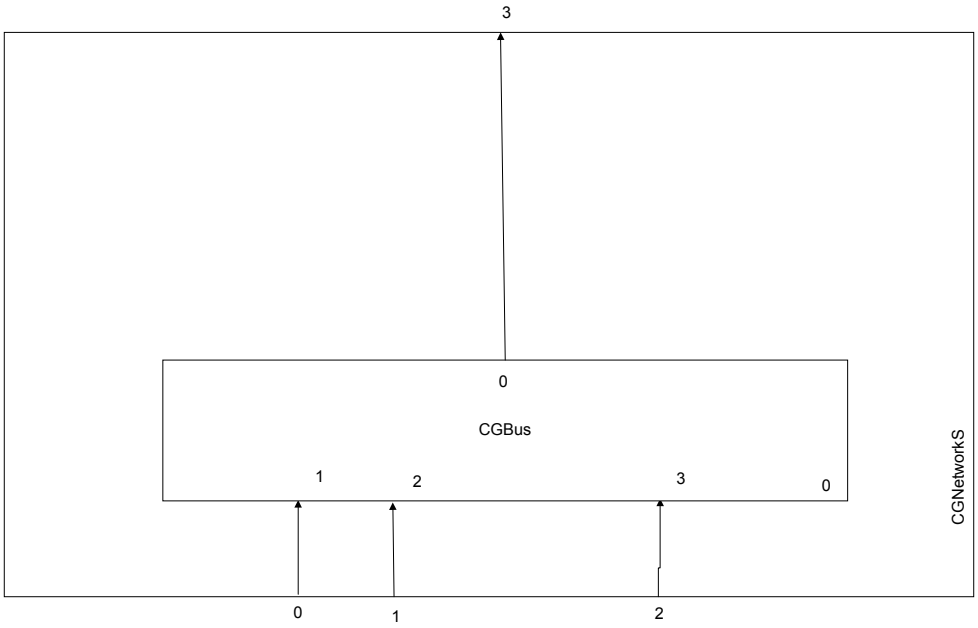
port\time	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	1	0	0	0	0	0
3	0	1	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	1.0	0	1.5	0	2.0	0	3.0	0	3.5	0
5	0	0	1.0	1.5	2.0	2.5	3.0	3.5	4.5	5.0	5.5	6.0	6.5
6’	0	0	0	1.0	0	1.5	0	2.0	0	2.5	0	3.0	0
6	0	0	0	0	1.0	1.5	3.0	3.5	5.5	6.0	8.5	9.0	12.0
7	0	0	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0

We have the following behavioral for the master network (3) for second test vectors

port\time	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	1	1	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	0	0
2	0	1	1	0	0	1	1	0	0	0	0	0	0
3	0	1	0	0	1	1	0	0	0	1	0	0	0
4	0	0	0	1	0	1.5	0	2.5	0	3.5	0	4.5	0
5	0	0	1	2	2.5	3.0	4.0	5.0	5.5	6.0	6.5	7.0	7.5
6’	0	0	0	1	0	1.5	0	2.5	0	3.0	0	3.5	0
6	0	0	0	0	1	1.5	3.0	3.5	6.0	6.5	9.5	10.0	13.5
7	0	0	1.0	1.5	2.0	3.0	4.0	5.0	5.5	6.0	6.5	7.0	7.5

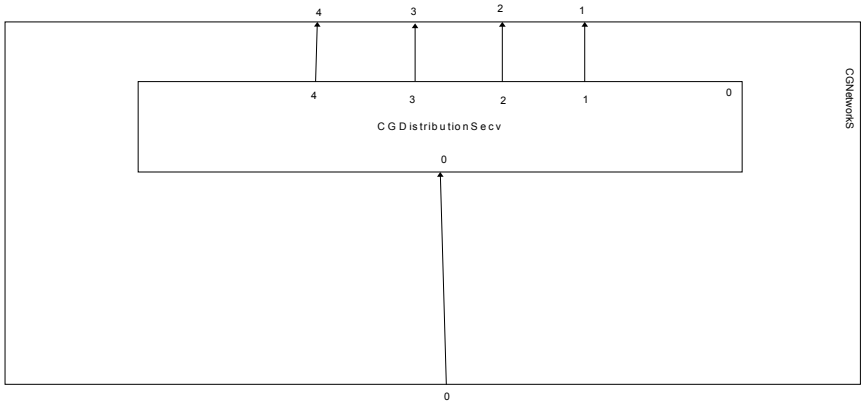
2.2 Testing Bus (testBuses)

The outputs of Bus (0 for CGBus and 3) from network have dimension 3.



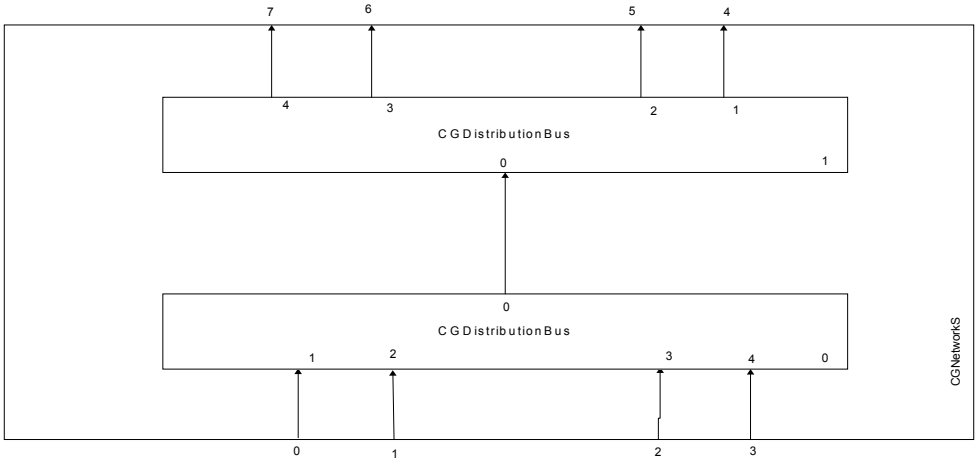
port\time	0	1	2	3	4	5	6	7	8	9
0	1	0	1	0	1	0	1	0	1	0
1	1	0	1	0	1	0	1	0	1	0
2	1	0	1	0	1	0	1	0	1	0
3	1,1,1	0,0,0	1,1,1	0,0,0	1,1,1	0,0,0	1,1,1	0,0,0	1,1,1	0,0,0

2.3 Test Distribution Sequential (testDistributions)



port\time	0	1	2	3	4	5	6	7	8	9
0	1	0	1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0	0	1	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0

2.4 Testing Distribution Buses (MUX and DMUX) (testMDM)



port/time	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1

2.5 Processor with multiple entry and multiple functions for output (testmprocs)

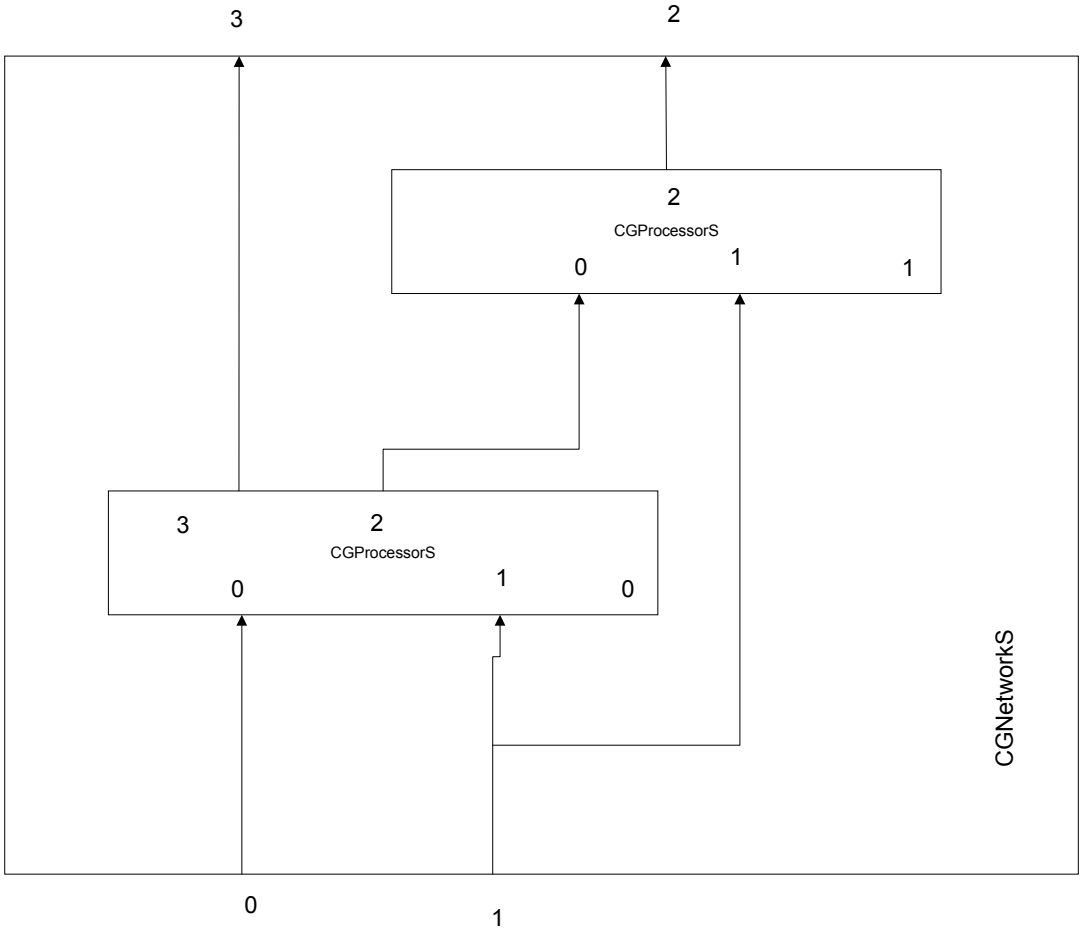
The equations for processor 0 are

- x1[0]+y2[0] in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2
- y3[0]+x0[0] in case of external function and port 3
- y3[0]+0.5 in case of internal function and port 3
- These functions are synchronized

