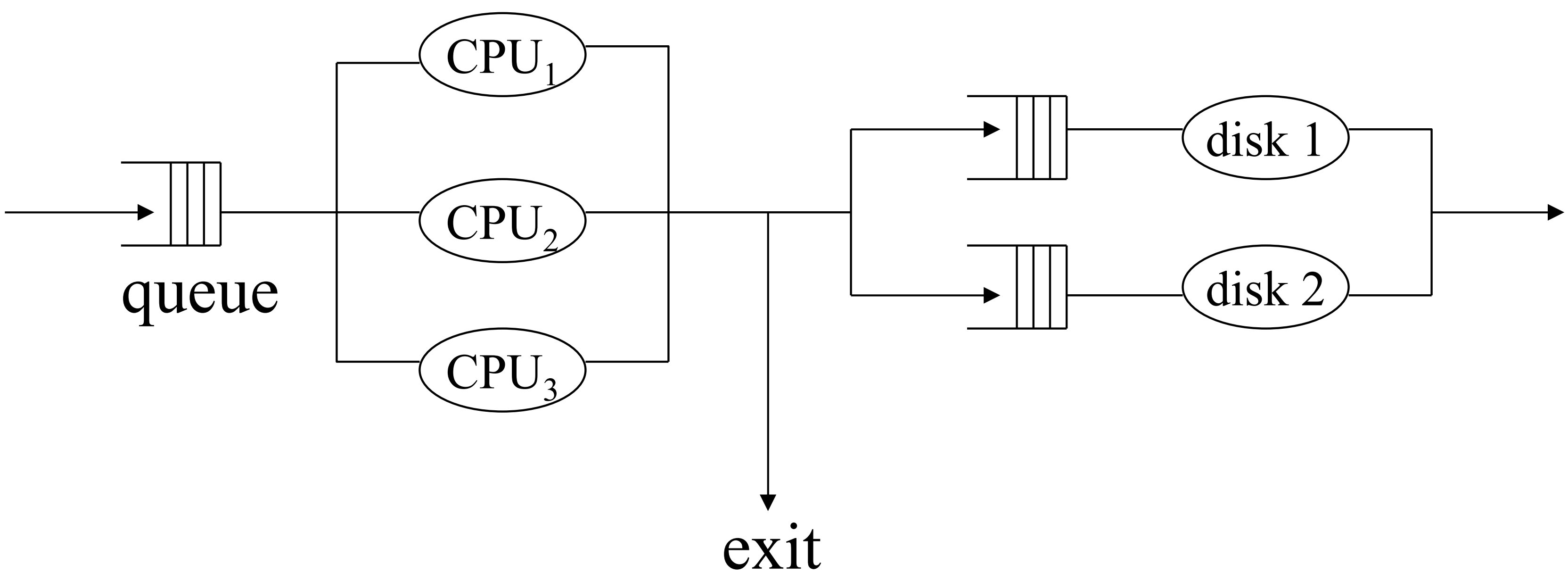
# Chap 5: Product-Form Queuing Networks (QN)

Entities: 1) service centers — with different service disciplines
2) customers (jobs) — single class
— multiple classes
(each/w a different workload)
3) links connecting service centers



## Service disciplines

1) FCFS
2) Priority — can be preemptive or non-preemptive
3) Round Robin (RR) — time-slot based
4) Processor Sharing (PS) — the server's capability is equally divided among all jobs
5) Last-Come-First-Serve Preemptive Resume (LCFSPR) — stack push-pop style

## Open vs. Closed QNM

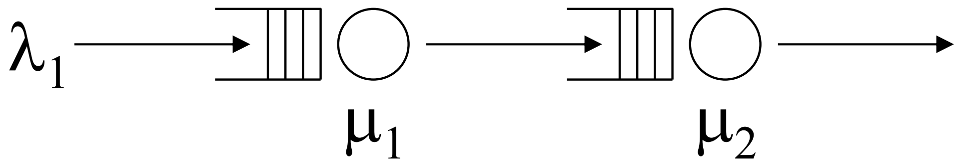
Open: customers arrive from an external source, spend time in the system & finally depart.

Closed: # of customers circulating among the service centers is a constant, i.e., no external source of jobs & no departure.

