Discrete Event Systems

The DEVS generator has the following specification:

Abbreviations

GEN : the generator

In1 : primary input port of the generator

In2 : secondary input port of the generator

σ : the time for the next scheduled internal event

ψ : the inter-arrival-time, the specified time between generated output events

φ : the phase, the discrete state of the generator

e : internal time elapsed since last event

Assumptions:

- Multiple input events with the same time-stamp are **not** allowed.
- When σ is zero the internal event must be generated immediately, hence there is an internal event at start-up
- Given a fixed set of input the internal state and output must be determinate.
- Neither the input value nor ψ can be zero or less, that would cause thrashing.

Functional Specification:

GEN has In1 which, upon receiving a value sets ψ to that value and adjusts σ. There is a bit of ambiguity here in the word “immediately”, there are several choices for σ. Candidate values for σ are:

- 0 : this would cause the immediate production of an internal event and the scheduling at the new time. (rejected)
- ψ : this is similar but would not create an immediate internal event. (rejected)
- ψ – e : a pretty good compromise, it credits the elapsed time. It has the problem that it could be 0, in that case 0 will be used. (accepted)

GEN has In2 which, upon receiving a value updates ψ to that value.

The initial state has: σ = 0; ψ = 10: and φ = “passive”.

The formal DEVS model:

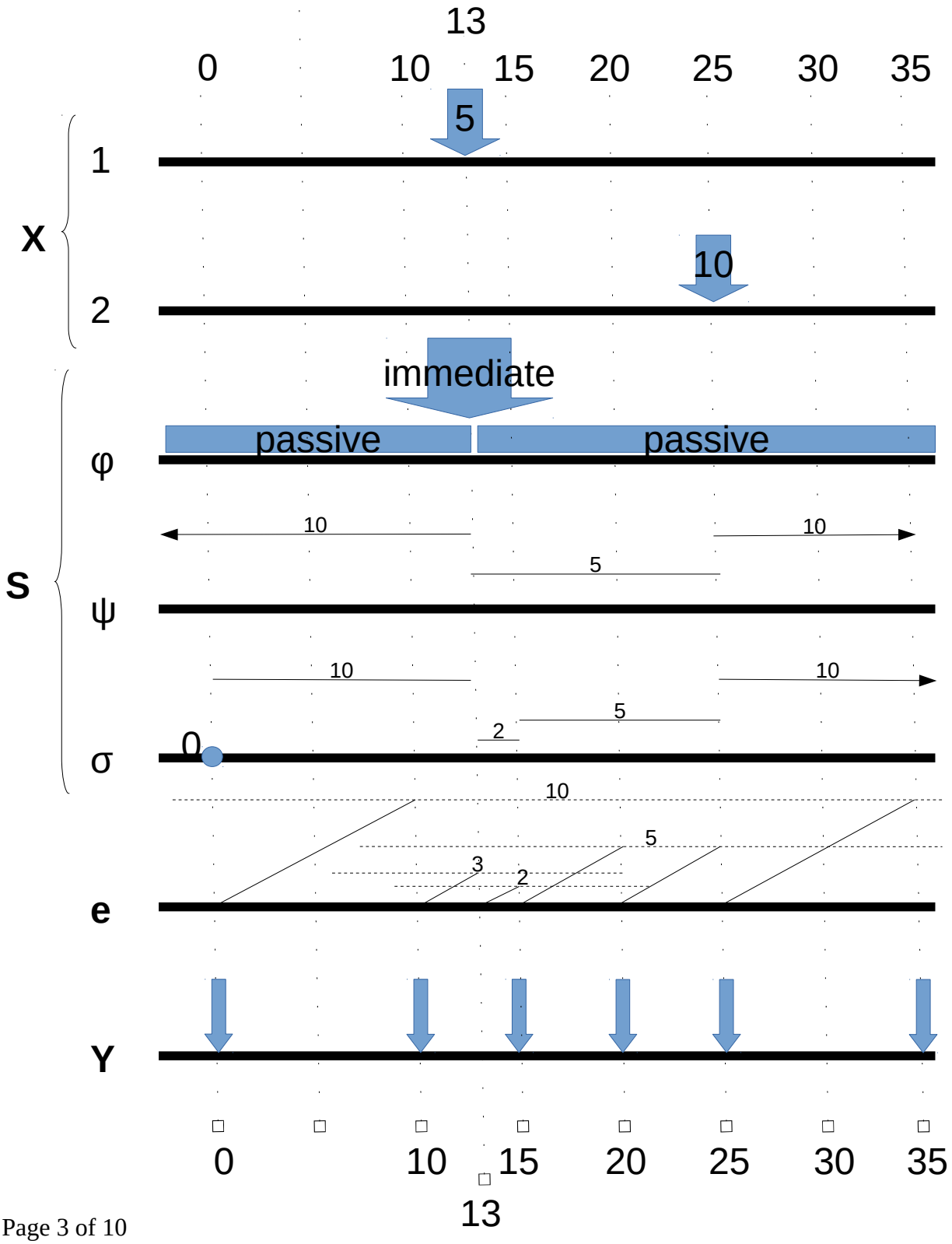
$DEVS_{\text{gen}} = [X_M, Y_M, S, \delta_{\text{ext}}, \delta_{\text{int}}, \delta_{\text{con}}, \lambda, ta]$.

