King Fahd University of Petroleum & Minerals Computer Engineering Dept

CSE 642 – Computer Systems Performance

Term 041

Dr. Ashraf S. Hasan Mahmoud

Rm 22-148-3

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa

Time Reversed Processes

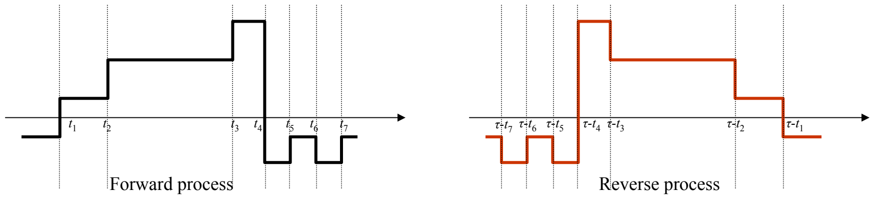
- **Refer to Leon Garcia's textbook section 8.5 for discussion of Time-Reversed Markov Chains**
- **Consider a continuous-time process X(t)**
- **Define the following process $X^r(t) = X(T-t)$, for an arbitrary T. For simplicity we set T to 0**
- **$X^r(t)$ is the reverse process for X(t)**

Time Reversed Processes – cont'd

- **Time reversibility – a process is time reversible if the following is true**

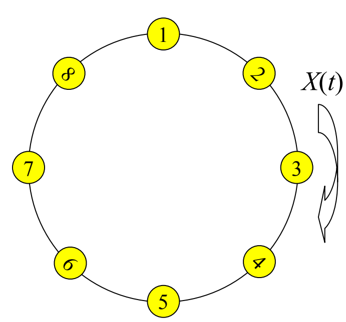
$$P(X(t_1)=i_1, X(t_2)=i_2, \cdots, X(t_m)=i_m)$$
$$= P(X(\tau - t_m)=i_m, X(\tau - t_{m-1})=i_{m-1}, \cdots, X(\tau - t_1)=i_1)$$

- **We say the process reversed in time has the same probabilistic properties as the forward process**



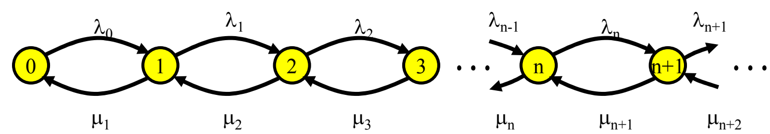
Time Reversed Processes – Observations

- **For a process to be reversible, it is not enough for the marginal probabilities for the forward and reverse processes to be equal.**
- **Example: consider the process X(t) depicted in figure. X(t) goes clockwise through states i→(i+1) mod 8. The probability of X(t) being in any of the state is 1/8. The same is true for the reverse process which travels counter clockwise. However, the process is clearly not reversible since state 2 can not follow state 1 in the reverse process, for example.**



Reversibility and Birth and Death Processes

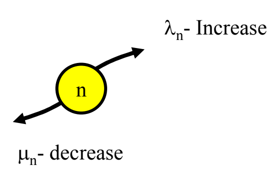
- **Review:**



- **Local balance equation: $P_n \lambda_n = P_{n+1} \mu_n$ – where P_n is the probability of being in state n**

Reversibility and Birth and Death Processes – cont'd

- **State = population equal to n**
  - **Time spends in state is exponentially distributed with mean $1/(\lambda_n+\mu_n)$**
  - **Probability of decrease (i.e. jumping to state n-1) = Probability of an departure occurring before an arrival**
  - **Probability of increase (i.e. jumping to state n+1) = Probability of an arrival occurring before an departure**
- **Show that:**
  - **Prob[Increase] = $\lambda_n/(\lambda_n+\mu_n)$**
  - **Prob[Increase] = $\mu_n/(\lambda_n+\mu_n)$**



Reversibility and Birth and Death Processes – cont'd

- **A birth-death process is reversible if and only if the local balance equations hold**

- **Proof: Refer to textbook – section 4.2.2**

Reversibility and Birth and Death Processes – Proof Part 1

- **Reversible birth-death process → Local balance equations hold**
- **Proof: Assume X(t) is a reversible BD process →**

$$P(X(t)=j, X(t+\delta) = k) = P(X(t)=k, X(t+\delta) = j)$$

Let $P_j = P(X(t)=j)$ and $P_k = P(X(t)=k)$, for $\delta \to 0$, we can assume k = j + 1 (i.e. one transition is possible), we can write:

$$P_j\, P(X(t+\delta) = j+1 \,/\, X(t) = j) = P_{j+1}\, P(X(t+\delta) = j \,/\, X(t) = j+1)$$

recognizing that,

$$\lim_{\delta \to 0} \frac{P(X(t+\delta) = j+1/X(t)=j)}{\delta} = \lambda_j$$

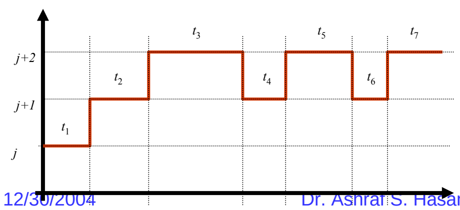
$$\lim_{\delta \to 0} \frac{P(X(t+\delta) = j/X(t)=j+1)}{\delta} = \mu_{j+1}$$