The equations for processor 1 are

- x0[0]+x1[0]+1 in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2

All processor have 1 unity delay



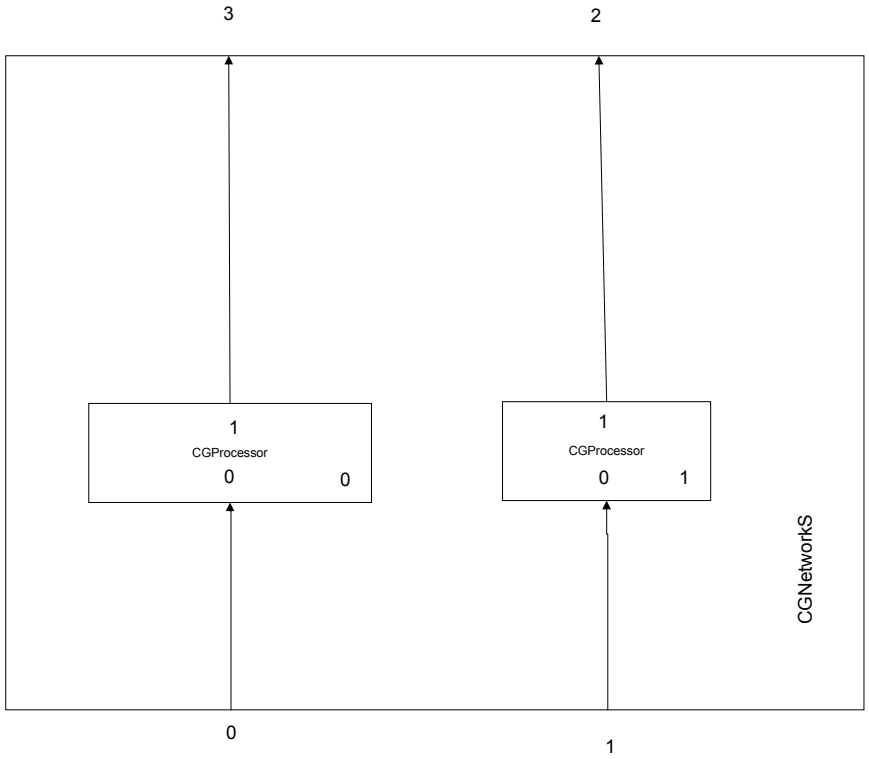
port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5
2	0	0	2.0	2.0	2.5	4.0	4.0	4.5	5.0
3	0	0	1.0	1.5	2.0	2.0	2.5	3.0	3.5

2.6 Copy statically a processor (testcps)

The equations for all processors are

- x0[0]+y1[0] in case of external function for port 1
- y1[0]+0.5 in case of internal function for port 1

All processor have the delay of 1.



The processor 1 is a copy of processor 0. The copy is done before sending the message 0 to the network.

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0
2	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5
3	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5

2.7 Test de compatibility of network with processors with multiple entries (testmprocsn)

The equations for processor 0 are

- x1[0]+y2[0] in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2
- y3[0]+x0[0] in case of external function and port 3
- y3[0]+0.5 in case of internal function and port 3
- These functions are synchronized

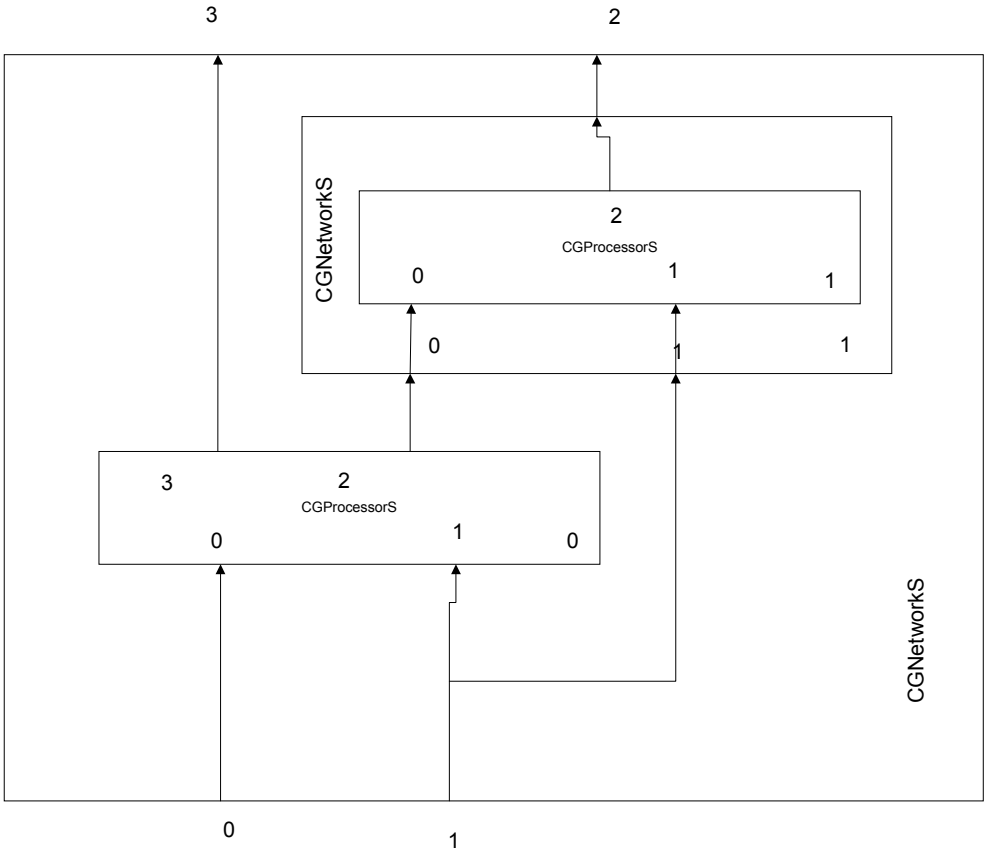
The equations for processor 1 are

- x0[0]+x1[0]+1 in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2

All processor have 1 unity delay

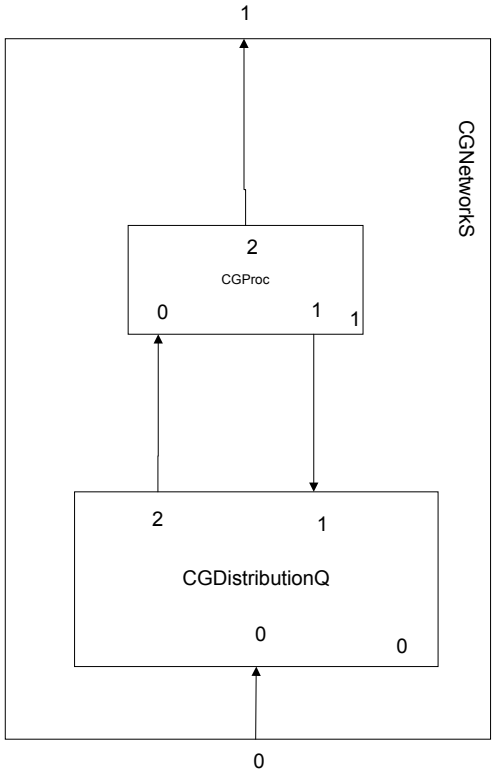
The reference for this example is example 1.5

All network have wire type set to false.



port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5
2	0	0	2.0	2.0	2.5	4.0	4.0	4.5	5.0
3	0	0	1.0	1.5	2.0	2.0	2.5	3.0	3.5

2.8 Test for Distribution Queue (testDQL)

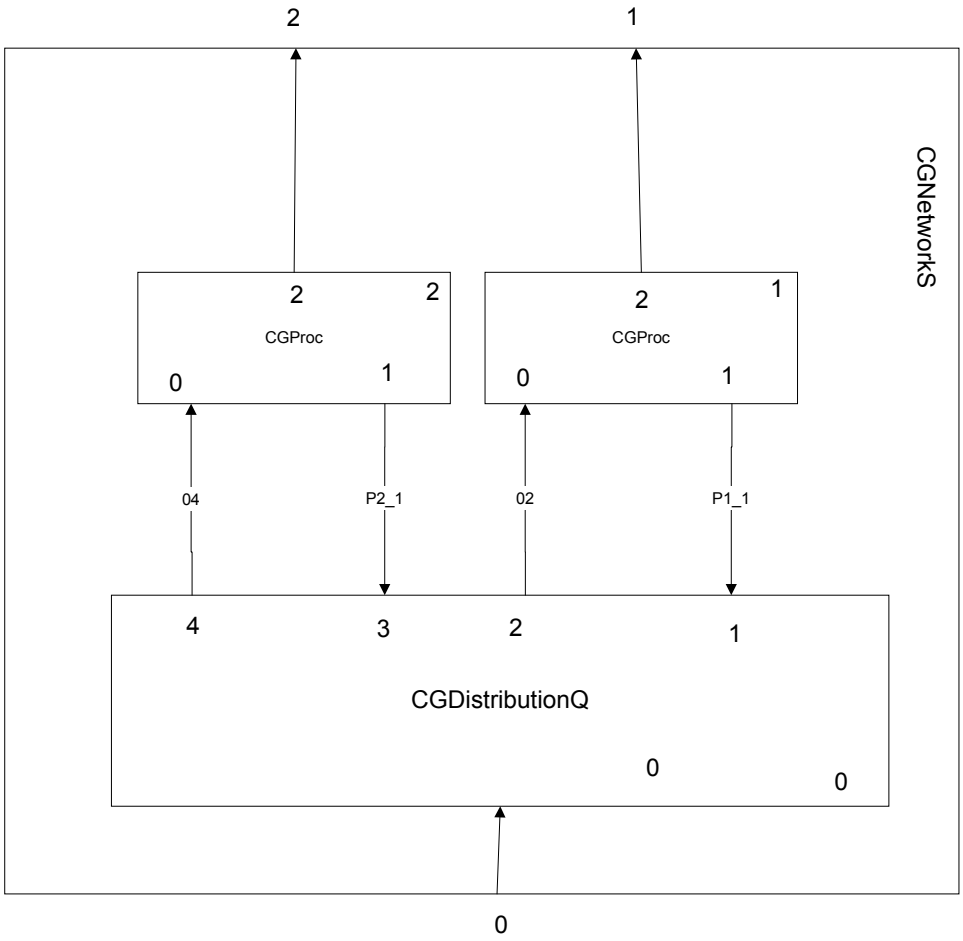


All CGProc have delay=3 and have disabled the queue.
All processors have the special transition class CTransition.

t	0	1	2	3	4	5	6	7	8	9	10	11
1.1	1	-	-	-	1	-	-	1	-	-	1	-
1.2	0	-	-	-	1	-	-	2	-	-	3	-
0.2	-	1	-	-	1	-	-	1	-	-	-	-
0.0	-	1	1	1	-	-	-	-	-	-	-	-
Q	0	0	1	1;1	1	1	1	0	0	0	0	0

2.9 Test Distribution Queue with 2 processors (testDQL_2)

This is a generalization of previous test (**testDQL**) where we have only one processor. All CGProc have delay=3 and have disabled the queue.