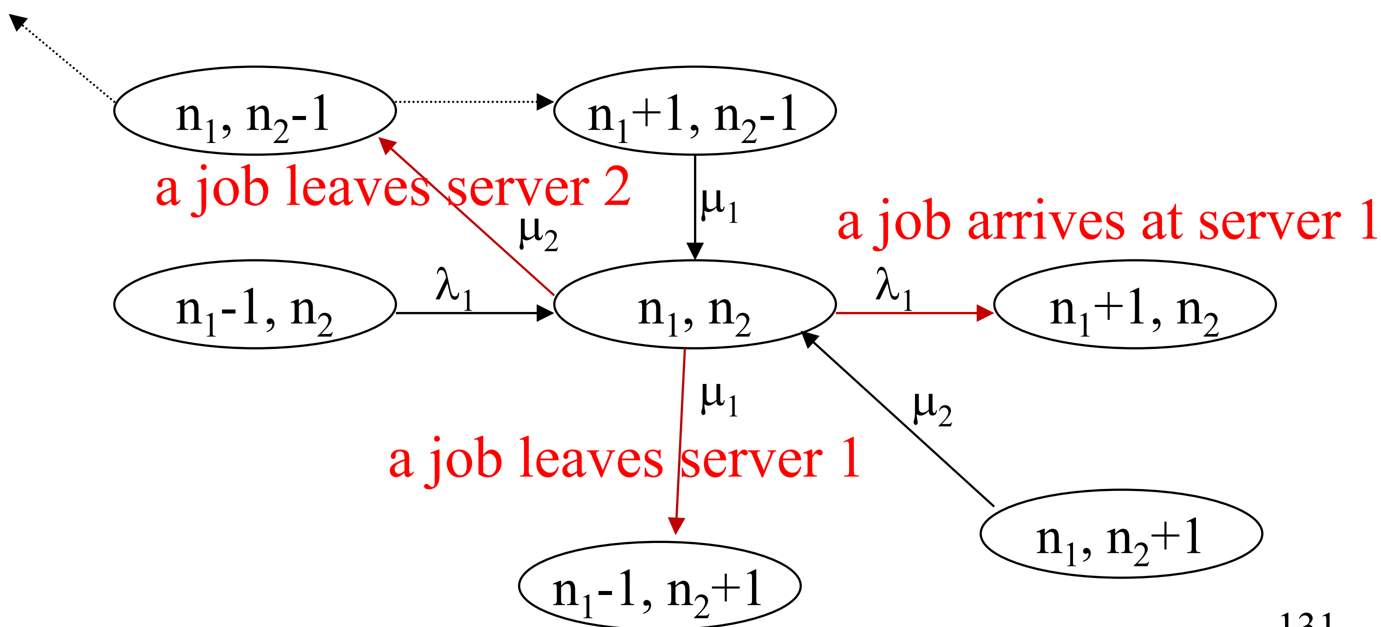
What is a "product-form" solution for a QNM?

- The joint probability of the queue sizes in the network is a product of the probabilities of queue sizes in individual service centers.

e.g., a tandem queuing network with 2 servers




Markov Model



By solving the steady-state global balance equations (one for each state), it can be shown that:

$$\underbrace{P(n_1, n_2)}_{\text{Prob. that the system has } n_1 \text{ at server 1 } \& \ n_2 \text{ at server 2}} = (1-\rho_1)\rho_1^{\,n_1}(1-\rho_2)\rho_2^{\,n_2} \quad \text{where } \rho_1 = \frac{\lambda_1}{\mu_1} \ \& \ \rho_2 = \frac{\lambda_1}{\mu_2}$$

$$= P(n_1) \cdot P(n_2)$$

For the 1st M/M/1 queue: $\lambda_1$, $\mu_1$

For the 2nd M/M/1 queue: $\lambda_1$, $\mu_2$

The joint population probability that there are $n_1$ jobs at server 1 & $n_2$ jobs at server 2 is the product of the population probabilities for two individual M/M/1 queues.

