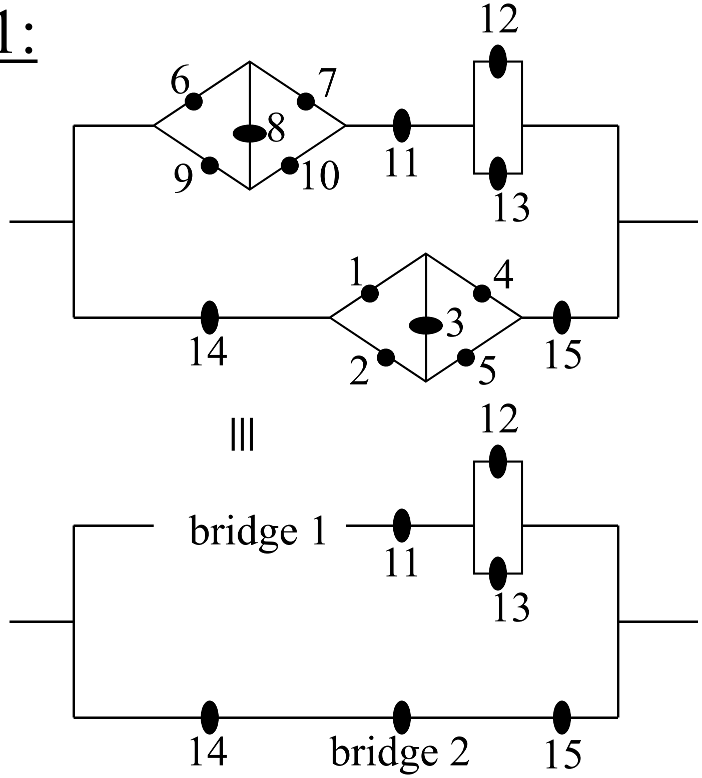
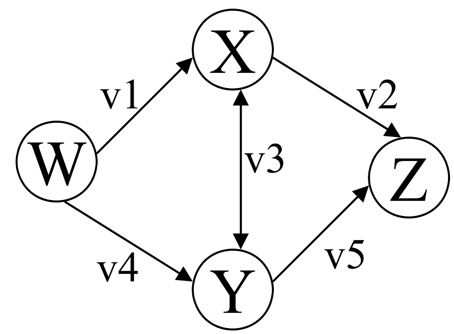
Chap 11: Hierarchical Models

Objective: to avoid large models so as to improve solution efficiency.

Ex1:



upper-level model
(a reliability block diagram)



Lower-level model for a bridge
(a reliability graph)

P.265

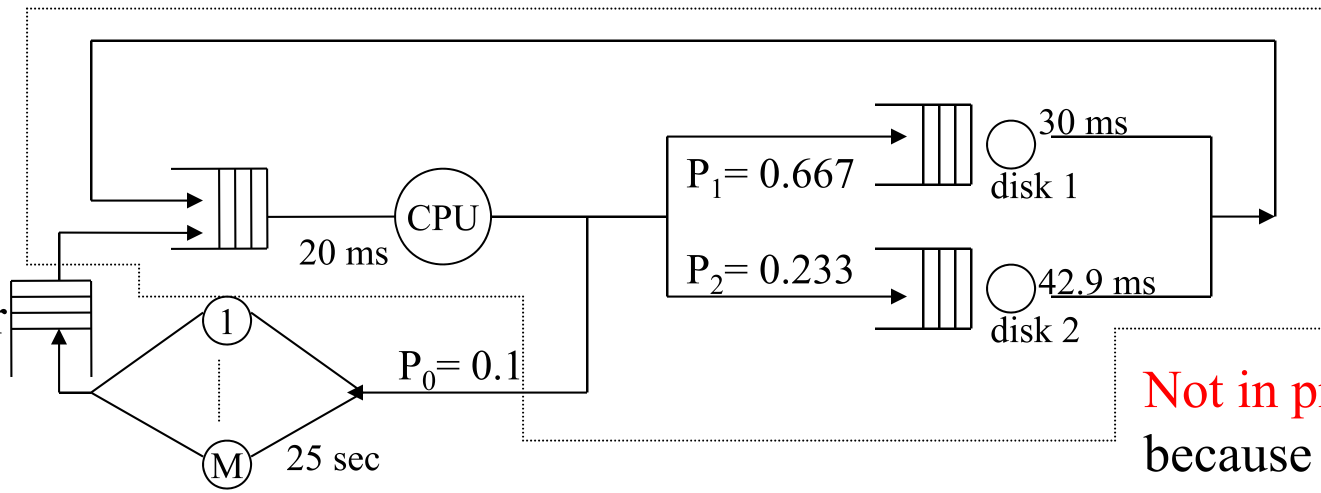
Partial sharpe code shown

```
relgraph   rbridge (v1, v2, v3, v4, v5)
w    x     exp (v1)
x    z     exp (v2)
w    y     exp (v4)
y    z     exp (v5)
bidirect
x    y     exp (v3)
end
block      rel-in-block
comp       11    exp (u11)
comp       12    exp (u12)
comp       13    exp (u13)
comp       14    exp (u14)
comp       15    exp (u15)
comp       bridge1   cdf (rbridge; u1, u2, u3, u4, u5)
comp       bridge2   cdf (rbridge; u6, u7, u8, u9, u10)
parallel   C   12   13
series     D   bridge1   11   C
series     E   14   bridge2   15
parallel   top   D   E
end
eval (rel-in-block)   0   50000   500
end
```