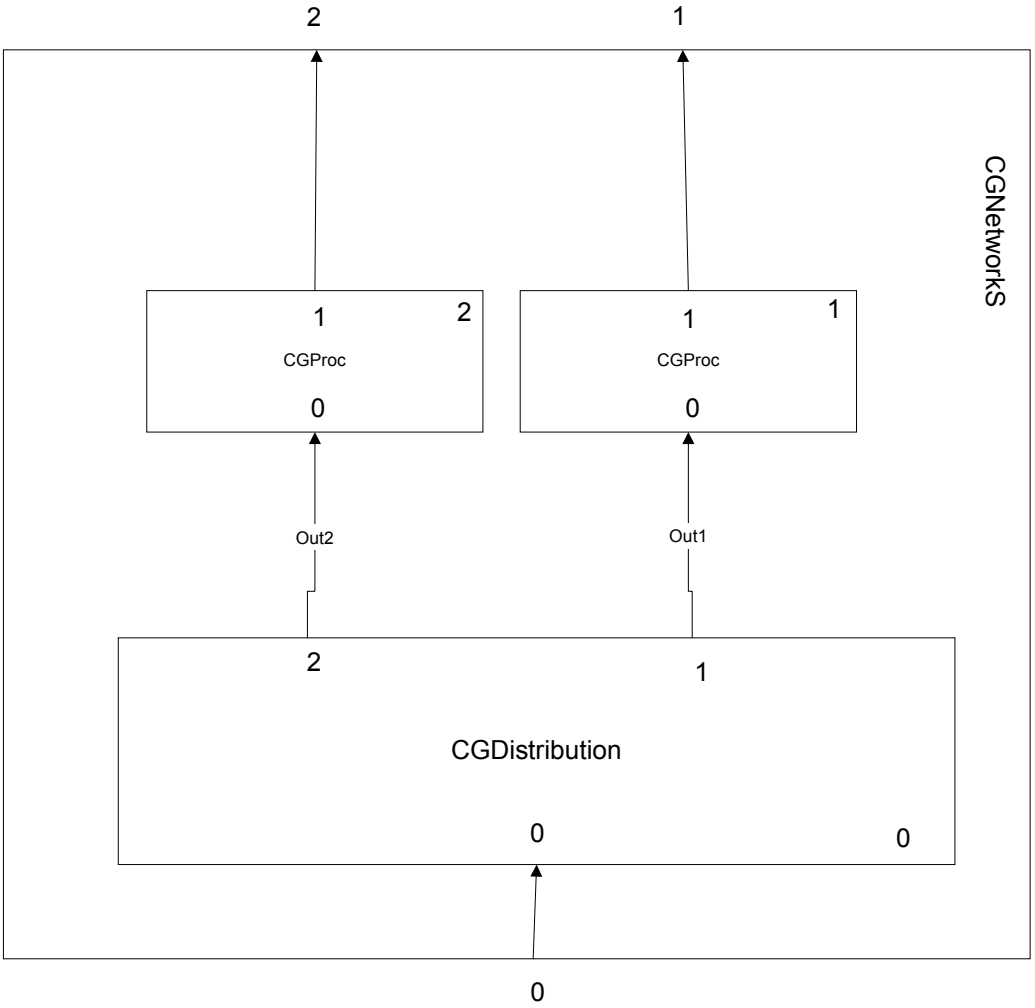


t	0	1	2	3	4	5	6	7
IN	-	1	1	1	-	-	-	-
P1_1	1	-	-	-	1	-	-	1
P2_1	1	-	-	-	-	1	-	-
02	-	1	-	-	1	-	-	-
04	-	-	1	-	-	-	-	-
1	0	-	-	-	1	-	-	2
2	0	-	-	-	-	1	-	-
Q	0	0	0	1	0	0	0	0

2.10 Test Distribution Sequential with two machines (testDistributions_2)

This is a generalization of test **testDistributions**.

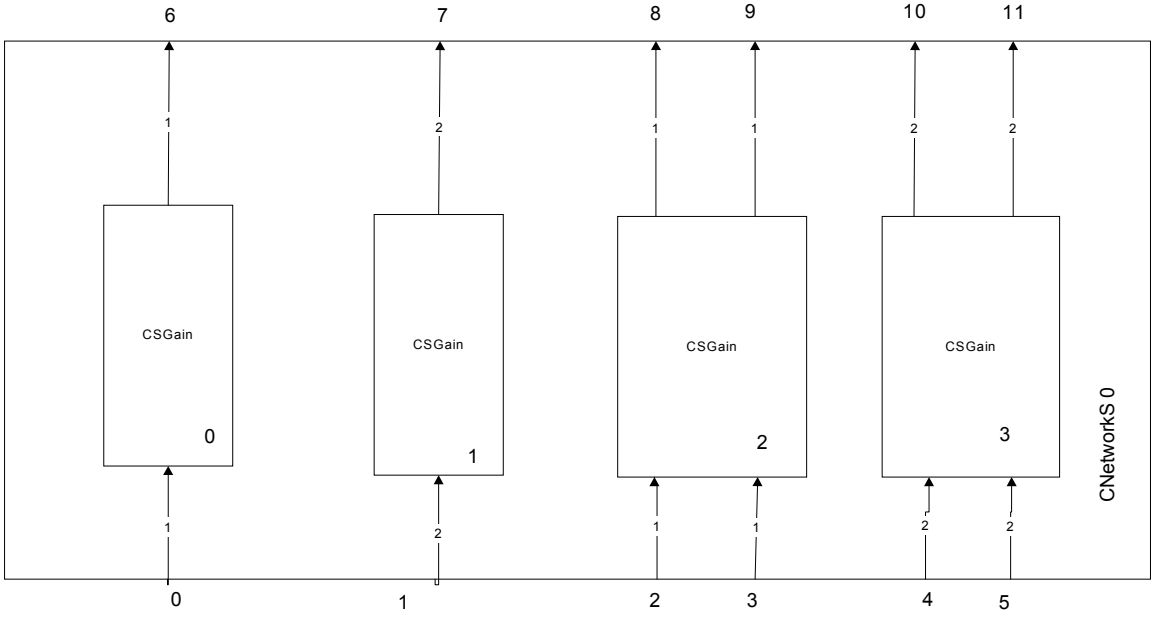
All processors have "x0[0]+y1[0]" for external and "y1[0]+0.5" for internal and delay 1.0.



t	0	1	2	3	4	5	6	7	8	9
IN	1	-	1	-	1	-	1	-	1	-
Out1	1	-	-	-	1	-	-	-	1	-
Out2	-	-	1	-	-	-	1	-	-	-
1	0	1	1.5	2	2.5	3.5	4	4.5	5	6
2	0	0	0	1	1.5	2	2.5	3.5	4	4.5

2.11 TestGain without delay (testGain_ND)

All processor have 0.5 delay and 2.0 multiplication factor.

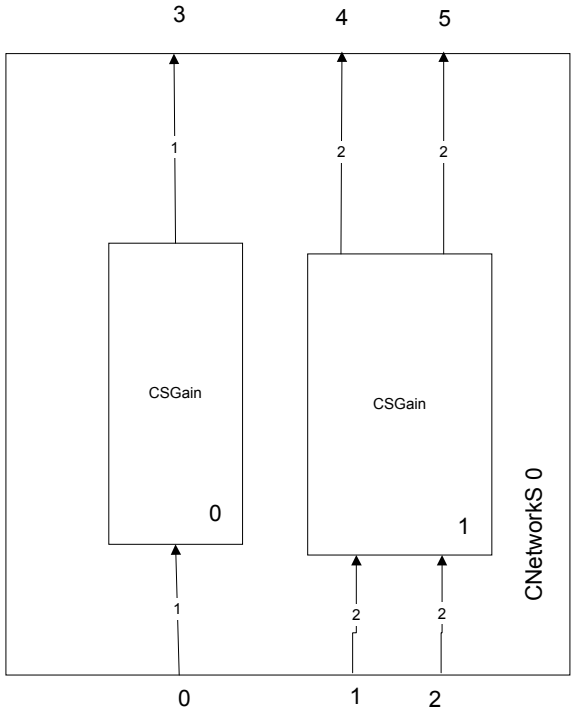


t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
0	0	0.5	-	-	-	2.5	-	-	4	4.5
1	0 0	0.5 1	-	-	-	2.5 3.0	-	-	4 4.5	4.5 5.0
2	0	0.5	-	-	-	2.5	-	-	4	4.5
3	0	1	-	-	-	3.0	-	-	5	5.0
4	0 0	0.5 1	-	-	-	2.5 3.0	-	-	4 4.5	4.5 5.0
5	0 0	1 1.5	-	-	-	3.0 3.5	-	-	4.5 5	5.0 5.5
6	0	1	-	-	-	5	-	-	8	9
7	0 0	1 2	-	-	-	5 6	-	-	8 9	9 10
8	0	1	-	-	-	5	-	-	8	9
9	0	2	-	-	-	6	-	-	9	10
10	0 0	1 2	-	-	-	5 6	-	-	8 9	9 10
11	0 0	2 3	-	-	-	6 7	-	-	9 10	10 11

t	5	5.5	6	6.5	7	7.5	8	8,.5	9
0	-	-	-	6.5	-	-	-	8	-
1	-	-	-	6.5 7	-	-	-	8 8.5	-
2	-	-	-	6.5	-	-	-	8	-
3	-	-	-	7	-	-	-	-	-
4	-	-	-	6.5 7	-	-	-	8 8.5	-
5	-	-	-	7 7.5	-	-	-	-	-
6	-	-	-	13	-	-	-	16	-
7	-	-	-	13 14	-	-	-	16 17	-
8	-	-	-	13	-	-	-	-	-
9	-	-	-	14		-	-	-	-
10	-	-	-	13 14	-	-	-	-	-
11	-	-	-	14 15	-	-	-	-	-

2.12 Test Gain with delay (testGain)

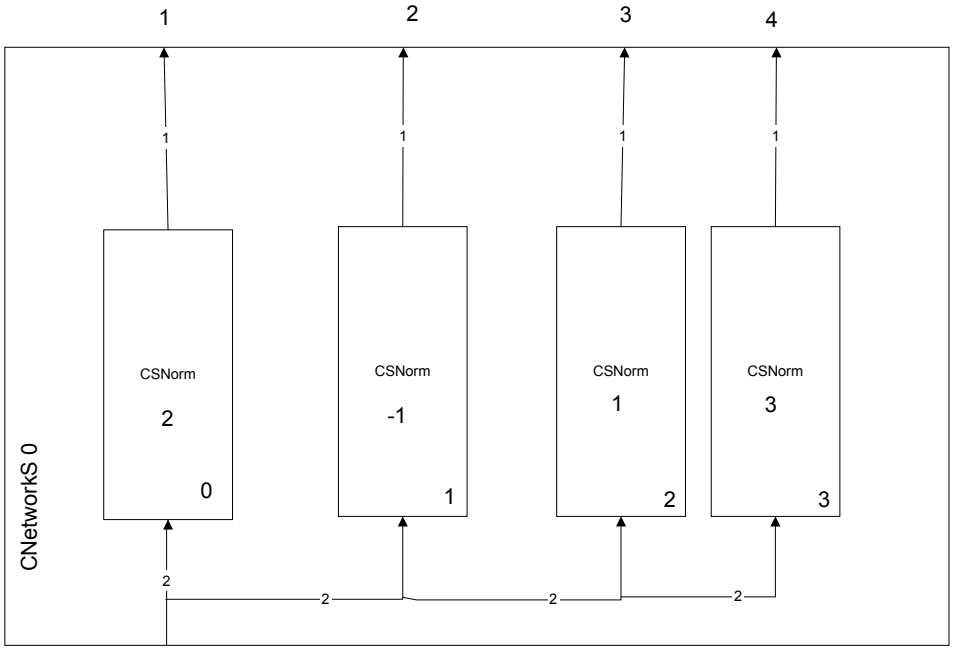
All processor have 0.5 delay and 2.0 multiplication factor.



t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6
0	0	0.5	-	-	2	2.5	-	-	4	-	-	5.5	-
1	0 0	0.5 1.0	-	-	2 2.5	2.5 3	-	-	4 4.5	-	-	5.5 6	-
2	0 0	1.0 1.5	-	-	2.5 3	3 3.5	-	-	4.5 5	-	-	-	-
3	0	-	1	-	-	4	5	-	-	8	-	-	11
4	0 0	-	1 2	-	-	4 5	5 6	-	-	8 9	-	-	-
5	0 0	-	2 3	-	-	5 6	6 7	-	-	9 10	-	-	-

2.13 Test Norm (testNorm)

Norm p=(sum($|x_i|^p$))$^{1/p}$ for p=1,2,… infinite (-1)=max$|x_i|$
All processor have no delay.