Therefore, $P_j\, \lambda_j = P_{j+1}\, \mu_{j+1}$

Reversibility and Birth and Death Processes – Proof Part 2

- **Local balance equations hold → Reversible birth-death process**
- **Proof:**
  Consider the sample path depicted of a birth-death process, the probability of the process taking this "exact path" is given by the following expression:

$$
\begin{aligned}
&P_j \times \frac{\lambda_j}{\lambda_j+\mu_j} \times (\lambda_j+\mu_j) \times e^{-(\lambda_j+\mu_j)t_1} dt_1 \times \frac{\lambda_{j+1}}{\lambda_{j+1}+\mu_{j+1}} \times (\lambda_{j+1}+\mu_{j+1}) \times e^{-(\lambda_{j+1}+\mu_{j+1})t_2} dt_2 \\
&\times \frac{\mu_{j+2}}{\lambda_{j+2}+\mu_{j+2}} \times (\lambda_{j+2}+\mu_{j+2}) \times e^{-(\lambda_{j+2}+\mu_{j+2})t_3} dt_3 \times \frac{\lambda_{j+1}}{\lambda_{j+1}+\mu_{j+1}} \times (\lambda_{j+1}+\mu_{j+1}) \times e^{-(\lambda_{j+1}+\mu_{j+1})t_4} dt_4 \\
&\times \frac{\mu_{j+2}}{\lambda_{j+2}+\mu_{j+2}} \times (\lambda_{j+2}+\mu_{j+2}) \times e^{-(\lambda_{j+2}+\mu_{j+2})t_5} dt_5 \times \frac{\lambda_{j+1}}{\lambda_{j+1}+\mu_{j+1}} \times (\lambda_{j+1}+\mu_{j+1}) \times e^{-(\lambda_{j+1}+\mu_{j+1})t_6} dt_6 \times e^{-(\lambda_{j+2}+\mu_{j+2})t_7}
\end{aligned}
$$



Note:
- P_j is the probability of starting from state j
- The term $(\lambda_j+\mu_j)\exp[-(\lambda_j+\mu_j)t_1]dt_1$ is probability density of the first interval
- $\exp[-(\lambda_{j+2}+\mu_{j+2})t_7]$ is the probability that the process remains in state j+2 for at least t_7

Reversibility and Birth and Death Processes – Proof Part 2 cont'd

- **The previous expression is simplified to be**

$$
\begin{aligned}
&P_j \lambda_j e^{-(\lambda_j+\mu_j)t_1} dt_1 \times \lambda_{j+1} e^{-(\lambda_{j+1}+\mu_{j+1})t_2} dt_2 \\
&\times \mu_{j+2} e^{-(\lambda_{j+2}+\mu_{j+2})t_3} dt_3 \times \lambda_{j+1} \times e^{-(\lambda_{j+1}+\mu_{j+1})t_4} dt_4 \\
&\times \mu_{j+2} e^{-(\lambda_{j+2}+\mu_{j+2})t_5} dt_5 \times \lambda_{j+1} e^{-(\lambda_{j+1}+\mu_{j+1})t_6} dt_6 \times e^{-(\lambda_{j+2}+\mu_{j+2})t_7}
\end{aligned}
$$

- **Using the local balance equation ($P_j \lambda_j = P_{j+1} \mu_j$, we can write:**

$$
P_j \lambda_j \lambda_{j+1} \mu_{j+2} \lambda_{j+1} \mu_{j+2} \lambda_{j+1} = P_{j+2} \mu_{j+2} \lambda_{j+1} \mu_{j+2} \lambda_{j+1} \mu_{j+2} \mu_{j+1}
$$

Reversibility and Birth and Death Processes – Proof Part 2 cont'd

- **Substitute the above equation in the former path evolution probability formula, we obtain**

$$
\begin{aligned}
&P_{j+2}\mu_{j+2}e^{-(\lambda_{j+2}+\mu_{j+2})t_1}dt_7 \times \lambda_{j+1}e^{-(\lambda_{j+1}+\mu_{j+1})t_6}dt_6 \\
&\times \mu_{j+2}e^{-(\lambda_{j+2}+\mu_{j+2})t_5}dt_5 \times \lambda_{j+1} \times e^{-(\lambda_{j+1}+\mu_{j+1})t_4}dt_4 \\
&\times \mu_{j+2}e^{-(\lambda_{j+2}+\mu_{j+2})t_3}dt_3 \times \mu_{j+1}e^{-(\lambda_{j+1}+\mu_{j+1})t_2}dt_2 \times e^{-(\lambda_j+\mu_j)t_1}
\end{aligned}
$$

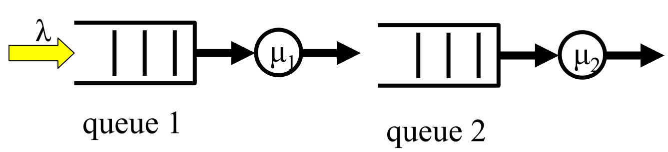
- **Which is the probability of the process starting with state j+2 and taking the same sample path in reverse**

- **➔ Process is reversible**

FeedForward Networks

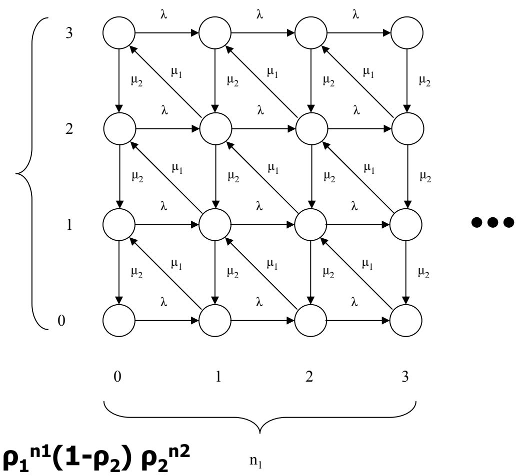
- **Consider the system depicted in figure where two are in tandem.**
- **How would the following system be analyzed?**