$X = \{[t, v]; t \in \{\text{In1}, \text{In2}\}; v \in \mathbb{R}_{pos}^{inf}\}$
$Y = \{1\}$
$S = \{[\varphi, \psi, \sigma]; \varphi \in \{\text{passive}, \text{immediate}\}; \psi \in \mathbb{R}_{\text{pos}}^{inf}; \sigma \in \mathbb{R}_{0}^{inf}\}$.
$\delta_{\text{ext}}([\varphi, \psi, \sigma], e, [t, v])$
$= [\text{immediate}, v, v - e]$; *if* $t = \text{In1} \wedge v > e$;
$= [\text{immediate}, v, 0]$; *if* $t = \text{In1} \wedge v \leq e$;
$= [\text{passive}, v, \sigma - e]$; *otherwise*.

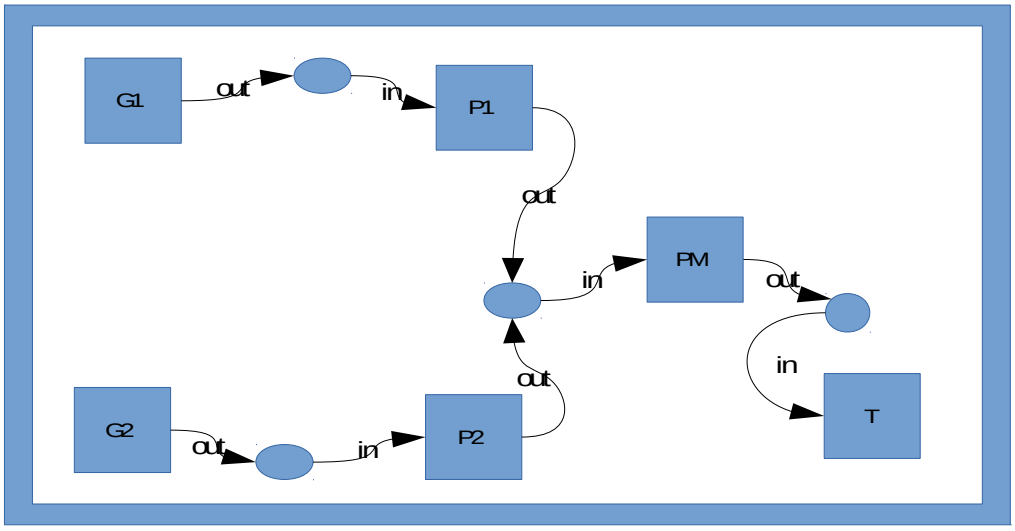
$\delta_{\text{int}}([\varphi, \psi, \sigma]) = [\text{passive}, \psi, \psi]$.
$\lambda([\varphi, \psi, \sigma]) = 1$.
$ta([\varphi, \psi, \sigma]) = \sigma$.

There is a slight ambiguity due to events which may occur at the same time, in the case of an internal and an external event the internal event is handled first so there is no ambiguity there. In the case of two external events a *confluent transition* function could be used to select, for example, the "in1" event over the "in2" event. Rather than do that I choose to simply prohibit the behavior as an assumption.