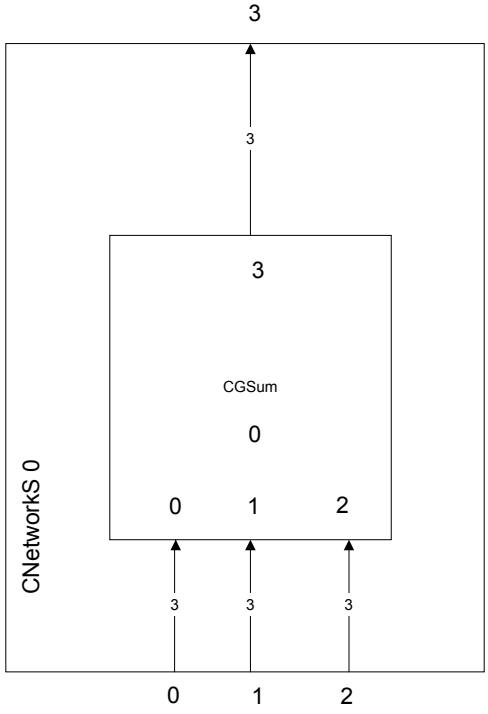


t	0.0	0.5	1.0
0	0 0	-4.0 5.0	-
1	0	6.4	-
2	0	5	-
3	0	9	-
4	0	5.73	-

2.14 Test Delay (testDelay)

2.15 Test Sumator without delay (testGSum)

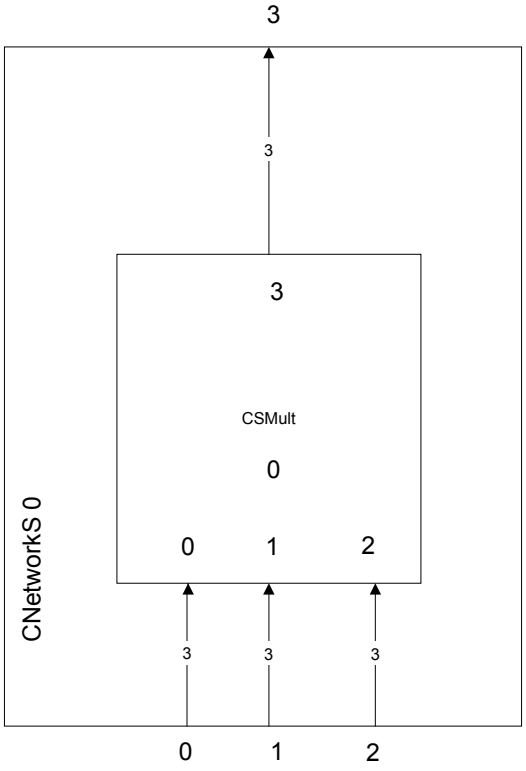
This is the vectors sum without delay.
It has unlimited inputs and dimension of wires but it has only one output.



t	0	0.5	1	1.5	2.0	2.5	3	3.5
0	0 0 0	1.5 2 3	-	0 0 1	-	-	-	0 0 1
1	0 0 0	1 0 0	-	-	-	0 1 0	-	0 1 0
2	0 0 0	0 -2 -3.5	-	0 1 0	-	0 -1 0	-	-
3	0 0 0	2.5 0 -0.5	-	0 1 1	-	0 0 0	-	0 1 1

2.16 Test Multiplication without delay (testSMult)

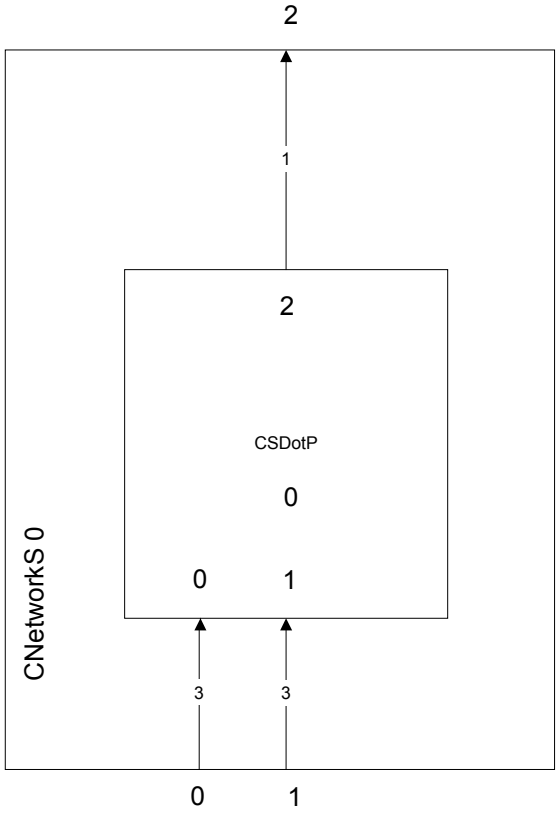
It is the component multiplication of vectors (a,b,c)*(d,e,f)=(a*d,b*e,c*f).
It has unlimited inputs and dimension but only out output.



t	0	0.5	1	1.5	2	2.5	3
0	0 0 0	2 3 0.5	-	2 0 3	-	1 2 3	1 2 3
1	0 0 0	1 1 1	-	1 4 5	1 2 3	-	1 2 3
2	0 0 0	-1 -1 -1	-	1 -1 3	1 2 3	1 2 3	-
3	0 0 0	-2 -3 -0.5	-	2 0 45	0 0 0	0 0 0	0 0 0

2.17 Test Dot Product without delay (testDotP)

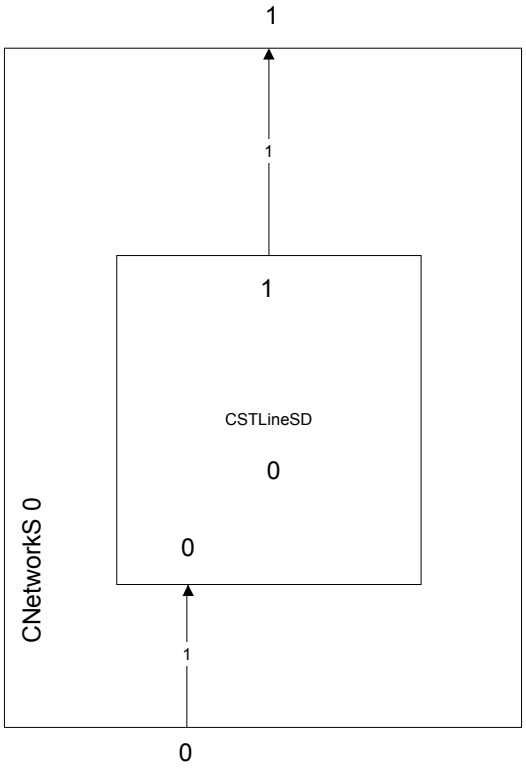
It is the dot product or scalar product.
It has two multidimensional inputs and one dimension output.



t	0	0.5	1	1.5	2	2.5
0	0 0 0	1 0 0	-	1 1 1	-	1 0 0
1	0 0 0	2 1 1	-	2 3 4	-	0 1 0
2	0	2	-	9	-	0

2.18 Test Transmission Line with standard delay (testCSTLineSD)

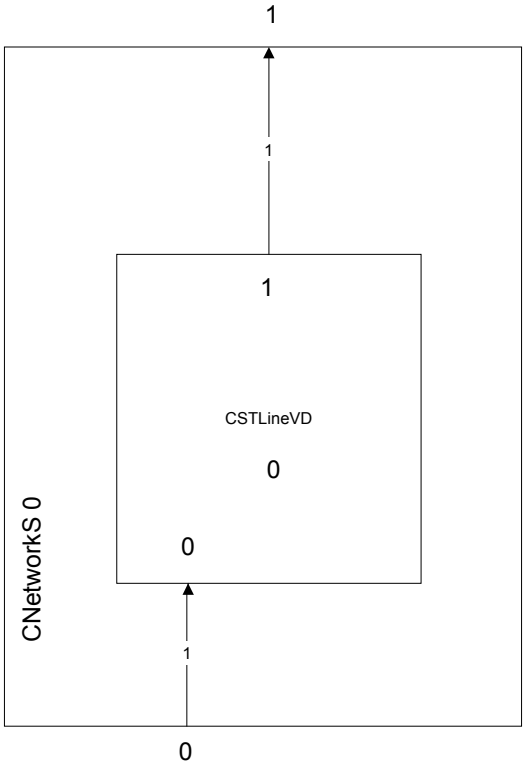
The transition line has a standard delay and the one input port and one output port. The input and output port are multidimensional.



t	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8
0	0	1	-	-	2	-	3	-	-	4	5	-	6	7	8	-	-
1	0	-	-	1	-	-	2	-	3	-	-	4	5	-	6	7	8

2.19 Test Transmission Line with variable delay and noise (testCSTLineVD)

The transition line has a standard delay and the one input port and one output port.
The input and output port are multidimensional.
The delay is 1.5 the noise is 0.0 and clock is 0.5 or 1.0



t	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7
0	0	1	-	-	2	-	3	-	-	4	5	-	-	-	-
1	0	-	-	-	1	-	-	2	-	3	-	-	4	5	-