- **Assume infinite storage and a single server with exponential service time for each queue**



Two Queues in Tandem

- **We can use the state diagram to solve this problem:**
  - **State = (n_1, n_2) where n_i is the number of customers in i^{th} queue**
- **Refer to Garcia's textbook section 9.8 for solution of this two dimensional state diagram**



- **We can show**

$$P(N_1 = n_1, N_2 = n_2) = (1-\rho_1)\,\rho_1^{n1}(1-\rho_2)\,\rho_2^{n2}$$

Where $\rho_1 = \lambda_1/\mu_1$ and $\rho_2 = \lambda_2/\mu_2$

Two Queues in Tandem – cont'd

- **Since for a single queue,**

$$P(N = n) = (1-\rho_1)\,\rho_1^{n1}$$

Then it is clear that

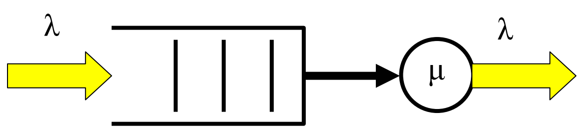
$$P(N_1 = n_1, N_2 = n_2) = P(N_1 = n_1)P(N_2 = n_2)$$

- **Therefore, the number of customers at queue 1 and the number of customers at queue 2 are independent random variables!!**

Burke's Theorem

- **Consider an M/M/1 or M/M/S or M/M/∞ queueing system at steady state with arrival rate λ, then**
  - **The departure process is Poisson with rate λ**
  - **At each time t, the number of customers in the system N(t) is *independent* of the sequence of departure times prior to t.**

M/M/1 example



Feedforward Networks – Example 1

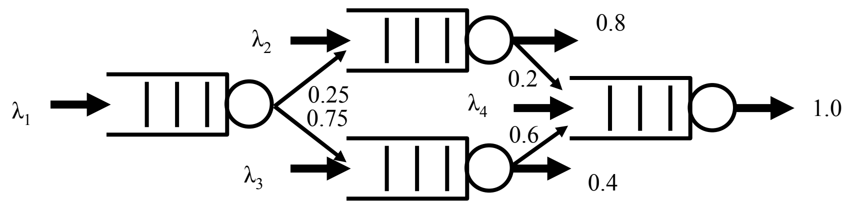
- **An application of Burk's theorem**
- **Example 1: M queues in tandem**

- **Direct extension to the two queues in tandem case**

$$P(Q_1(t)=k_1, Q_2(t)=k_2, \cdots, Q_M(t)=k_M) = \prod_{i=1}^{M}(1-\rho_i)\rho_i^{k_i}$$

Feedforward Networks – Example 2

- **Example2: feedforward acyclic networks (i.e. no feedback paths)**



- **Since joining and splitting of Poisson streams results in Poisson streams – Burks' theorem still applicable**
- **Solution key: deal with the individual queues after determining the total flow to the queue**

Traffic Equation and Routing Matrix

- **Consider a network of N queues, each having an independent exponential server and an infinite buffer**
- **External arrivals at each node – Poisson with rate λ_i**
- **Messages are routed probabilistically:**
  - **q_{ji}: i,j = 1, 2, 3, …, N is the probability of a message being routed from node j to node I**
  - **q_{jN+1}: is the probability of a message being routed outside the network**
  - **Note:** $\sum_{i=1}^{N+1} q_{ji} = 1$

Traffic Equation and Routing Matrix – cont'd

- **Let Λ_i: total flow into the i^{th} node**
- **Clearly, one can write**

$$\Lambda_i = \lambda_i + \sum_{j=1}^{N} q_{ji}\Lambda_j$$

- **The matrix version is**

$$[\Lambda_1 \quad \Lambda_2 \quad \cdots \quad \Lambda_N] = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_N] + [\Lambda_1 \quad \Lambda_2 \quad \cdots \quad \Lambda_N]\begin{bmatrix} 0 & q_{12} & \cdots & q_{1N} \\ q_{21} & 0 & \cdots & q_{2N} \\ \vdots & & 0 & \\ q_{N1} & q_{N2} & \cdots & 0 \end{bmatrix}$$

or

$$\Lambda = \lambda + \Lambda Q$$

Note: $q_{jj} = 0$ – no routing to same source node

Traffic Equation and Routing Matrix – cont'd

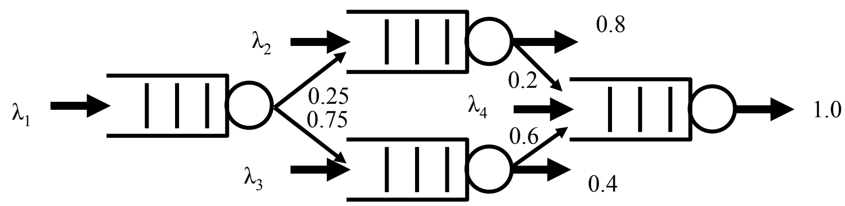
- **Normally the inputs and the routing matrix are known, the total flow to each node can be found using**

$$\Lambda = \lambda[I - Q]^{-1}$$

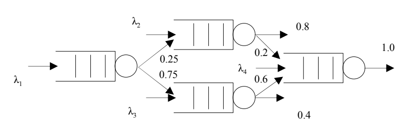
where *I* is the NxN identity matrix

Example:

- **<u>Problem</u>: Consider the network of queue depicted in figure. If the arrival rates are given by λ = [2.0, 1.0, 0.5, 3.0], and the service rates are μ = [4.0, 6.0, 11.0, 9.9],**
  - **a) compute the total flow into each node**
  - **b) Find the joint pmf for number of customers in queues**



Example:



- **<u>Solution</u>:**

a) The routing matrix for the network is given by

$$Q = \begin{bmatrix} 0 & 0.25 & 0.75 & 0 \\ 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0.6 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore, total flows are given by

Λ = [2.0, 1.5, 2.0, 4.5]

b) The loads for the queues are given by

$R = \Lambda./\mu$ (./ is the element-by-element division – Matlab notation)

= [1/2 , 1/4, 2/11, 5/11]

The joint pmf for the number of customers is given by

$$P(Q_1 = k_1, Q_2 = k_2, Q_3 = k_3, Q_4 = k_4) = \prod_{i=1}^{4} (1 - R_i) R_i^{k_i}$$
$$= (81/484)(1/2)^{k_1} (1/4)^{k_2} (2/11)^{k_3} (5/11)^{k_4}$$

Open Network – Flows within Feedback Paths

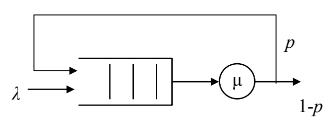
- **Open networks – at least one external source of arrivals ➔ there must be a flow to outside network (exit path)**
  - **i.e. sum q_{ki} < 1 for at least one k = 1, 2, …, N**

- **Feedback to a node -**

Example: M/M/1 queue with Feedback

- **Problem: Consider the following system – Find the pmf for number of customers in the system.**

- **Solution:**



$\Lambda = \lambda + p\Lambda$ ➔ $\Lambda = \lambda/(1-p)$
Therefore, traffic load, R is given by
$R = \Lambda/\mu = \lambda/[\mu(1-p)]$
Prob(N = k) = $(1-R)R^k$ k=0, 1, 2, …

Note R < 1 ➔ $\lambda/[\mu(1-p)] < 1$ or $\lambda < \mu(1-p)$ – this imposes a limit on the maximum arrival rate

What is the average number of customer visits to the queue?

$E[N] = R/(1-R)$
$E[T] = E[N] / \lambda$ - direct application of Little's formula

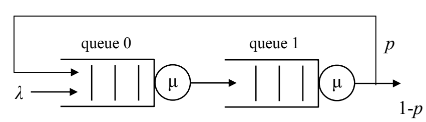
For a general solution of an M/G/1 with Bernoulli feedback check: L. Takács, "A Single-Server Queue with Feedback," Bell Technical Journal, March 1963, pp. 505-519.

Assignment # 3

- **Use example 9.23 of Leon Garcia's textbook to do the following:**
  - **(a) Using the theoretical analysis supplied in the solved example in the textbook:**
    - **Plot the average total number of customers in network versus the external arrival rate**
    - **Plot the average end-to-end delay of a customer versus the external arrival rate**
  - **(b) Develop an opnet simulation model and produce simulation results and compare them against those obtained in part (a)**
    - **Produce comparative curves similar to those found on slide 39.**
  - **For part (a) and (b) use p = 0.9 and p = 0.6 (note p is the probability of exiting the network from queue 1)**
- **Deadline: Sunday January 2nd, 2005 – class time**

Feedback Violates the Poisson Departure process

- **Let us examine the following system***



- **<u>Solution:</u> The analytic solution for the depicted system is as follows:**

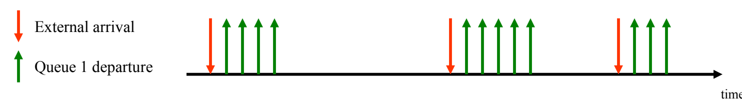
$\Lambda = \lambda + p\Lambda \rightarrow \Lambda = \lambda/(1-p)$
Therefore, traffic load, R is given by
$R_0 = R_1 = R = \Lambda/\mu = \lambda/[\mu(1-p)]$
$\text{Prob}(N_0 = k) = \text{Prob}(N_1 = k) = (1-R)R^k \quad k=0, 1, 2, \ldots$

Note $R < 1 \rightarrow \lambda/[\mu(1-p)] < 1$ or $\lambda < \mu(1-p)$
$E[N_0] = E[N_1] = R/(1-R) = \lambda/[\mu(1-p) - \lambda]$
$E[N] = E[N_0] + E[N_1] = 2R/(1-R) = 2\lambda/[\mu(1-p) - \lambda]$
End-to-end delay for a customer is computed as
$E[T] = E[N] / \lambda = 2/[\mu(1-p) - \lambda]$

*I can show the same behavior using the single-server queue system considered in the last example – but I am using the system proposed the textbook to give another example on opnet modeling

Feedback Violates the Poisson Departure process – cont'd

- **To depict the violation of the Poisson arrival/departure process**
  - **Assume a very low external arrival rate λ – say 1 packet every 2 hours – i.e. mean interarrival time of 7200 seconds**
  - **Assume a very small mean service time 1/μ – say 10^{-9} second**
  - **Let p = 0.999**
- **This setting translates the following:**
  - **One customer arrives – the next arrival is 1000s of seconds away on average**
  - **The customer is TRAPPED in the system circulating (since p ≈ 1**
  - **So if we monitor the customer departures of queue 0 or 1 (prior to the feedback branching) – we expect to see departure bursts**