2) The following timing diagram illustrates the behavior of the generator. The priming internal event causes an output at zero. The other interesting case is where both an internal and external event arrive at time 25. The initial event is processed first.


13
0
10
15
20
25
30
35
5
1
X
10
2
immediate
passive
passive
φ
10
10
5
S
ψ
10
10
5
2
0
σ
10
5
3
2
e
Y
0
10
15
20
25
30
35
13

Problem 2: A three-processor system with two classes of job arrivals. The size of the buffers at all processors are infinite. The jobs are periodic and the event times are as indicated in the model.



The formal DEVS model has a mechanism in the "Select" function which will use a reasonable mechanism when the "PM" processor has a job of each class. A round-robin approach would be such a reasonable approach. In this model no initial events are generated by either of the generators. This is ambiguous in the specification. The following composite DEVS model is used for the simulation.

[Note: I am using a slightly different notation that what was introduced in class. I specify the state as a tuple inclosed in '[]' so it can be differentiated easily in the `δext` function. Also, when specifying the state I used the '{}' set notation rather than the cartesian product so that the fields can be named.]

$N_{\text{mpipe}} = \{X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC, Select\}.$

$V =$ the set of allowed jobs and their states
$J = \{1,2\}$ the types of jobs, the job classes
$InPorts = \{\text{in}\}; OutPorts = \{\text{out}\};$ possible port names

$X = \{(p,c,v) | \ p \in InPorts \wedge c \in J \wedge v \in V\};$ the set of input events representing the jol
$Y = \{(p,c,v) | \ p \in OutPorts \wedge c \in J \wedge v \in V\};$ the set of output events representing the
$D = \{G1, G2, P1, P2, M0, PM, T\};$ the names of the subcomponents

$M_{G1} = GEN(20); \quad M_{G2} = GEN(21);$ the job generators
$M_{P1} = BPROC(5); \quad M_{P2} = BPROC(7);$ the buffered job processors
$M_{PM} = BPROC(10); \quad T = SINK();$ the final job processor and sink
$EIC = \emptyset; \quad EOC = \emptyset;$ the generators and transducer are part of the model

$IC =$
$\{[(G1,\text{out}),(P1,\text{in})]$
$[(P1,\text{out}),(PM,\text{in1})]$
$[(G2,\text{out}),(P2,\text{in})]$
$[(P2,\text{out}),(PM,\text{in})]$
$[(PM,\text{out}),(T,\text{in})]\}$
$Select(D') =$ fair selection of a component ready to produce an event

This model makes use of several DEV models, some of which take parameters. Such as the following generator mode which takes a “PERIOD” representing the time between job events.

Has a control variable "PERIOD".
$GEN\ (PERIOD\ ) = [\ X_M , Y_M , S , \delta_{\text{ext}} , \delta_{\text{int}} , \delta_{\text{con}} , \lambda , ta\ ].$

$X = \{\}$. takes no input
$Y = \{1\}$
$S = \{[\varphi , \sigma] \mid \varphi \in \{\text{passive}\ , \text{active}\ \}; \sigma \in \mathbb{R}_0^{inf}\}.$
$\delta_{\text{ext}}([\varphi , \sigma], e , x) = \varnothing$. no input so no input events
$\delta_{\text{int}}([\varphi , \sigma]) = [\text{active}\ , PERIOD\ ].$
$\lambda([\varphi , \sigma]) = 1.$
$ta([\varphi , \sigma]) = \sigma .$