# Product-Form Queuing Networks

A QN is said to have a product-form solution if

$$\boxed{P(n_1, n_2, \ldots n_J) = \prod_{j=1}^{J} P_j\,(n_j)}$$

$P_j(n_j)$ is a function only of the $j$-th center

This is true when the following characteristics hold (p. 93, text):

1. The routing of customers from one service center to the next must be history independent, i.e., memory less (or Markovian).
2. The queuing disciplines may be FCFS, PS (Processor Sharing), IS (Infinite Server) or LCFSPR (Last Come First Serve with Preemptive-Resume)
3. For an FCFS center, the service time distribution must be exponential; for other servers, the service time distribution does not have to be exponential but must be differentiable (w. r. t. time)
4. A product-form network may have multiple chains (multiple classes) of jobs and may be <u>open</u> with respect to some chains of jobs and <u>closed</u> with respect to others. External arrivals for all open chains must be Poisson.

# Open product-form QNs:

All jobs arrive from an external source & depart to a sink.

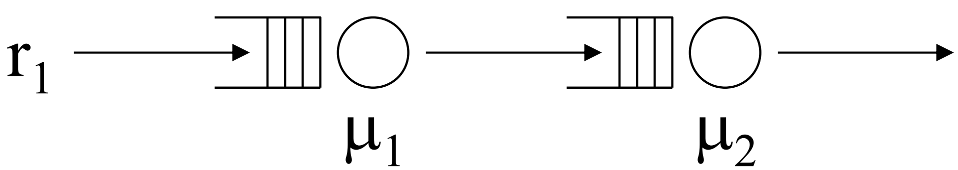
Arrival rate to j:

$$\lambda_j = r_j + \underbrace{\sum_{k=1}^{J}}_{\text{From other centers in the network}} \lambda_k \, q_{kj}$$

From the external source

$q_{kj}$ : prob {a customer goes to $j$ when it departs $k$}

Ex:




$$\lambda_1 = r_1$$
$$\lambda_2 = \lambda_1 \times 1$$

$$\therefore \ \lambda_1 = \lambda_2 = r_1$$

A general method to solve an open system QNM with a PF solution:

1) get input arrival rate for each center.
2) analyze each center separately.
3) get aggregate measures, e.g.,

$$n = \sum_k n_k$$

e.g. $r_1 = 0.5$, $\mu_1 = 1$ & $\mu_2 = 2$ $\Rightarrow$ $\rho_1 = \dfrac{\lambda_1}{\mu_1} = 0.5, \rho_2 = \dfrac{\lambda_2}{\mu_2} = \dfrac{0.5}{2} = 0.25$

Evaluate each center independently

$$\overline{n_1} = \frac{\rho_1}{1-\rho_1} = \frac{0.5}{1-0.5} = 1;\ \overline{n_2} = \frac{\rho_2}{1-\rho_2} = \frac{0.25}{1-0.25} = \frac{1}{3}$$

$$R_1 = \frac{\overline{n_1}}{\lambda_1} = \frac{1}{0.5} = 2; \qquad R_2 = \frac{\overline{n_2}}{\lambda_2} = \frac{\frac{1}{3}}{0.5} = \frac{2}{3}$$

$\therefore$ time spent in the system by a customer:

$$1 + \underbrace{1}_{\text{Waiting time at center 1}} + \underbrace{\frac{1}{6}}_{\text{Waiting time at center 2}} + \frac{1}{2} = {}^{8}/_{3} \text{ time units}$$

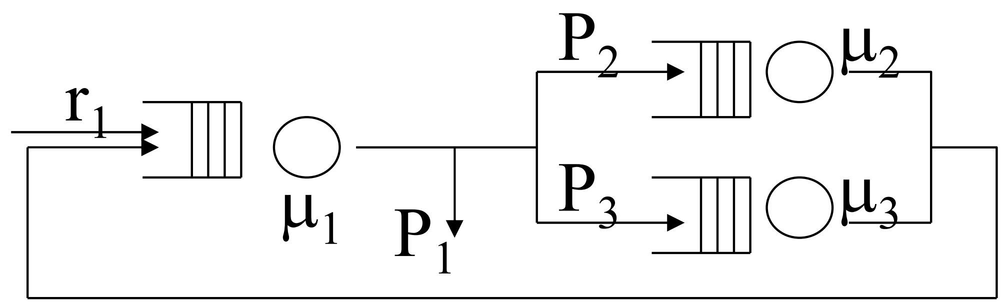
Waiting time at center 1,

$$\overline{w_1} = R_1 - {}^{1}/_{\mu_1} = 2 - {}^{1}/_{1} = 1$$

Waiting time at center 2,

$$\overline{w_2} = R_2 - {}^{1}/_{\mu_2} = \frac{2}{3} - \frac{1}{2} = \frac{1}{6}$$