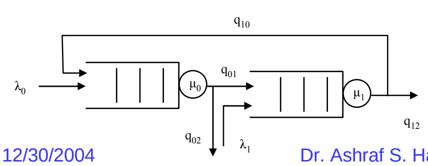
Proof Of the Product Form: Two-Node Network

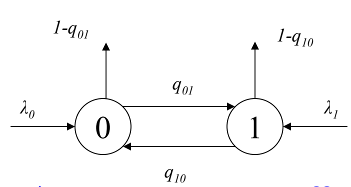
- **Consider the two queues network depicted in figure**
- **The total flow equations are given by**

$$[\Lambda_0 \quad \Lambda_1] = [\lambda_0 \quad \lambda_1] + [\Lambda_0 \quad \Lambda_1]\begin{bmatrix} 0 & q_{01} \\ q_{10} & 0 \end{bmatrix}$$

- **System state: (k_0, k_1) where k_0 is number of customers in queue 0 while k_1 is number of customers in queue 1**

- **Define $P(Q_0(t)=k_0, Q_1(t)=k_1) = P(k_0, k_1; t)$ – as the probability of k_i customers in the respective queue at time t.**





Proof Of the Product Form: Two-Node Network – cont'd

- **The Kolmogorov differential equation for $P(k_0, k_1; t)$:**

$$\begin{aligned}P(k_0, k_1; t+\delta) = {} & P(k_0-1, k_1; t)\lambda_0\delta + P(k_0, k_1-1; t)\lambda_1\delta \\ & + P(k_0+1, k_1; t)\mu_0(1-q_{01})\delta \\ & + P(k_0, k_1+1; t)\mu_1(1-q_{10})\delta \\ & + P(k_0+1, k_1-1; t)\mu_0 q_{01}\delta \\ & + P(k_0-1, k_1+1; t)\mu_1 q_{10}\delta \\ & + P(k_0, k_1; t)(1- (\lambda_0+\lambda_1+\mu_0+\mu_1))\delta\end{aligned}$$

- **The terms on the RHS:**
  - First two terms – arrivals to either queues
  - Second pair – departures from system
  - Third pair – transfers between queues
  - Final term – no arrivals, departures, or transfers
- **The Kolmogorov D.E is given by**

$$\begin{aligned}dP(k_0, k_1; t)/dt = {} & P(k_0-1, k_1; t)\lambda_0 + P(k_0, k_1-1; t)\lambda_1 \\ & + P(k_0+1, k_1; t)\mu_0(1-q_{01}) \\ & + P(k_0, k_1+1; t)\mu_1(1-q_{10}) \\ & + P(k_0+1, k_1-1; t)\mu_0 q_{01} \\ & + P(k_0-1, k_1+1; t)\mu_1 q_{10} \\ & - P(k_0, k_1; t)(\lambda_0+\lambda_1+\mu_0+\mu_1)\end{aligned}$$

- **At steady state $dP(k_0, k_1; t)/dt = 0$**

Proof Of the Product Form: Two-Node Network – cont'd

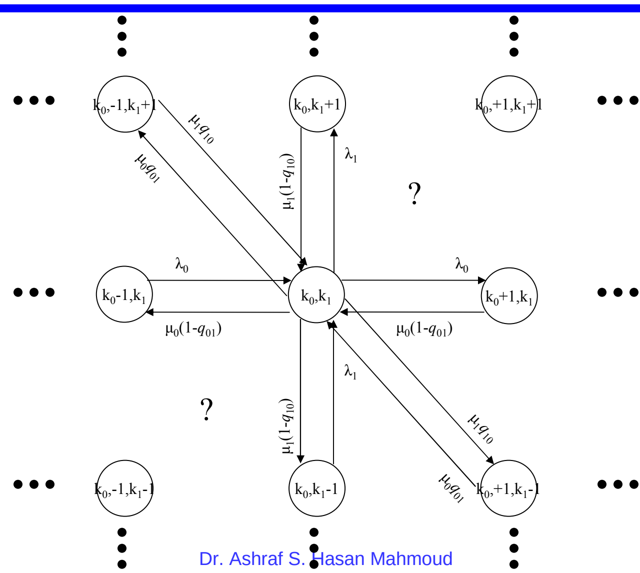
- **The steady state probabilities are then given by:**

$$\begin{aligned}P(k_0, k_1)(\lambda_0+\lambda_1+\mu_0+\mu_1) = {} & P(k_0-1, k_1)\lambda_0 + P(k_0, k_1-1)\lambda_1 \\ & + P(k_0+1, k_1)\mu_0(1-q_{01}) \\ & + P(k_0, k_1+1)\mu_1(1-q_{10}) \\ & + P(k_0+1, k_1-1)\mu_0 q_{01} \\ & + P(k_0-1, k_1+1)\mu_1 q_{10} \quad \forall\, k_0,k_1 \ge 0\end{aligned}$$