Ex2: A queuing model with resource constraints

P.277

# of running jobs in the central server is limited to n < M

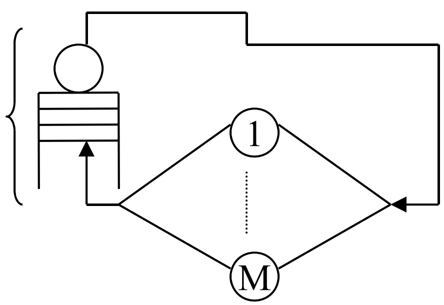


Within the dashed line is the central server system

Not in product-form because of resource limitations causing input flow ≠ output flow

|||

A load dependent center

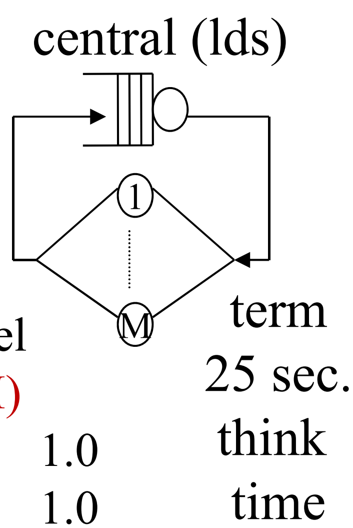


$$X(i) = \begin{cases} X(i), \text{if } i = 1,2,\ldots n \\ X(n), \text{if } i > n \end{cases}$$

In product-form: both servers can be evaluated independently

* low-level model

pfqn inner(n)
CPU disk1 P1
CPU disk2 P2
CPU CPU $1-P_1P_2$
disk1 CPU 1
disk2 CPU 1
end
CPU fcfs 1000/20
disk1 fcfs 1000/30
disk2 fcfs 1000/42.9
end
chain1 n
end
end



* high-level model

pfqn outer(M)
term central 1.0
central term 1.0
end

*station types
term is 1/25
central lds X(1), X(2), X(3), X(4)
end
chain1 M
end

* define function for lds throughput X(n)
func X(n) tput(inner,CPU;n)*(1- P1-P2)

* can also be obtained as

* (1000/20 * util(inner,CPU;n))*(1- P1-P2)

* by Little's Law, i.e., $x_{CPU} = \mu_{CPU} * \rho_{CPU}$

Service rate of CPU

Utilization of CPU

bind
P1 0.667
P2 0.233
end

* reporting each terminal user's response time in
* the central system as the number of users (M)
* increases