## Ex: open product-form QNM with feedback




$$\left.\begin{aligned}\lambda_1 &= r_1 + \lambda_2 + \lambda_3 \\ \lambda_2 &= \lambda_1 P_2 \\ \lambda_3 &= \lambda_1 P_3\end{aligned}\right\}$$

$$\therefore \lambda_1 = r1 + \lambda_1 P_2 + \lambda_1 P_3$$

$$\text{or } \lambda_1 = \frac{r_1}{1 - P_2 - P_3} = \frac{r_1}{P_1}$$

Q: X, n, R?

$$X = \lambda_1 P_1 = (r_1 / P_1) \times P_1 = r_1$$

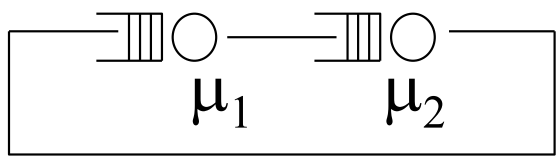
$$n = n1+n2+n3$$

$$R = n/X$$

# Closed Product-Form Networks

A network with a set of jobs circulating indefinitely or a network in which a job leaving the network will be replaced instantly by a statistically identical new job. e.g.,

N = 4




In general, consider a network with J service centers serving N jobs:

**Visit count to center j**

$$\left\{ v_j = \sum_{k=1}^{J} v_k * \underbrace{q_{kj}} \right.$$

probability that a job leaving center k moves to center j

A particular center's visit count is set to one based on the model's physical meaning.

Solution Technique: Mean Value Analysis Algorithm — it yields the average values of performance measures.

In a closed system with N jobs, when a job arrives, it actually sees only (N-1) jobs distributed in the system.

Notation:

$\mu_{\mathrm{j}}$: service rate at center j

$\overline{n_j}(k)$: average # of jobs at center j when k jobs are in the system

$\overline{r_j}(k)$: average response time of a job at center j
when there are k jobs in the system

$\overline{T}(k)$ : system throughput

$\overline{t_j}(k)$: throughput at center j

# Formulas:

$\overline{r_j}(k)$ is estimated just like in M/M/1 except that population is one less

$$\overline{r_j}(k) = \frac{1}{\mu_j}\left(1 + \overline{n_j}(k-1)\right)$$

$$\overline{T}(k) = \frac{k}{\overline{R}} = \frac{k}{\sum_{j=1}^{J} v_j * \overline{r_j}(k)} \quad \text{by Little's Law}$$

$$\overline{t_j}(k) = v_j * \overline{T}(k)$$

$$\overline{n_j}(k) = \overline{t_j}(k) * \overline{r_j}(k) \quad \text{by Little's Law}$$

# Recursion:

Given

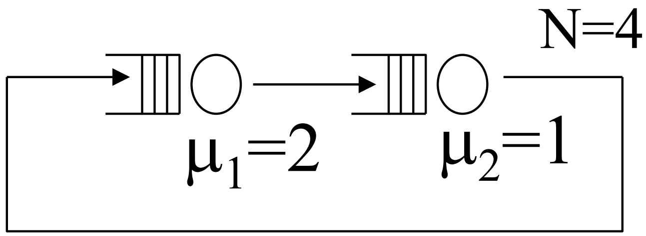
$\left.\begin{matrix}\mu_j \\ v_j\end{matrix}\right\}$ *for all j's*

$$\overset{k=0}{n_j(0) = 0} \rightarrow \overset{k=1}{r_j(k)} \rightarrow \overline{R} = \sum_{j=1}^{J} v_j \overline{r_j}(k) \rightarrow \overline{T}(k) \rightarrow \overline{t_j}(k) \rightarrow \overline{n_j}(k)$$

*k=k+1* until *k=N*

A particular center's visit count is set to one based on the model's physical meaning.

e.g.,




Set $v_1$ to 1 then $v_2 = 1$
(∵ relative visit count is the same for both centers in this example)