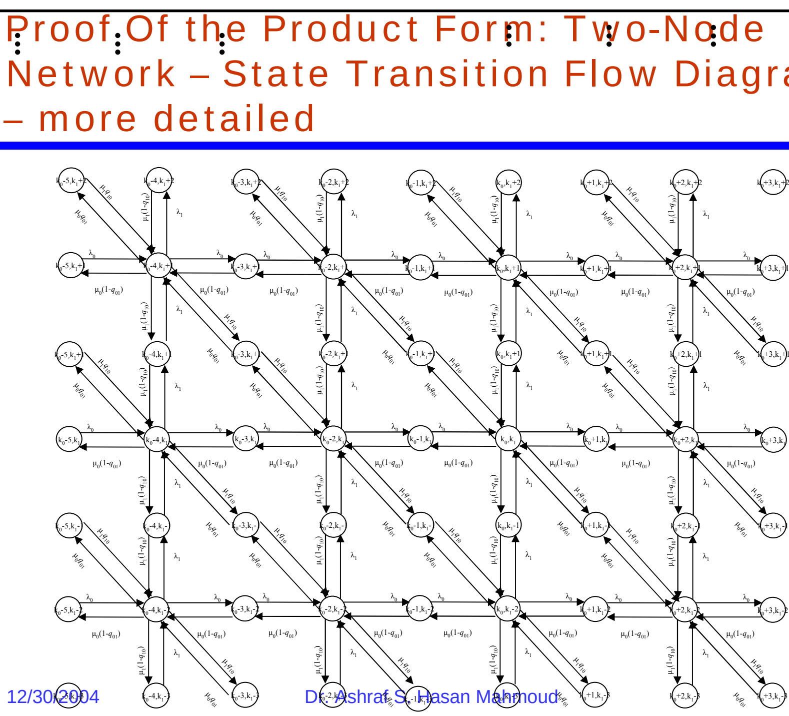
- **Note that the set of equilibrium equations stated above together with the normalizing condition $\sum_{\forall k_0,k_1} P(k_0,k_1) = 1$ can be solved to obtain the complete pmf – however, as we will show, the closed form solution turns out to be simple**

- **The state transition flow diagram is shown on the next slide**

Proof Of the Product Form: Two-Node Network – State Transition Flow Diagram



Proof Of the Product Form: Two-Node Network – State Transition Flow Diagram – more detailed



Proof Of the Product Form: Two-Node Network – Rewriting the equations In Terms of Total Flow

- **Rewriting the pervious equations in terms of the total flows Λ_0 and Λ_1 results in:**

$$
\begin{aligned}
&P(k_0, k_1)(\Lambda_0+\Lambda_1+\mu_0+\mu_1) \\
&+ P(k_0-1, k_1)q_{10}\Lambda_1 + P(k_0, k_1-1)\, q_{01}\Lambda_0 \\
&+ P(k_0+1, k_1)\mu_0 q_{01} + P(k_0, k_1+1)\mu_1 q_{10} \\
&\qquad = P(k_0+1, k_1)\mu_0 + P(k_0, k_1+1)\mu_1 \\
&\qquad + P(k_0-1, k_1)\, \Lambda_0 + P(k_0, k_1-1)\, \Lambda_1 \\
&\qquad + P(k_0+1, k_1-1)\mu_0 q_{01} + P(k_0-1, k_1+1)\mu_1 q_{10} \\
&\qquad + P(k_0, k_1)(q_{01}\Lambda_0+q_{10}\Lambda_1) \quad \forall\, k_0,k_1 \ge 0
\end{aligned}
$$

- **Solving the above equations, yields**

$P(k_0, k_1)\Lambda_0 = P(k_0+1, k_1)\mu_0$

$P(k_0, k_1+1)\mu_1 = P(k_0, k_1)\Lambda_1$

- **These can be written as**

$P(k_0+1, k_1) = \rho_0\, P(k_0, k_1)$

$P(k_0, k_1+1) = \rho_1\, P(k_0, k_1)$

where ρ_i is given by Λ_i/μ_i

Proof Of the Product Form: Two-Node Network – Final Solution

- **Therefore, solving iteratively and using the normalizing condition, one can write**

$$P(k_0,k_1) = (1-\rho_0)(1-\rho_1)\, \rho_0^{k0}\, \rho_1^{k1} \qquad k_0,k_1 = 0, 1, \ldots$$

i.e. the product form applies.

Example: Two-Node Network

- **Problem: Let λ = [2.0, 1.0], and the service rates are μ = [15.625, 3.75] – Let the routing parameters q01 = 0.4 and q10 = 0.5.**
  - **A) compute the total flow into each queue**
  - **B) Compute the traffic utilization of each queue – write an expression for the joint pmf of number of customers in the network**
  - **C) Compute the average number of messages in node 0 and the average number of messages in node 1**
  - **D) Calculate the average end-to-end delay for a customer**

Example: Two-Node Network – cont'd

- **Solution:**

A) The total flow is found by solving the following set of equations:

$$[\Lambda_0\ \Lambda_1] = [\lambda_0\ \lambda_1] + [\Lambda_0\ \Lambda_1]\begin{bmatrix} 0 & q01 \\ q10 & 0 \end{bmatrix}$$

Therefore, $[\Lambda_0\ \Lambda_1]$ = [3.125 2.25]