loop i, 0, 4, 1
expr 5*(2^i)
expr rtime(outer, central; 5*(2^i))
end
end

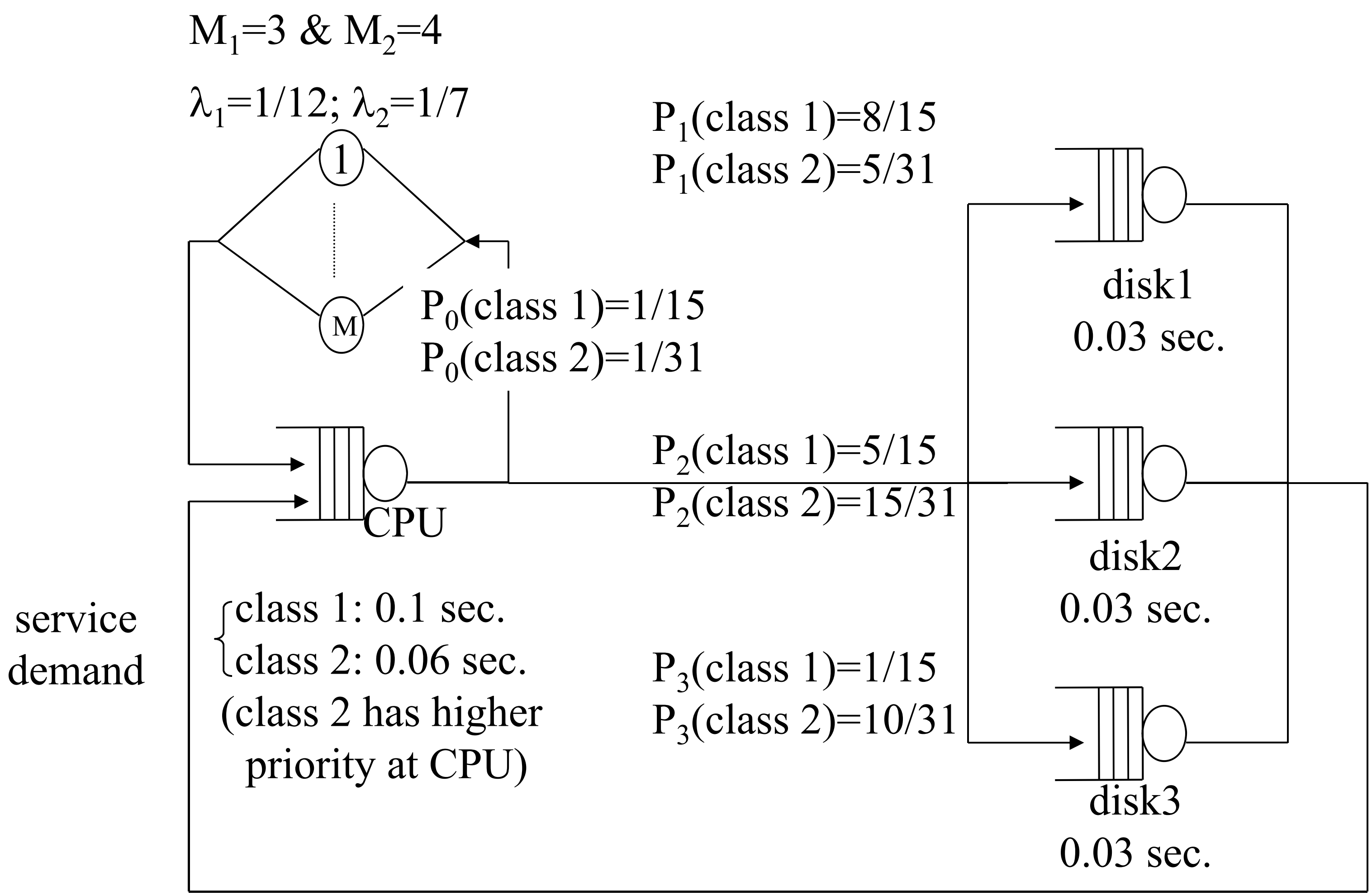
Ex3: A queuing model with job priorities

P.284 text

two classes of jobs: 1 & 2

1 → low priority

2 → high priority at the CPU only



* performance measures of interest: response time & queue length at CPU.
* not in product-form because of priority scheduling.

Approximation solution: suppose u_2 is the utilization of the CPU dedicated to class 2 jobs. Then the CPU service rate for class 1 jobs is slowed down by a factor of $(1-u_2)$

* we don't know u_2 since it is an output, but we need it as an input for class 1 jobs.