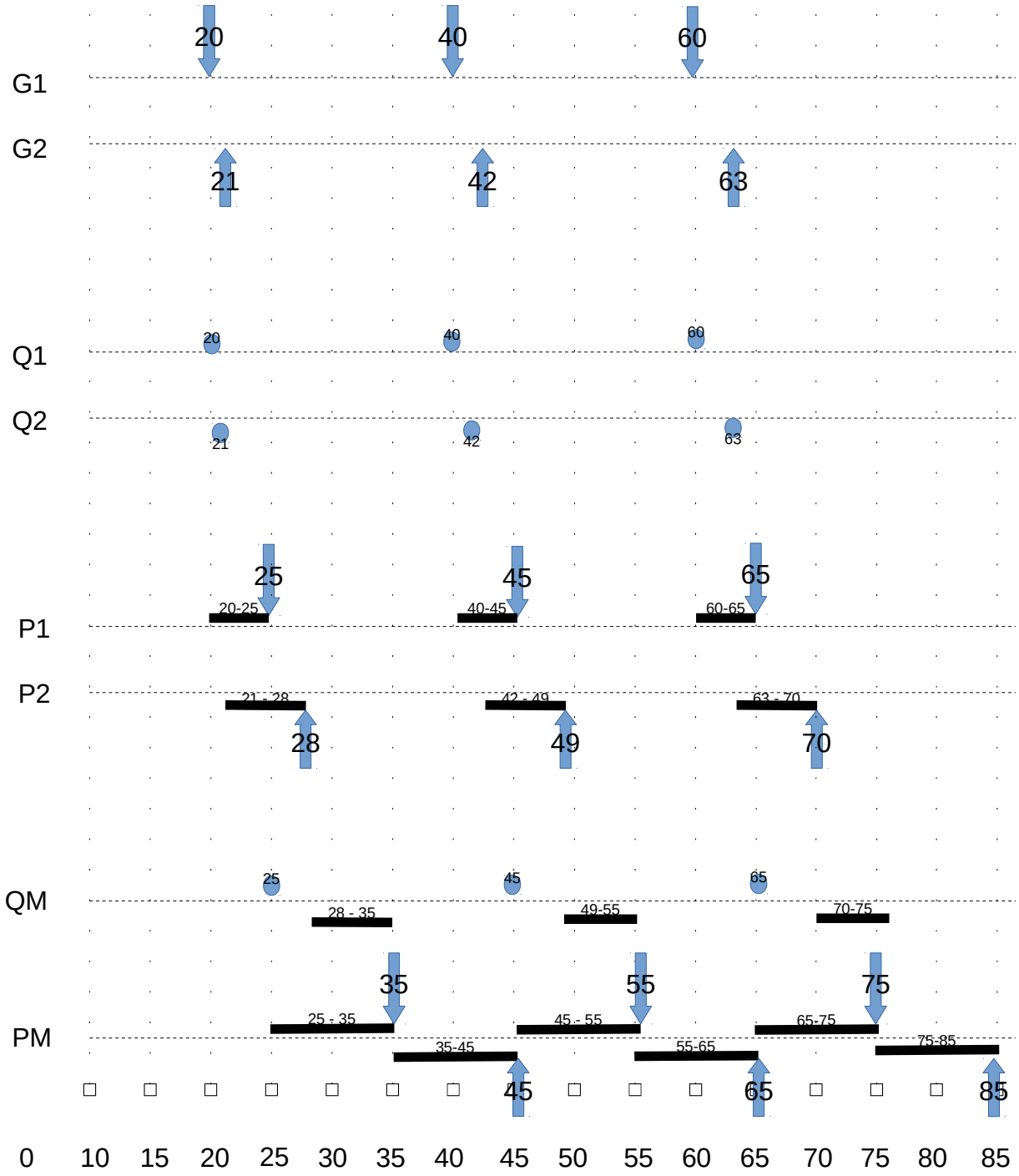
The transducer (sink) takes no parameters. It is very simple.

$SINK\ = [X , Y , S , \delta_{\text{ext}} , \delta_{\text{int}} , \lambda , ta\ ].$

$X = \{\text{anything}\ \}$
$Y = \varnothing$
$S = \{[\varphi] \mid \varphi \in \{\text{passive}\ \}.$
$\delta_{\text{ext}}([\varphi], e , x) = [\text{passive}\ ]\};$
$\delta_{\text{int}}([\varphi]) = [\text{passive}\ ].$
$\lambda([\varphi]) = 1.$
$ta([\varphi]) = \infty .$