B) Traffic Utilization: $R = \Lambda./\mu$ = [0.2 0.6]

$$P(k_0, k_1) = 0.32(0.2)^{k0}(0.6)^{k1} \text{ for } k0, k1 = 0, 1, \ldots$$

C) $E[N_0] = R_0/(1 - R_0) = 0.25$

$E[N_1] = R_1/(1 - R_1) = 1.5$

Note that $E[N] = E[N_0] + E[N_1] = 1.75$

D) $E[T] = E[N]/(\lambda_0+\lambda_1) = 0.583$ **seconds**

N-Node Open Jackson Networks – Problem Specification

- **Consider an N-Node open network that is characterized by**
  - **Probabilistic routing matrix Q = {q_{ij}}**
  - **Set of external flows λ_i; i=1, 2, …, N - Poisson processes**
  - **Infinite storage at each node**
- **Assume S_i servers at each node i – each having exponentially distributed service time**
  - **Departure rate from state k_i (i.e. k_i customers in node i) is equal to**

$$\mu_i d(k_i) = \begin{cases} \mu_i k_i & k_i \le S_i \\ \mu_i S_i & k_i > S_i \end{cases}$$

N-Node Open Jackson Networks – Problem Specification

- **The total flow to the ith node is computed using:** $\Lambda_i = \lambda_i + \sum_{j=1}^{N} q_{ji}\Lambda_j$
- **The queue at node i is stable if $\Lambda_i < \mu_i S_i$; i=1,2, …, N**

N-Node Open Jackson Networks – Global Balance Equations

- **Along the same lines we followed for the two-node system, the global are given by**

$$P(k_1,k_2,\ldots,k_N)\left[\sum_{i=1}^{N}\left(\lambda_i+\mu_i d(k_i)\right)\right]=\begin{cases}\displaystyle\sum_{i=1}^{N}P(k_1,k_2,\ldots,k_i-1,\ldots,k_N)\lambda_i \\ \displaystyle+\sum_{i=1}^{N}P(k_1,k_2,\ldots,k_i+1,\ldots,k_N)\mu_i d(k_i+1)q_{i,N+1} \\ \displaystyle+\sum_{i=1}^{N}\sum_{j=1}^{N}P(k_1,k_2,\ldots,k_i+1,\ldots,k_j-1,\ldots,k_N)\mu_i d(k_i+1)q_{i,j} \\ \forall k_1,k_2,\ldots,k_N\geq 0\end{cases}$$

Flow out of state $(k_1,k_2,\ldots,k_N)$

Flow into of state $(k_1,k_2,\ldots,k_N)$

Due to an arrival

Due to a departure to outside the network

Due to a departure from node i to node j

N-Node Open Jackson Networks – Global Balance Equations – cont'd

- **We use the following substitutions in the previous global balance equation:**

$$\lambda_i=\Lambda_i-\sum_{j=1}^{N}\Lambda_i q_{ji} \qquad i=1,2,\ldots,N$$

$$q_{i,N+1}=1-\sum_{j=1}^{N}q_{ij} \qquad i=1,2,\ldots,N$$

N-Node Open Jackson Networks – Global Balance Equations – cont'd

- **Rewriting the global balance equation:**

$$
\begin{aligned}
&P(k_1,k_2,\ldots,k_N)\Lambda_i+\sum_{i=1}^{N}P(k_1,k_2,\ldots,k_i-1,\ldots,k_N)\mu_i d(k_i)\\
&\quad+\sum_{i=1}^{N}\sum_{j=1}^{N}P(k_1,k_2,\ldots,k_j-1,\ldots,k_N)\Lambda_i q_{ij}\\
&\quad+\sum_{i=1}^{N}\sum_{j=1}^{N}P(k_1,k_2,\ldots,k_i+1,\ldots,k_N)\mu_i d(k_i+1)q_{ij}\\
&=\sum_{i=1}^{N}P(k_1,k_2,\ldots,k_i+1,\ldots,k_N)\mu_i d(k_i+1)\\
&\quad+\sum_{i=1}^{N}P(k_1,k_2,\ldots,k_i-1,\ldots,k_N)\Lambda_i\\
&\quad+\sum_{i=1}^{N}\sum_{j=1}^{N}P(k_1,k_2,\ldots,k_i+1,\ldots,k_j-1,\ldots,k_N)\mu_i d(k_i+1)q_{ij}\\
&\quad+\sum_{i=1}^{N}\sum_{j=1}^{N}P(k_1,k_2,\ldots,k_N)\Lambda_i q_{ij}
\end{aligned}
$$

N-Node Open Jackson Networks – Joint Probability Mass Function

- **The global balance equation is satisfied if**

$$\mu_i d(k_i+1)P(k_1,k_2,\ldots,k_i+1,\ldots,k_N)=\Lambda_i P(k_1,k_2,\ldots,k_i,\ldots,k_N)$$

or

$$P(k_1,k_2,\ldots,k_i+1,\ldots,k_N)=\frac{R_i}{d(k_i+1)}P(k_1,k_2,\ldots,k_i,\ldots,k_N)$$

where $R_i = \Lambda_i / \mu_i$

- **The solution to the above equation is given by**

$$P(k_1,k_2,\ldots,k_N)=G^{-1}\prod_{i=1}^{N}\frac{R_i^{k_i}}{\prod_{j=1}^{k_i}d(j)};\quad \forall k_1,k_2,\ldots,k_N\geq 0$$

where G^{-1} is the normalization constant

N-Node Open Jackson Networks – Joint Probability Mass Function – Single Server Nodes

- **Assume single server nodes – i.e. $S_i = 1 \ \forall$ i = 1, 2, …, N**
- **Therefore,** $d(j) = 1; \quad j = 1,2,\ldots,N$
- **The joint PMF is given by**

$$P(k_1,k_2,\ldots,k_N) = G^{-1}\prod_{i=1}^{N} R_i^{k_i}; \quad \forall k_1,k_2,\ldots,k_N \geq 0$$