3. Dynamic functionality

3.1 Delete a port and processors (testdelp)

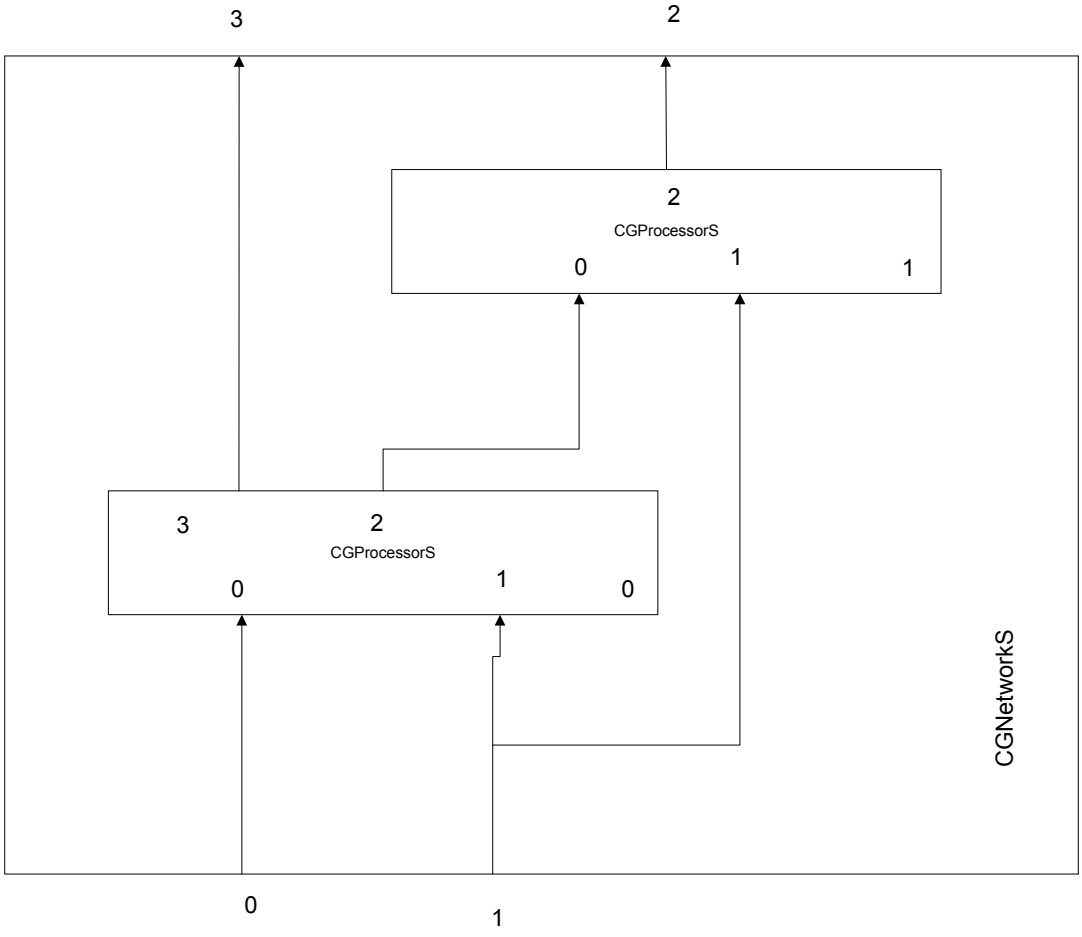
The equations for processor 0 are

- x1[0]+y2[0] in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2
- y3[0]+x0[0] in case of external function and port 3
- y3[0]+0.5 in case of internal function and port 3
- These functions are synchronized

The equations for processor 1 are

- x0[0]+x1[0]+1 in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2

All processor have 1 unity delay



3.1.1 Delete a port from a wire with multiple entry

We will delete the port 1 from processor 0. The connection from input 1 to processor 1 must be preserved and in external equations for processor 0 the variable x1 and t1 must disappear.

We consider the reference the example from 1.5.

The port will be deleted at time moment 4 using *procd->DelPort(pdel1);*

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	2.0	2.5	3.0	3.5	4.0
2	0	0	2	2	2.5	4.0	3.5	4.0	4.5
3	0	0	1.0	1.5	2.0	2.5	3.0	3.5	4.0

3.1.2 Delete an internal output port with only one input port in a wire

We will delete the port 2 form processor 0. The internal connection must be deleted, and the port 0 from processor 1 with variables x0 and t0.

We consider the reference the example from 1.5.

The port will be deleted at time moment 4 using *procd->DelPort(pdel2);*

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	-	-	-	-	-
2	0	0	2	2	2.5	2.0	2.5	3.0	3.5
3	0	0	1.0	1.5	2.0	2.0	2.5	3.0	3.5

3.1.3 Delete an processor output with only one output correspondent port in the wire (output port of a network)

We will delete the port 3 from processor 0. The output connection must be deleted, and the port 3 from network must be deleted.

We consider the reference the example from 1.5.

The port will be deleted at time moment 4 using *procd->DelPort(pdel3);*

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5
2	0	0	2.0	2.0	2.5	4.0	4.0	4.5	5.0
3	0	0	1.0	1.5	-	-	-	-	-

3.1.4 Delete a processor test 1

We delete the processor 0. The internal connection must be deleted and the network port 4 also must be deleted, but the processor 1 must be like in example 2.1.2.

The processor will be deleted at time moment 4 using one of the following combinations:

- procd->kill(); delete procd;
- net->DelProc(procd);

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	-	-	-	-	-
2	0	0	2.0	2.0	2.5	2.0	2.5	3.0	3.5
3	0	0	1.0	1.5	-	-	-	-	-

3.1.5 Delete a processor test 2

We delete the processor 1. The internal connection must be deleted and the network port 2 must be deleted.

The processor will be deleted at time moment 4 using one of the following combinations:

- procd->kill(); delete procd;
- net->DelProc(procd);

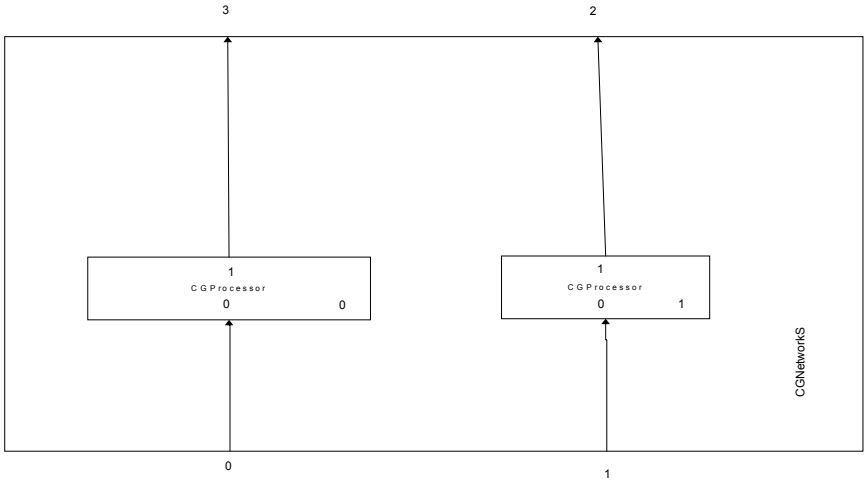
port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	-	-	-	-	-
2	0	0	2	2	-	-	-	-	-
3	0	0	1.0	1.5	2.0	2.0	2.5	3.0	3.5

3.2 Add a processor

The equations for all processors are

- x0[0]+y1[0] in case of external function for port 1
- y1[0]+0.5 in case of internal function for port 1

All processor have the delay of 1.



The reference for this example is example 1.6.

2.2.1 Add without copy a processor (testapd)

The processor 1 is a copy of processor 0 without used copy and adds proc at time 4.

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	1	0	0	0	0
1	-	-	-	-	1	0	0	0	0
2	-	-	-	-	0	1.0	1.5	2.0	2.5
3	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5

2.2.2 Add with copy a processor (testacpd)

The processor 1 is a copy of processor 0 with using copy and adds proc at time 4.

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	1	0	0	0	0
1	-	-	-	-	1	0	0	0	0
2	-	-	-	-	2.0	3.0	3.5	4.0	4.5
3	0	0	1.0	1.5	2.0	3.0	3.5	4.0	4.5

3.3 Delete ports and networks

The equations for processor 0 are

- x1[0]+y2[0] in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2
- y3[0]+x0[0] in case of external function and port 3
- y3[0]+0.5 in case of internal function and port 3
- These functions are synchronized

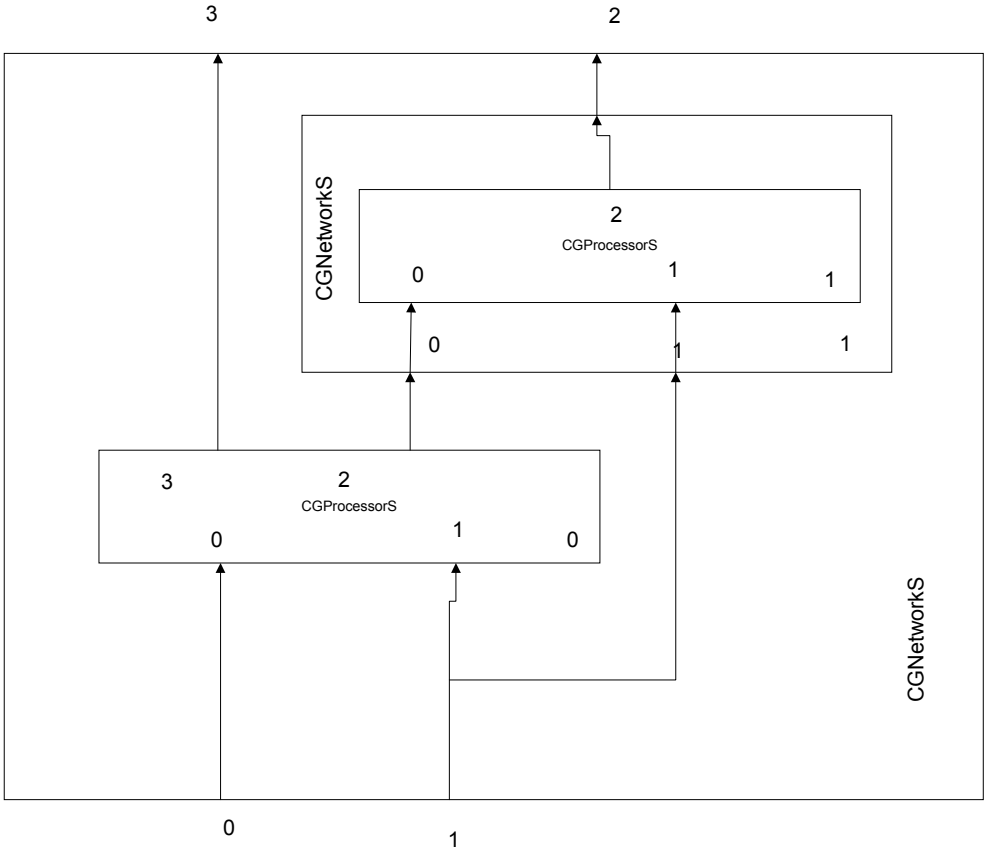
The equations for processor 1 are

- x0[0]+x1[0]+1 in case of external function and port 2
- y2[0]+0.5 in case of internal function and port 2

All processor have 1 unity delay

The reference for this example is example 1.5

All network have wire type set to false.



2.3.1 Delete an internal output port with only one input port in a wire (testmdelpn)

We will delete the port 2 form processor 0. The internal connection must be deleted, and the port 0 from processor 1 with variables x0 and t0.

We consider the reference the example from 1.5.

The port will be deleted at time moment 4 using *procd->DelPort(pdel2);*

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	-	-	-	-	-
2	0	0	2	2	2.5	2.0	2.5	3.0	3.5
3	0	0	1.0	1.5	2.0	2.0	2.5	3.0	3.5

2.3.2 Delete a network (testmdelpn)

We delete the processor 1. The internal connection must be deleted and the network port 2 also must be deleted.