*P.100, text*

k=0: Starting with $\overline{n_1}(0) = 0;\ \overline{n_2}(0) = 0$

k=1:

$$\overline{r_1}(1) = \frac{1}{2}(1+0) = \frac{1}{2}$$

$$\overline{r_2}(1) = \frac{1}{1}(1+0) = 1$$

$$\overline{R}(1) = \frac{1}{2} + 1 = \frac{3}{2}$$

$$\overline{T}(1) = \frac{1}{\frac{3}{2}} = \frac{2}{3} \text{ (Little's Law)}$$

$$\overline{t_1}(1) = \frac{2}{3} \ \&\ \overline{t_2}(1) = \frac{2}{3} \ (\because\ v_1 = v_2 = 1)$$

$$\overline{n_1}(1) = \frac{2}{3} * \frac{1}{2} = \frac{1}{3}$$

$$\overline{n_2}(1) = \frac{2}{3} * 1 = \frac{2}{3}$$

k=2:

$$\overline{r_1}(2) = \frac{1}{2}\left(1 + \frac{1}{3}\right) = \frac{2}{3}$$

$$\overline{r_2}(2) = \frac{1}{1}\left(1 + \frac{2}{3}\right) = \frac{5}{3}$$

$$\overline{R}(2) = \frac{2}{3} + \frac{5}{3} = \frac{7}{3}$$

$$\overline{T}(2) = \frac{2}{\frac{7}{3}} = \frac{6}{7}$$

$$\overline{t_1}(2) = \frac{6}{7} \ \&\ \overline{t_2}(2) = \frac{6}{7}$$

$$\overline{n_1}(2) = \frac{6}{7} * \frac{2}{3} = \frac{4}{7}$$

$$\overline{n_2}(2) = \frac{6}{7} * \frac{5}{3} = \frac{10}{7}$$

(last iteration)

$$k=3\quad\left\{\begin{aligned}
\overline{r_1}(3) &= \frac{1}{2}\left(1+\frac{4}{7}\right)=\frac{11}{14}\\
\overline{r_2}(3) &= \frac{1}{1}\left(1+\frac{10}{7}\right)=\frac{17}{7}\\
\overline{R}(3) &= \frac{11}{14}+\frac{17}{7}=\frac{45}{14}\\
\overline{T}(3) &= \frac{3}{45/_{14}}=\frac{14}{15}\\
\overline{t_1}(3) &= \overline{t_2}(3)=\frac{14}{15}\\
\overline{n_1}(3) &= \frac{14}{15}*\frac{11}{14}=\frac{11}{15}\\
\overline{n_2}(3) &= \frac{14}{15}*\frac{17}{7}=\frac{34}{15}
\end{aligned}\right.$$

$$k=4\quad\left\{\begin{aligned}
\overline{r_1}(4) &= \frac{1}{2}\left(1+\frac{11}{15}\right)=\frac{13}{15}\\
\overline{r_2}(4) &= \frac{1}{1}\left(1+\frac{34}{15}\right)=\frac{49}{15}\\
\overline{R}(4) &= \frac{13}{15}+\frac{49}{15}=\boxed{\frac{62}{15}}\\
\overline{T}(4) &= \frac{4}{62/_{15}}=\frac{60}{62}=\boxed{\frac{30}{31}}\\
\overline{t_1}(4) &= \overline{t_2}(4)=\frac{30}{31}\\
\overline{n_1}(4) &= \frac{30}{31}*\frac{13}{15}=\frac{26}{31}\\
\overline{n_2}(4) &= \frac{30}{31}*\frac{49}{15}=\frac{98}{31}
\end{aligned}\right.$$

# Using sharpe to solve closed QNMs

Single-chain Multiple-chain

There are 6 types of service centers in a closed QNM which can be specified in a sharpe program:

Q: how to calculate $r_j(k)$ for an IS center?

1. FCFS — Syntax: station-name fcfs rate
2. IS — Syntax: station-name is rate — there are infinite # of servers in the center
3. MS — Syntax: station-name ms #servers rate — there are multiple servers in the center, each with the identical service rate
4. LCFSPR — Syntax: station-name lcfspr rate
5. PS — Syntax: station-name ps rate — all n jobs present at the center share one server with each job seeing the server speed reduced by a factor of n

   Automatically computed by sharpe

6. LDS (Load-dependent server) — Syntax: station-name lds rate1, rate2, …

— all jobs at the center again share one server but the service rate of the server is load dependent (i.e., depending on the # of jobs present in the center)