$\therefore$ use iterative technique

Create two CPUs, one for class 1 & the other for class 2, with the CPU service rate to class 1 jobs reduced by a factor of $(1-u_2)$

Sharpe code *(see p.285, text)*

```
mpfqn    iter (M1, M2, u2)
* chain 1 for class 1 jobs
chain 1
CPU1      disk1        8/15
CPU1      disk2        5/15
CPU1      disk3        1/15
CPU1      terminals    1/15
disk1     CPU1         1
disk2     CPU1         1
disk3     CPU1         1
end
* chain 2 for class 2 jobs
chain 2
CPU2      disk1        5/31
CPU2      disk2        15/31
CPU2      disk3        10/31
CPU2      terminals    1/31
disk1     CPU2         1
disk2     CPU2         1
disk3     CPU2         1
terminals CPU2         1
end
end
```

Section 1: routing prob. per class

```
* Section 2: server types
CPU1      fcfs    (1-u2)*1/0.1
end
CPU2      fcfs    1/0.06
end
disk1     fcfs    1/0.03
end
disk2     fcfs    1/0.03
end
disk3     fcfs    1/0.03
end
terminals    is
1            1/12
2            1/7
end
end
```

Section 2

class 1 → 1

class 2 → 2

Service rate of class 1 jobs is reduced by a factor of $(1-u_2)$

```
* Section 3: number of jobs per class
1         M1
2         M2
end
```

Section 3

* we don't know what the initial value of u_2 is,
* so make a guess u_2=0 initially

  <u>bind</u> u_2 <u>mutil</u> (iter, CPU2, 2; $\underbrace{3, 4, 0}$)

  system name — station name — chain 2 — parameters for M1, M2, & u_2

* continue this for a sufficient # of iterations

  until u_2 converges ⇒ try 5 times

  <u>loop</u> i, 1, 5, 1

  <u>bind</u> u_2 <u>mutil</u> (iter, CPU2, 2; 3, 4, u_2)

  <u>end</u>

* outputs are:
* i=1 $u_2 \leftarrow 0.659839$
* i=2 $u_2 \leftarrow 0.659838$
* i=3 $u_2 \leftarrow 0.659838$
* (converged after 3 iterations)
* try starting u_2 with another initial value,
* say u_2 =0.9

  <u>bind</u> u_2 0.9

  <u>loop</u> 1, 1, 5, 1

  <u>bind</u> u_2 mutil (iter, CPU2, 2; $\underbrace{3, 4, u_2}$)

  <u>end</u>

M1=3; M2=4 & u_2 is equal to the u_2 in the previous iteration

* outputs are
* i=1 $u_2 \leftarrow 0.660454$
* i=2 $u_2 \leftarrow 0.659839$
* i=3 $u_2 \leftarrow 0.659838$

u_2 is also converged in 3 iterations

* printing response time & queue size

  <u>expr</u> <u>mrtime</u> (iter, CPU1, 1; 3, 4, u_2)

  <u>expr</u> <u>mrtime</u> (iter, CPU2, 2; 3, 4, u_2)

  <u>mqlength</u> (iter, CPU1, 1; 3, 4, u_2)

  <u>mqlength</u> (iter, CPU2, 2; 3, 4, u_2)

* outputs are
* $R_{1,CPU}$=0.47534
* $R_{2,CPU}$=0.10511
* $\overline{n}_{1,CPU} = 1.0911$
* $\overline{n}_{2,CUP} = 1.1559$

  <u>end</u>

to be compared with the corresponding parameter values without priority scheduling

$R_{1,CPU}$=0.28483

$R_{2,CPU}$=0.15834

$\overline{n}_{1,CPU} = 0.76545$

$\overline{n}_{2,CUP} = 1.5197$

Ex4: M/M/1/k queue with server failure & repair

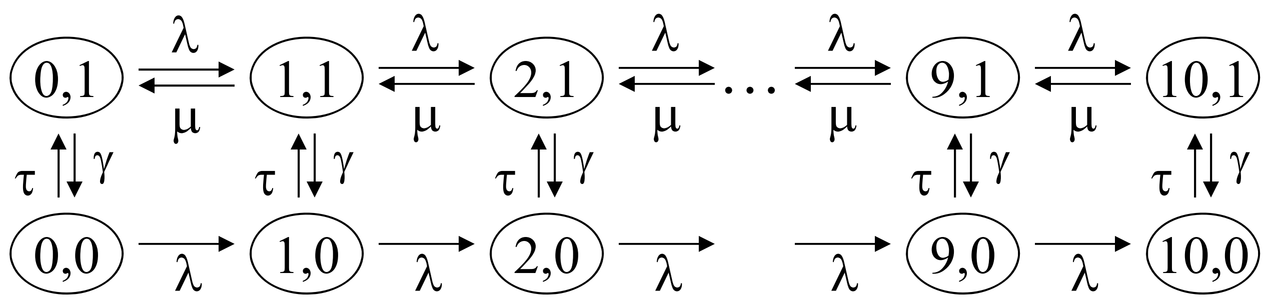
P.233, text & p.294

γ: failure rate λ: job arrival rate

τ: repair rate μ: job service rate

1-level model

M/M/1/10



State representation (a, b)