The processor will be deleted at time moment 4 using one of the following combinations:

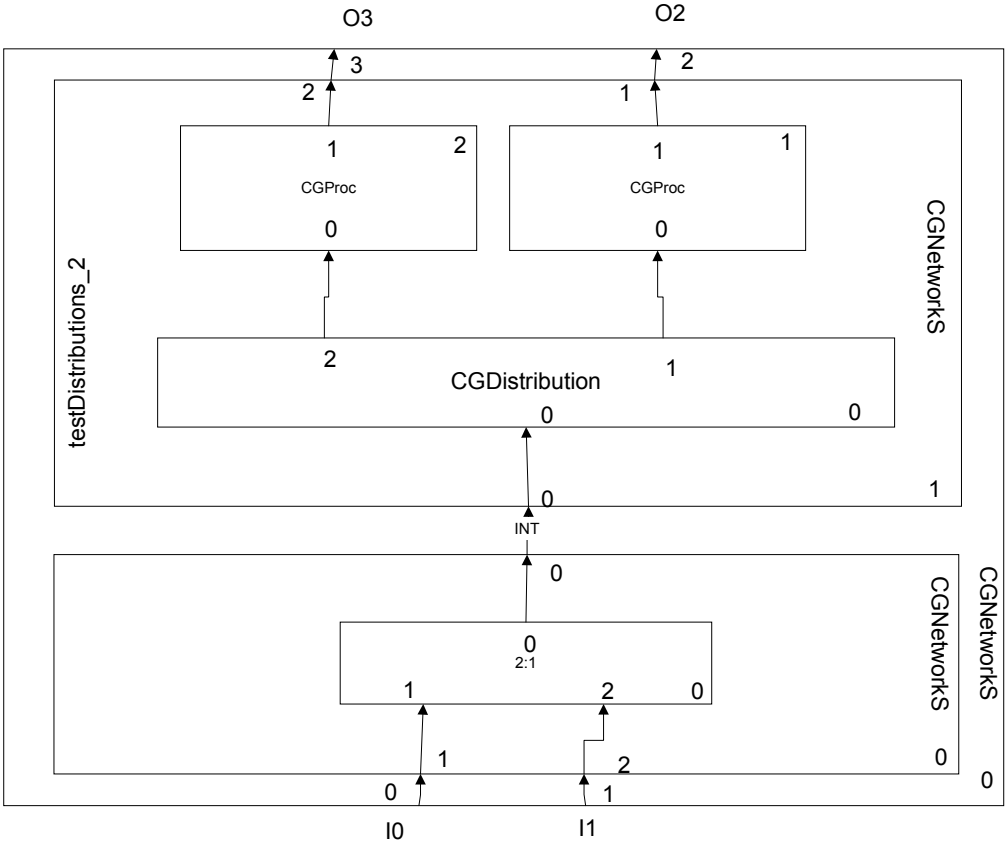
- netd->kill(); delete netd;
- net->DelNet(netd);

port/time	0	1	2	3	4	5	6	7	8
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
internal	0	0	1.0	1.5	-	-	-	-	-
2	0	0	2	2	-	-	-	-	-
3	0	0	1.0	1.5	2.0	2.0	2.5	3.0	3.5

4. System Functionality

4.1 Distributions system 1 (testS_1)

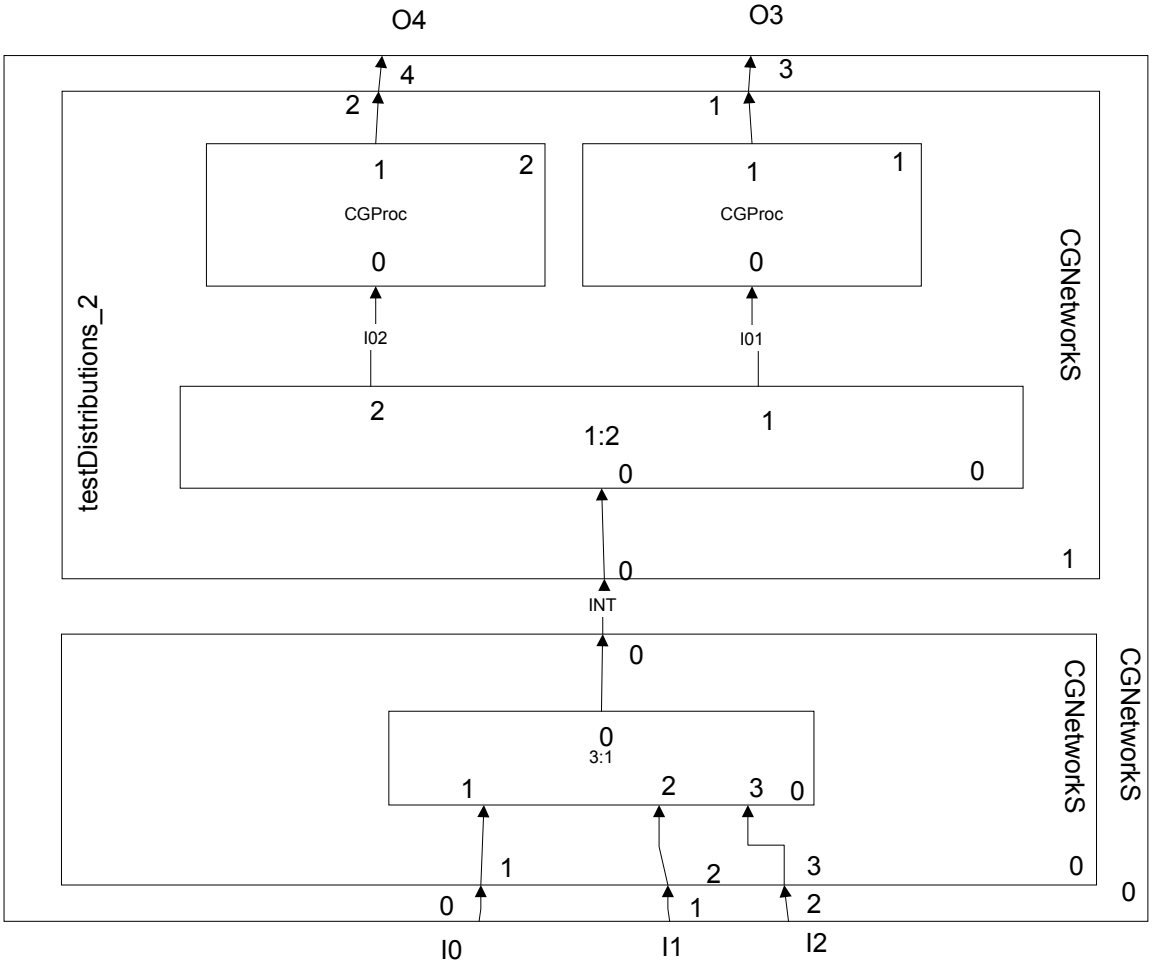
This is an extension of **testDistributions_2** but with a mutex 2:1. All processors have 1 unity delay.



t	0	1	2	3	4	5	6	7	8	9
I0	1	-	-	-	1	-	-	-	1	-
I1	-	-	1	-	-	-	1	-	-	-
INT	1	-	1	-	1	-	1	-	1	-
02	0	1	1.5	2	2.5	3.5	4	4.5	5	6
03	0	0	0	1	1.5	2	2.5	3.5	4	4.5

4.2 Distributions system 2 (testS_2)

This is a tester for mutex.. All processors have 1 unity delay. Mutex 3:1 has bypass. Equations for processors are: x0[0]+y1[0] for external and y1[0]+0.5 for internal.

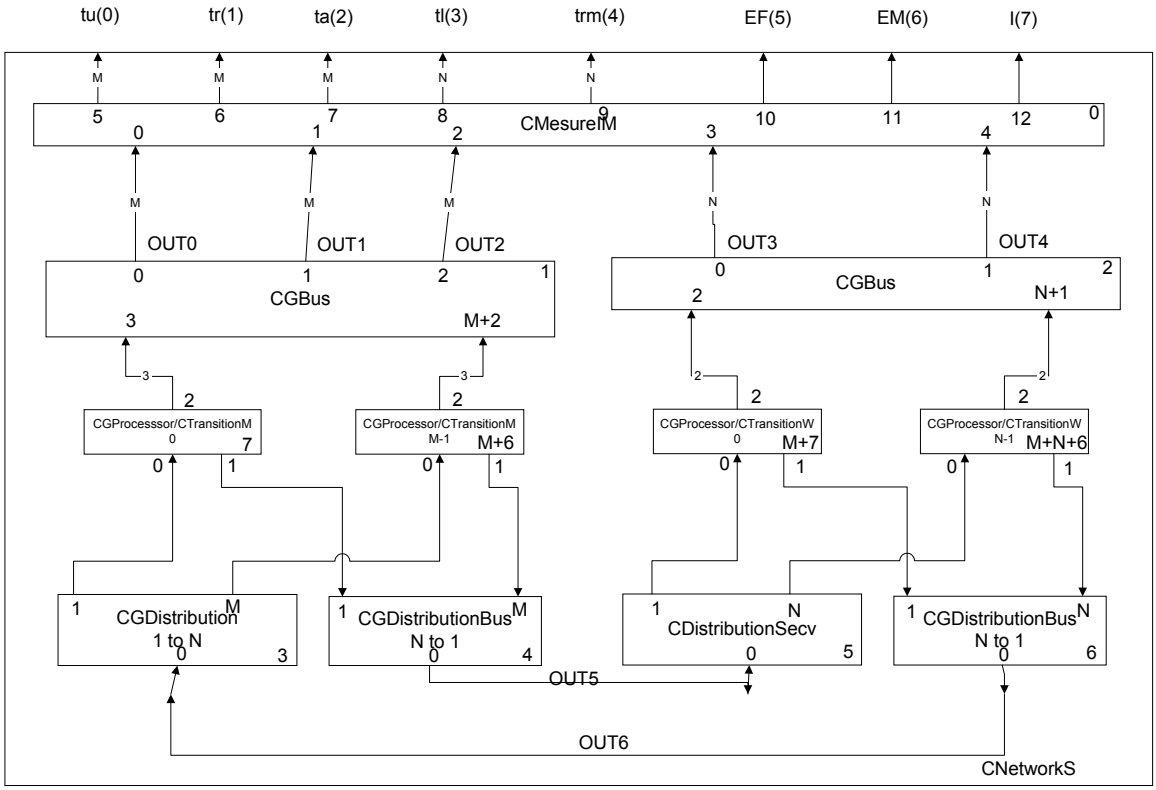


t	0	1	2	3	4	5	6	7	8	9
I0	-	-	-	-	1;1	-	-	-	-	-
I1	1;1	-	-	-	-	-	1;2	-	1;1	-
I2	-	-	1;2	-	-	-	-	-	-	-
INT	1;1	-	1;2	-	1;1	-	1;2	-	1;1	-
O3	0	1	1.5	2	2.5	3.5	4	4.5	5	6
O4	0	0	0	1	1.5	2	2.5	3.5	4	4.5
I01	1	-	-	-	1	-	-	-	1	-
I02	-	-	1	-	-	-	1	-	-	-