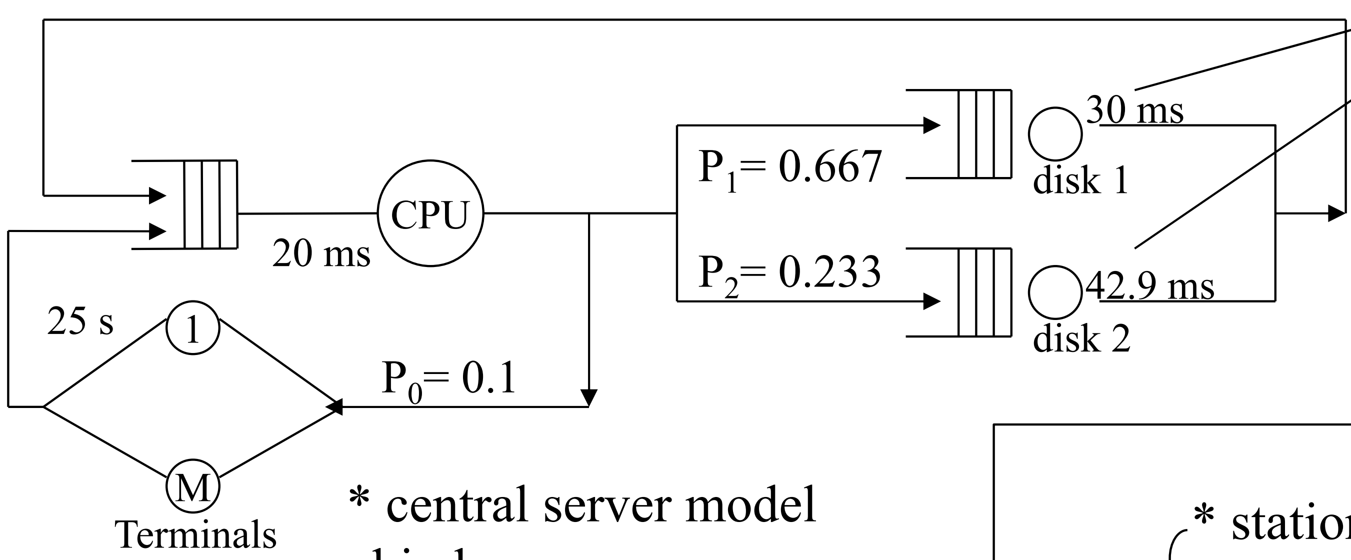
Must be specified in the sharpe code

*B.4.6. P.361 text*

Program structure for defining a <u>single-chain</u> (or single class) product-form queuing network model (for a closed system)

<u>pfqn</u> {(para-list)}
* section 1: station-to-station probabilities
< station-name station-name expression>
<u>end</u>
* section 2: station types & parameters
< station-name station-type expression, …>
<u>end</u>
* section 3: number of customers per chain (or per class)
< chain-name expression>
<u>end</u>

# e.g., *p.222 & p.102*




Service time per visit

These are rate parameters

Section 1

```
* central server model
bind
        P1  0.667
        P2  0.233
end
pfqn        csm
CPU         disk1        P1
CPU         disk2        P2
CPU         terminals    1-P1-P2
terminals   CPU    1
disk1       CPU    1
disk2       CPU    1
end
```

Section 2

```
* station types & parameters
terminals   IS      1/25
CPU         fcfs    1000/20
disk1       fcfs    1000/30
disk2       fcfs    1000/42.9
end
```

Section 3

```
* number of jobs
chain1   M
end
```

*For other service disciplines see p.362, text* *qlength returns per-center population*

```
* (continued)
* reporting per center (CPU) measures
loop    i, 2, 10, 2
   bind    M    i
   expr    tput (csm, CPU)
   expr    util (csm, CPU)
   expr    qlength (csm, CPU)
   expr    rtime (csm, CPU)
end
* calculate the system response time
* by applying Little's Law
```

$* \; R = \overline{n}/x,$ where

$* \; \overline{n}$: *population* in the central system

$* \; x$: *throughput* of the central system

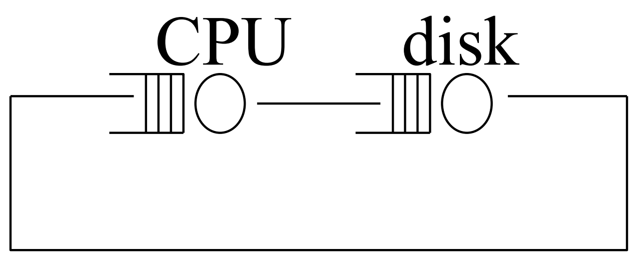
```
func      x() \
          tput (csm, CPU) * (1-P_1-P_2)
func      nbar() \
          qlength (csm, CPU) + \
          qlength (csm, disk1) + \
          qlength (csm, disk2)
   bind   M          10
* calculate the average response time per
* terminal user once it enters the central
* system when M=10 in the terminal center
   expr   nbar()/x()
end       /* end the entire program */
```