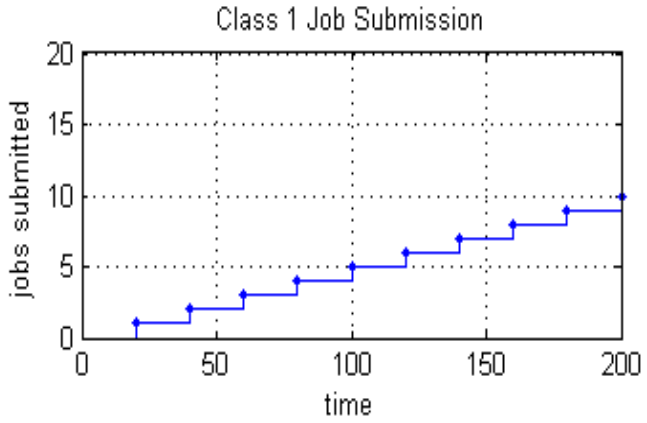
The buffered processor model takes a parameter for the processing-time, PROC_TIME, but as it has an infinite buffer no buffer length is required. This model does not carry a state variable for the specific job. This could be added but would require a queue for the jobs so I left it out. It is not needed for the specified analysis.

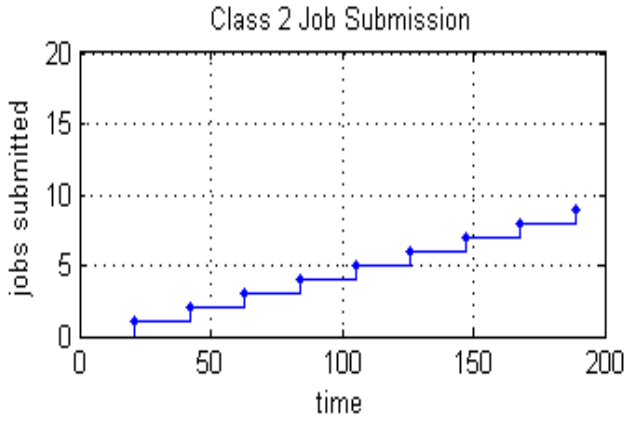
b) The following timing diagram shows the first 6 events (3 of each class). Note that there is no initial event the first event comes after the generator period has expired. The figure shows Class-1 events above the line and Class-2 below the line.

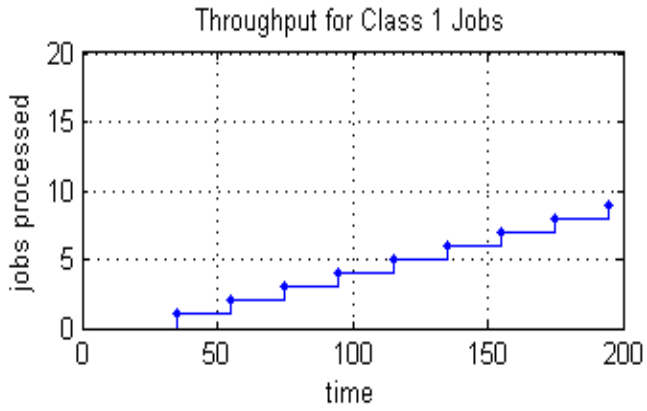


I also split the BPROC into a Q# and P# part for the timing diagram so that the in-queue period and the busy periods could be differentiated. It can be seen that the turn-around times for the Class-2 jobs improves approaching the optimal time of (7 + 10 = 17) time units. The Class-1 jobs are consistently at their optimal (5 + 10 = 15) time units.