5. Waiting systems

5.1 Machines repaired by workers (sa_m_w)

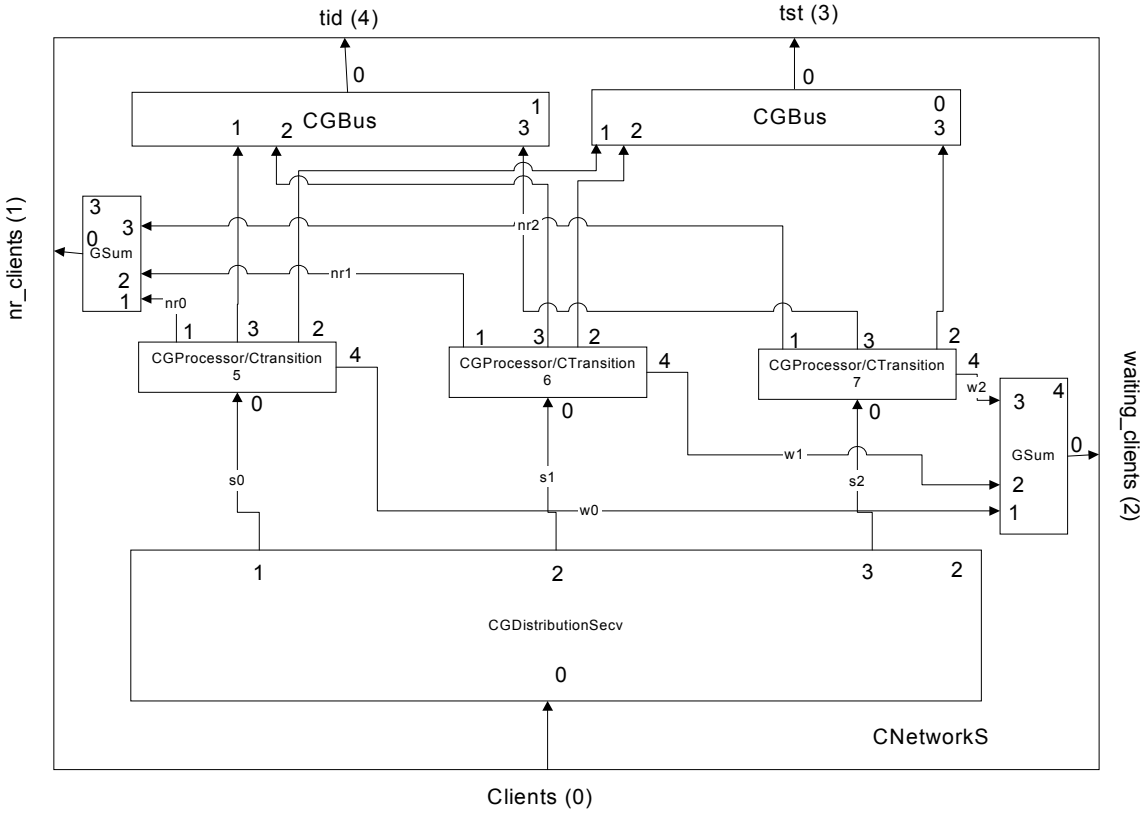
tu is the effective run time
tr is the repairing time
ta is waiting to be repaired time
tl=L is lazy time for workers
trm=W is working time for workers
First machine has “1” unity delay.
Second machine has “2” unity delay.
All workers have “3” unity delay.
Coordinator may be or may be not disabled.
CGDistributionBus with nr 6 is with bypass.



t	0	1	2	3	4	5	6	7	8	9
OUT0(tu)	0	1	1	1	1	2	2	2	2	3
	0	1	2	2	2	2	3	4	4	4
OUT1(tr)	0	0	1	2	3	3	4	5	6	6
	0	0	0	1	2	3	3	3	4	5
OUT2(ta)	0	0	0	0	0	0	0	0	0	0
OUT3(L)	0	1	1	1	1	2	2	2	3	3
	0	1	2	2	2	2	3	4	4	4
OUT4(W)	0	0	1	2	3	3	4	5	6	6
	0	0	0	1	2	3	3	3	4	5
OUT5(R)	-	1;3	1;4	-	-	1;3	-	1;4	-	1;3
OUT6(A)	-	-1;3	-1;4	-	1;3	-1;3 1;4	-	-1;4	1;3	-1;3
M0	W	R	R	R	W	R	R	R	W	R
M1	W	W	R	R	R	W	W	R	R	R
W0	L	W	W	W	L	W	W	W	L	W
W1	L	L	W	W	W	L	L	W	W	W

5.2. Parallel services without priorities version 1 (sa_m_sp_1)

Processor 0 has 2.0 delay; Processor 1 has 2.5 delay; Processor 2 has 3.0 delay.
Clock precision is 0.5.
The system consists in M parallel stations (M=3), each with own queue. Each station sits in a circular queue; so when a station receives a client, it is put on the end of the queue. When a client arrive it is assigned to the first station in the queue, whatever the station has clients or not. The number of served clients is counting. If the station is idle, (has no client) an idle time (tid) is count for each station; and if the station is running a service time (tst) is count for each station. When a client come to be served the waiting time for him is added to the waiting time of the clients.



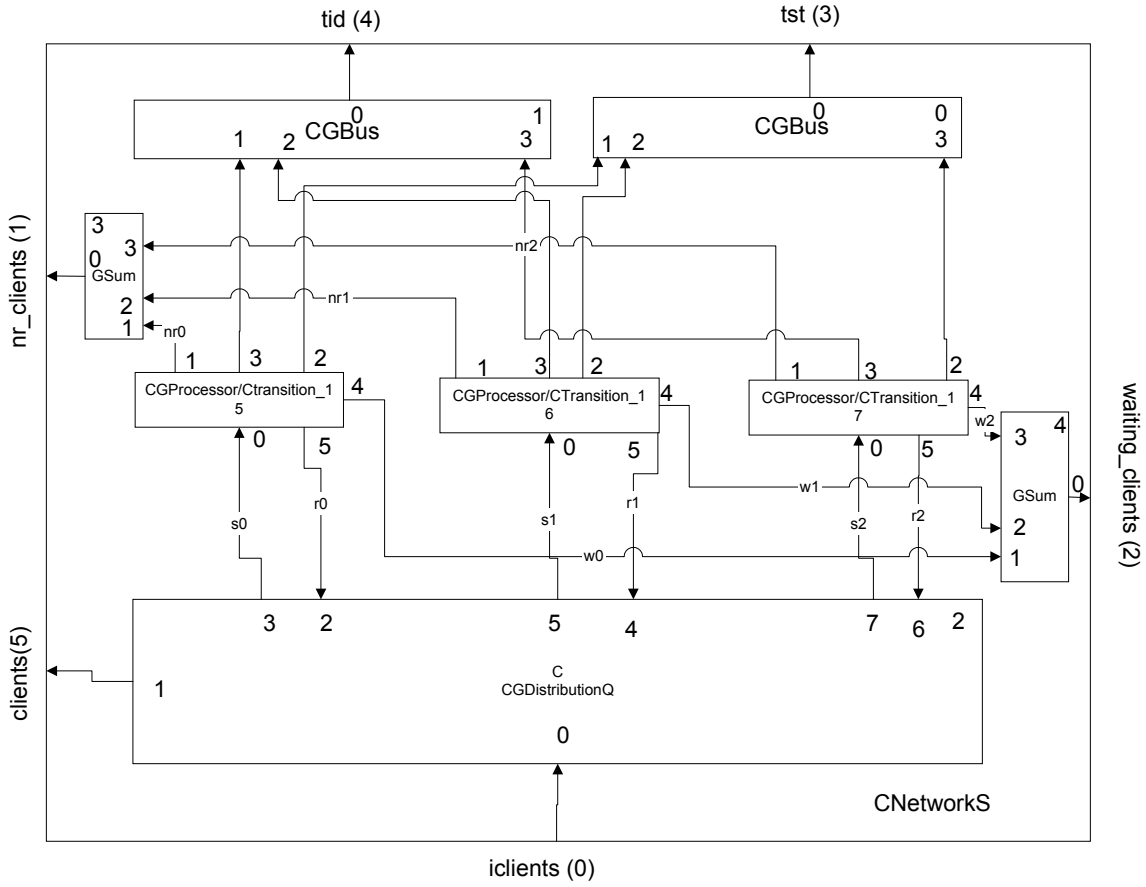
t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
clients	0	1	1	1	-	1	1	1	-	-	-
s0	-	1	-	-	-	1	-	-	-	-	-
s1	-	-	1	-	-	-	1	-	-	-	-
s2	-	-	-	1	-	-	-	1	-	-	-
nr0	0	0	0	0	0	1	1	1	1	2	2
nr1	0	0	0	0	0	0	0	1	1	1	1
nr2	0	0	0	0	0	0	0	0	0	1	1
w0	0	0	0	0	0	0	0	0	0	0	0

w1	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5
w2	0	0	0	0	0	0	0	0	0	1	1
nr_clients	0	0	0	0	0	1	1	2	2	4	4
waiting_cl	0	0	0	0	0	0	0	0.5	0.5	1.5	1.5
tid	0 0 0	0.5 0.5 0.5	0.5 1 1	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	1 1 1.5
tst	0 0 0	0 0 0	0.5 0 0	1 0.5 0	1.5 1 0.5	2 1.5 1	2.5 2 1.5	3 2.5 2	3.5 3 2.5	4 3.5 3	4 3.5 3
qs	s0 s1 s2	s1 s2 s0	s2 s0 s1	s0 s1 s2	s0 s1 s2	s1 s2 s0	s2 s0 s1	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2
q0	-	-	-	-	-	-	-	-	-	-	-
q1	-	-	-	-	-		3.0	-	-	-	-
q2	-	-	-	-	-	-	-	3.5	3.5	-	-

t	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10
clients	1	1	1	-	-	-	-	-	-	-
s0	1	-	-	-	-	-	-	-	-	-
s1	-	1	-	-	-	-	-	-	-	-
s2	-	-	1	-	-	-	-	-	-	-
nr0	2	2	2	2	3	3	3	3	3	3
nr1	1	2	2	2	2	2	3	3	3	3
nr2	1	1	1	1	2	2	2	2	2	2
w0	0	0	0	0	0	0	0	0	0	0
w1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
w2	1	1	1	1	2	2	2	2	2	2
nr_clients	4	5	5	5	7	7	8	8	8	8
waiting_clients	1.5	1.5	1.5	1.5	2.5	2.5	2.5	2.5	2.5	2.5
tid	1.5 1 1.5	1.5 1 1.5	1.5 1 1.5	1.5 1 1.5	1.5 1 1.5	2 1 1.5	2.5 1 1.5	3 1.5 1.5	3.5 2 1.5	4 2.5 1.5
tst	4 4.5 4	4.5 5 4.5	5 5.5 5	5.5 6 5.5	6 6.5 6	6 7 6.5	6 7.5 7	6 7.5 7.5	6 7.5 8	6 7.5 8.5
qs	s1 s2 s0	s2 s0 s1	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2
q0	-	-	-	-	-	-	-	-	-	-
q1	-	-	-	-	-	-	-	-	-	-
q2	-	-	6.5	6.5	-	-	-	-	-	-