***Apply MVA to this closed system and set the visit count to 1 for the terminals center. You should get the same output. Note: set $r_{terminals}$ = 25s because it is a IS center.***

## Program structure for defining a multiple-chain product-form queuing network model (for a closed system)

```
mpfqn  {(parameter-list)}
* section 1: station to station probabilities for each chain.
<chain  chain-name
          <station-name  station-name  expression>
                          :
 end>
end
* section 2: station types & parameters
<<station-name  station-type  expression, …>
 <chain-name  expression, …>
                  :
 end>
end
* section 3: number of jobs per chain
<chain-name  expression>
end
```

Example:



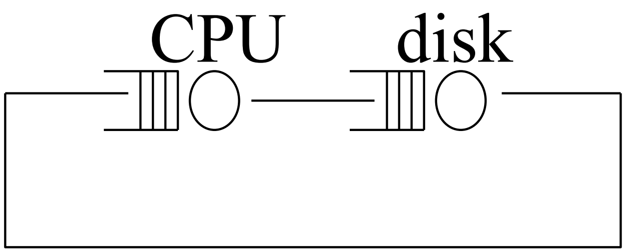

Two classes:
chain A: 2 jobs
chain B: 1 job

Service time (service demand) $D_{A,CPU} = 2; D_{A,disk} = 1$
$D_{B,CPU} = 3; D_{B,disk} = 2$

Performance measures of interest?

— Response time of a job (system)
— Response time of a class A job (per class)
— Response time of a class B job in the CPU center (per center per class)
— Throughput of the CPU center for class A jobs (per center per class)
— Utilization of the CPU center for class B jobs (per center per class)

* An example of using sharpe for solving a multiple-class product form queuing network
* Two classes: A and B; assume visit count is 1 for each center
* number of jobs: (2A, 1B)
* number of stations: 2 -- cpu and disk
* $D_{A,cpu} = 2$
* $D_{A,disk} = 1$
* $D_{B,cpu} = 3$
* $D_{B,disk} = 2$
*want to know $R_A$, $X_A$,($n_{A,cpu}$),($U_{A,cpu}$)




$D_{A,CPU} = 2$; $D_{A,disk} = 1$
$D_{B,CPU} = 3$; $D_{B,disk} = 2$

mpfqn simple
* section1: station to station transition probabilities

```
chain A
cpu disk 1
disk cpu 1
end
chain B
cpu disk 1
disk cpu 1
end
end
```

```
*section 2: station types and parameters
cpu ps
A 1/2
B 1/3
end
disk ps
A 1/1
B 1/2
end
end
*section 3: number of customers in each
*              chain
A 2
B 1
end
```

Per-center per class-> per class measures

Summation applies to population only

Per-class measures -> system measures

Summation applies to population and throughput

Once you know population and throughput You can know the response time by Little's Law

*In general, need to calculate $R_A=n_A/X_A$.

*But for the simple system here, we

*can calculate $R_A = R_{A,cpu} + R_{A,disk}$

```
expr mrtime(simple,cpu,A)
     +mrtime(simple,disk,A)
```

*$X_A = X_{A,\ cpu}$ for this simple system

```
expr mtput(simple,cpu,A)
```

*population of class A at CPU: $n_{A,cpu}$

```
expr mqlength(simple,cpu,A)
```

*utilization of class A at CPU: $U_{A,cpu}$

```
expr mutil(simple,cpu,A)

end
-----------------------------------------------
Output:
mrtime(simple,cpu,A)
+mrtime(simple,disk,A):   6.3478e+00
-----------------------------------------
mtput(simple,cpu,A):   3.1507e-01
-----------------------------------------
mqlength(simple,cpu,A):   1.4795e+00
-----------------------------------------
mutil(simple,cpu,A):   6.3014e-01
```