# of jobs

$\begin{cases} 1 & \text{alive} \\ 0 & \text{failed} \end{cases}$

prob {idle server} = $\text{prob}_{(0,0)} + \text{prob}_{(0,1)}$

rejection probability = $\text{prob}_{(10,0)} + \text{prob}_{(10,1)}$

Two-level model

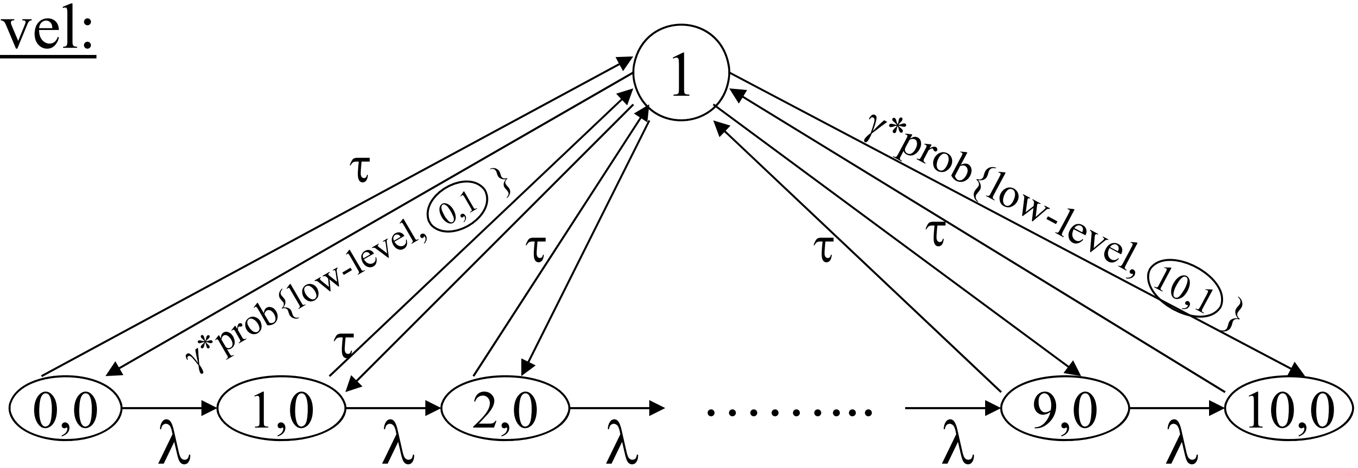
observation: $\underbrace{\text{job arrivals/services are much faster than server failures/repairs}}$

$\therefore$ the assumption below is justified:

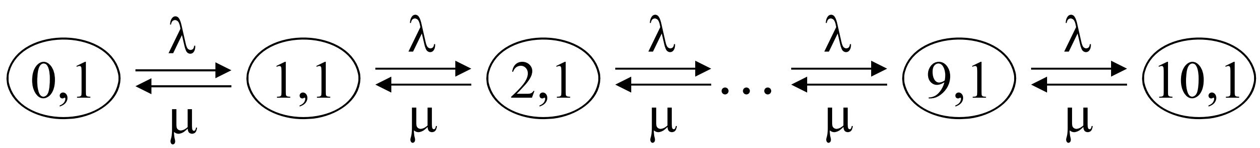
$\underbrace{\text{" the set of states (0,1) (1,1) .... (9,1) (10,1) whose transitions are job arrivals and departures will reach equilibrium between the times when a failure/repair occurs."}}$

$\Rightarrow$ isolate out the fast recurrent set of states from the 1-level model, analyze it for steady-state probabilities & replace it by a single state in the original model.

High-level:



Low-level:



Prob{idle server} = prob(high-model, 0,0)
+ prob(high-model, 1) * prob(low-model, 0,1)

Rejection prob = prob(high-model, 10,0)
+ prob(high-model, 1) * prob(low-model, 10,1)