5.3. Parallel service without priorities version 2 (sa_m_sp_2)

Processor 0 has 2.0 delay; Processor 1 has 2.5 delay; Processor 2 has 3.0 delay.
Clock precision is 0.5.
The system consists in M parallel stations (M=3), each is without queue; the queue is in the system and it is without priorities. Each station sits in a circular queue; so when a station is free it is put on the end of the queue. When a client arrives, it is assigned to the first station in the queue. The number of served clients is counting. If the station is idle, (has no client) an idle time (tid) is count for each station; and if the station is running a service time (tst) is count for each station. When a client come to be served the waiting time for him is added to the waiting time of the clients.



t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
iclients	0	1	1	1	-	1	1	1	-	-	-
clients	0	1	2	3	-	4	5	6	-	-	-
r0	1	-	-	-	-	1	-	-	-	1	-
r1	1	-	-	-	-	-	-	1	-	-	-
r2	1	-	-	-	-	-	-	-	-	1	-
s0	-	1(0.5)	-	-	-	1(2.5)	-	-	-	1(3.5)	-

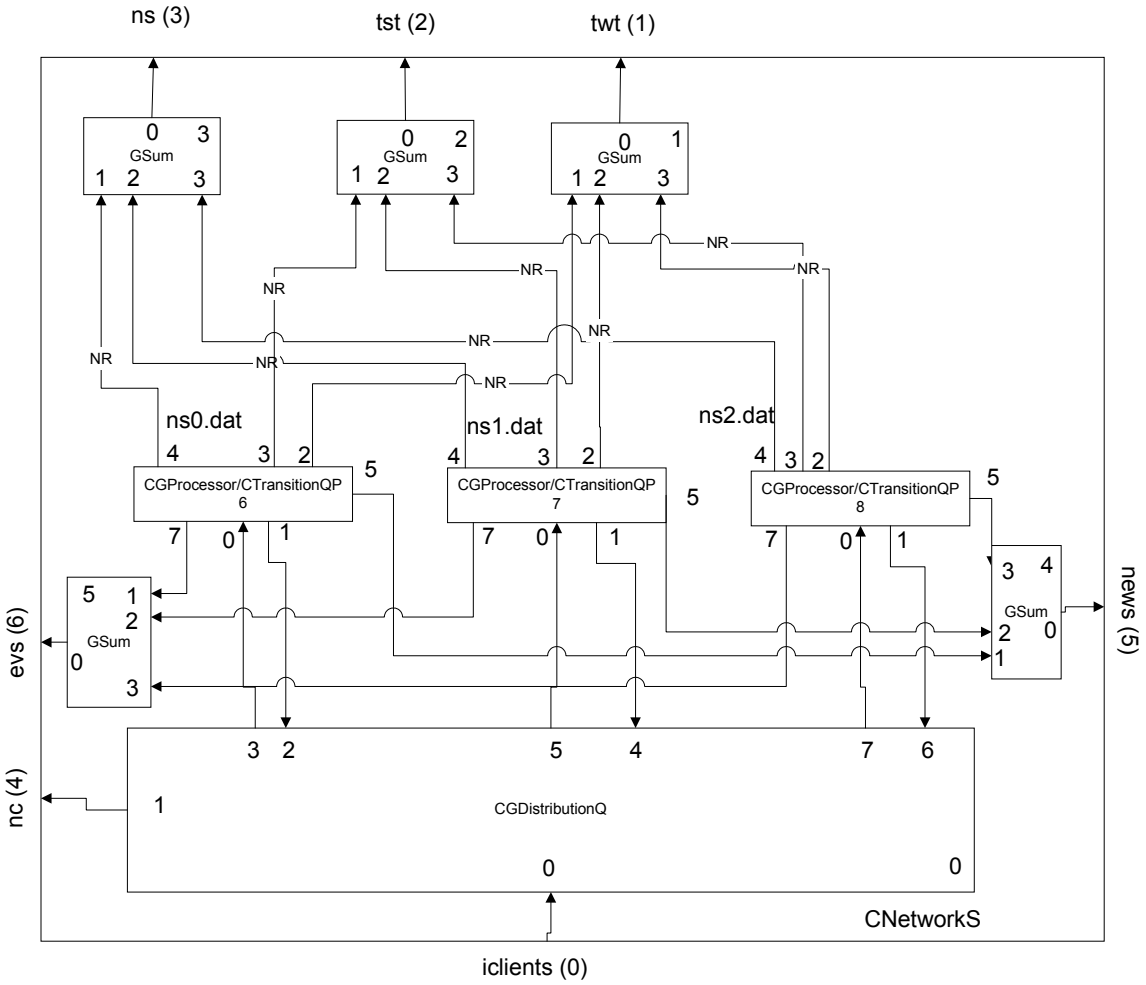
s1	-	-	1(1)	-	-	-	-	1(3)	-	-	-
s2	-	-	-	1(1.5)	-	-	-	-	-	-	-
nr0	0	0	0	0	0	1	1	1	1	2	2
nr1	0	0	0	0	0	0	0	1	1	1	1
nr2	0	0	0	0	0	0	0	0	0	1	1
w0	0	0	0	0	0	0	0	0	0	1	1
w1	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5
w2	0	0	0	0	0	0	0	0	0	0	0
nr_clients	0	0	0	0	0	1	1	2	2	4	4
cl_waiting	0	0	0	0	0	0	0	0.5	0.5	1.5	1.5
tid	0 0 0	0.5 0.5 0.5	0.5 1 1	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 1.5	0.5 1 2
tst	0 0 0	0 0 0	0.5 0 0	1 0.5 0	1.5 1 0.5	2 1.5 1	2.5 2 1.5	3 2.5 2	3.5 3 2.5	4 3.5 3	4.5 4 3
qs	s0 s1 s2	s1 s2 s0	s2 s0 s1	s0 s1 s2	s0 s1 s2	s1 s2 s0	s2 s0 s1	s0 s1 s2	s0 s1 s2	s0 s1 s2	s0 s1 s2
q	-	-	-	-	-	-	3.0	3.0 3.5	3.5	-	-
qw	s0 s1 s2	s1 s2	s2	-	-	-	-	-	-	s0 s2	s2

t	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10
iclients	1	1	1	-	-	-	-	-	-	-
clients	7	8	9	-	-	-	-	-	-	-
r0	-	-	1	-	-	-	1	-	-	-
r1	-	1	-	-	-	-	1	-	-	-
r2	-	-	-	-	-	-	1	-	-	-
s0	-	-	1(6.5)	-	-	-	-	-	-	-
s1	-	1(6)	-	-	-	-	-	-	-	-
s2	1(5.5)	-	-	-	-	-	-	-	-	-
nr0	2	2	3	3	3	3	4	4	4	4
nr1	1	2	2	2	2	2	3	3	3	3
nr2	1	1	1	1	2	2	2	2	2	2
w0	1	1	1	1	1	1	1	1	1	1
w1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
w2	0	0	0	0	0	0	0	0	0	0
nr_clients	4	5	6	6	6	6	9	9	9	9
cl_waiting	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
tid	0.5 1 2.5	0.5 1 2.5	0.5 1 2.5	0.5 1 2.5	0.5 1 2.5	0.5 1 2.5	0.5 1 2.5	1 1.5 3	1.5 2 3.5	2 2.5 4

tst	5 4.5 3	5.5 5 3.5	6 5.5 4	6.5 6 4.5	7 6.5 5	7.5 7 5.5	8 7.5 6	8 7.5 6	8 7.5 6	8 7.5 6
q	-	-	-	-	-	-	-	-	-	-
qw	-	-	-	-	-	-	-	-	-	-

5.4. Parallel service with priorities version 1 (sa_m_qp_1)

Processor 0 has 2.0 delay; Processor 1 has 2.5 delay; Processor 2 has 3.0 delay.
Clock precision is 0.5. M=3 number of stations;
nc=number of clients; news=evs=ns=number of served clients; twt=time of waiting for served clients; tst=time of served clients; iclients=input clients



t	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
iclients	-	1	2	1	-	1	2	1	-	-	-
r0	1	-	-	-	-	1	-	-	-	1	-
r1	1	-	-	-	-	-	-	1	-	-	-
r2	1	-	-	-	-	-	-	-	-	1	-
s0	-	1(1)	-	-	-	1(1)	-	-	-	1(2)	-
s1	-	-	1(2)	-	-	-	-	1(1)	-	-	-
s2	-	-	-	1(1)	-	-	-	-	-	-	-
ns0	0;0	-	-	-	-	1;0	-	-	-	2;0	-
ns1	0;0	-	-	-	-	-	-	0;1	-	-	-
ns2	0;0	-	-	-	-	-	-	-	-	1;0	-

nc	0 0	1 0	1 1	2 1	-	3 1	3 2	4 2	-	-	-
ns	0 0	-	-	-	-	1 0	-	1 1	-	3 1	-
tst	0 0	-	-	-	-	2 0	-	2 2.5	-	7 2.5	
twt	0 0	-	-	-	-	-	-	-	-	-	-
news	0	-	-	-	-	1	-	2	-	4	-
evs	0	-	-	-	-	1	-	2	-	4	-
q	-	-	-	-	-	-	3(2)	2.5(1)- 3(2)	3(2)	-	-
qw	s0 s1 s2	s1 s2	s2	-	-	s0-	-	s1-	-	s0- s2	s2

t	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10	10.5	11
iclients	2	1	2	1	1	-	-	-	-	-	-	-
r0	-	-	1	-	-	-	1	-	-	-	1	-
r1	-	1	-	-	-	-	1	-	-	-	-	1
r2	-	-	-	-	-	-	1	-	-	-	-	-
s0	-	-	1(2)	-	-	-	1(1)	-	-	-	-	-
s1	-	1(1)	-	-	-	-	1(1)	-	-	-	-	-
s2	1(2)	-	-	-	-	-	-	-	-	-	-	-
ns0	-	-	2;1	-	-	-	2;2	-	-	-	3;2	-
ns1	-	1;1	-	-	-	-	2;1	-	-	-	-	3;1
ns2	-	-	-	-	-	-	1;1	-	-	-	-	-
nc	4 3	5 3	5 4	6 4	7 4	-	-	-	-	-	-	-
ns	-	4 1	4 2	-	-	-	5 4	-	-	-	6 4	7 4
tst	-	9.5 2.5	9.5 4.5	-	-	-	12 9.5	-	-	-	14.5 9.5	17 9.5
twt	-	-	0 1.5	-	-	-	-	-	-	-	1.5 1.5	-
news	-	5	6	-	-	-	8	9	-	-	10	11
evs	-	5	6	-	-	-	8	9	-	-	10	11
q	-	-	7(1)	7(1) 7.5(1)	7(1) 7.5(1)	-	-	-	-	-	-	-
qw	-	-	-	-	-	s0- s1- s2	s2	s2	s2	s2	s2 s0	s2 s0 s1