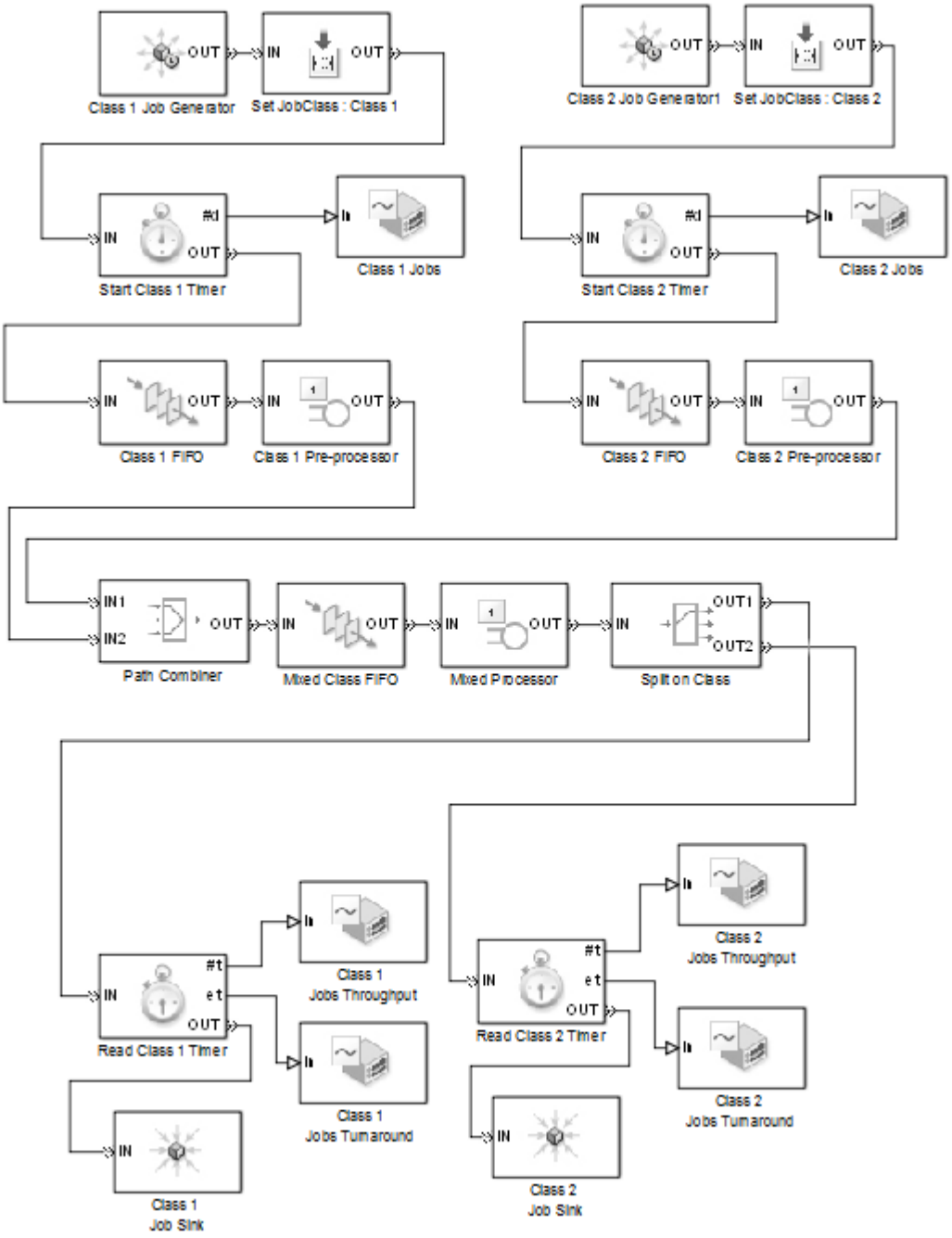
c) This model has been implemented with Matlab/SimEvents using server blocks as the processor. The following figures are from that model. The job classes have been implemented the attribute JobClass, setting it to 1 for Class-1 and 2 for Class-2. There is an opportunity for a Class-1 and a Class-2 job to arrive at QM at the same time. I allow this indeterminacy but a priority can be set to prevent this.







Confirming what was obtained in the timing study of part (b).


OUT
Class 1 Job Generator
IN
OUT
Set JobClass : Class 1
OUT
Class 2 Job Generator1
IN
OUT
Set JobClass : Class 2
IN
#i
OUT
Start Class 1 Timer
In
Class 1 Jobs
IN
#i
OUT
Start Class 2 Timer
In
Class 2 Jobs
IN
OUT
Class 1 FIFO
IN
1
OUT
Class 1 Pre-processor
IN
OUT
Class 2 FIFO
IN
1
OUT
Class 2 Pre-processor
IN1
IN2
OUT
Path Combiner
IN
OUT
Mixed Class FIFO
IN
1
OUT
Mixed Processor
IN
OUT1
OUT2
Split on Class
IN
#t
et
OUT
Read Class 1 Timer
In
Class 1
Jobs Throughput
In
Class 1
Jobs Turnaround
IN
Class 1
Job Sink
IN
#t
et
OUT
Read Class 2 Timer
In
Class 2
Jobs Throughput
In
Class 2
Jobs Turnaround
IN
Class 2
Job Sink