- **Hence, the normalization constant is given by** $G^{-1} = \prod_{i=1}^{N}(1-R_i)$
- **Rewriting the PMF results in**

$$P(k_1,k_2,\ldots,k_N) = \prod_{i=1}^{N}(1-R_i)R_i^{k_i}; \quad \forall k_1,k_2,\ldots,k_N \geq 0$$

N-Node Open Jackson Networks – Joint Probability Mass Function – Infinite Server Nodes

- **Assume infinite server nodes – i.e. $S_i = \infty \ \forall$ i = 1, 2, …, N**
- **Therefore,** $d(j) = j; \quad j = 1,2,\ldots,N$
  and $\prod_{j=1}^{k_i} d(j) = k_i!$
- **The joint PMF is given by**

$$P(k_1,k_2,\ldots,k_N) = G^{-1}\prod_{i=1}^{N} \frac{R_i^{k_i}}{k_i!}; \quad \forall k_1,k_2,\ldots,k_N \geq 0$$

- **Hence, the normalization constant is given by** $G^{-1} = \prod_{i=1}^{N} e^{-R_i}$
- **Rewriting the joint PMF results in**

$$P(k_1,k_2,\ldots,k_N) = \prod_{i=1}^{N} \frac{e^{-R_i}}{k_i!} R_i^{k_i}; \quad \forall k_1,k_2,\ldots,k_N \geq 0$$

N-Node Open Jackson Networks – Performance Calculations

- **The marginal probability of node i having k_i messages is given by**

$$P(Q_i = k_i) = (1 - R_i^{k_i}) R_i^{k_i}; \quad i = 1, 2, \ldots, N$$

- **If the nodes in the system have limited buffer size M, then probability of buffer overflow may be approximated by**

$$P(Q_i \geq M) = \sum_{k_i = M}^{\infty} (1 - R_i^{k_i}) R_i^{k_i} = R_i^{M}; \quad i = 1, 2, \ldots, N$$

- **The mean and variance of total number of customers in N nodes is given by**

- Prove these two equations?
- For large number of nodes, one may invoke the central-limit theorem to approximate the distribution of the total number of customers in the system by a Gaussian distribution

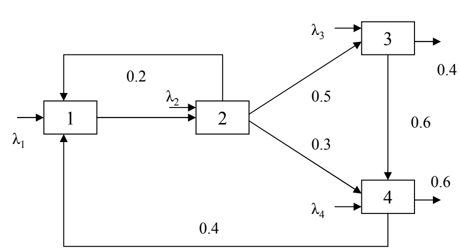
$$E[k_1 + k_2 + \ldots + k_N] = \sum_{i=1}^{N} \frac{R_i}{1 - R_i}$$

$$Var[k_1 + k_2 + \ldots + k_N] = \sum_{i=1}^{N} \frac{R_i}{(1 - R_i)^2}$$

Example: N-Node Open Jackson Networks

- **<u>Problem</u>: Consider the network of queues depicted in figure. Assume λ = [2.0, 1.0, 0.5, 0.3] and μ = [0.1, 0.07, 0.03, 0.075]**

a) **Find the routing matrix**
b) **Calculate the total traffic flow vector, and the resulting loads at each queue node**
c) **Approximate the probability mass function of the total number of customers in the system using the Gaussian distribution**
d) **Find the exact probability mass function of the total number of customers and compare it to the one obtained in part (c).**



Example: N-Node Open Jackson Networks – cont'd

- **Solution:**

a) **The routing matrix, Q, is given by** $Q=\begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0.2 & 0 & 0.5 & 0.3 \\ 0 & 0 & 0 & 0.6 \\ 0.4 & 0 & 0 & 0 \end{bmatrix}$

b) **Therefore the total traffic flow is given by** $\Lambda=\lambda[I-Q]^{-1}=[4.7857 \quad 5.7857 \quad 3.3929 \quad 4.0714]$

and the loads are $R=[0.4786 \quad 0.4050 \quad 0.1018 \quad 0.3054]$

c) The mean total messages in system is given by $E[k_1+k_2+k_3+k_4]=m=\sum_{i=1}^{4}\frac{R_i}{1-R_i}=2.1514$

while the variance is given

$$Var[k_1+k_2+k_3+k_4]=\sum_{i=1}^{4}\frac{R_i}{(1-R_i)^2}=3.6632$$

or the standard deviation is

$$\sigma=\sqrt{\sum_{i=1}^{4}\frac{R_i}{(1-R_i)^2}}=1.9139$$

Example: N-Node Open Jackson Networks – cont'd

- **Solution-cont'd:**

c) Assuming the total number of customers can be approximated by a Gaussian distribution, then $f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-(x-m)^2/(2\sigma^2)}$

where m and σ quantities are computed earlier.

Then CDF of total number of customers can be computed using $F(x)=0.5+0.5erf\left(\frac{x-m}{\sigma\sqrt{2}}\right)$

Refer to the Gaussian distribution material

Example: N-Node Open Jackson Networks – cont'd

- **Solution-cont'd:**

d) To calculate the exact distribution we need to evaluation the following

$$\Pr ob(total = j) = \sum_{\forall(k_1+k_2+k_3+k_4=j)} \prod_{i=1}^{4} (1-R_i)R_i^{k_i} \qquad j = 0,1,2,\ldots$$

to obtain the PMF function.

Subsequently, the CDF is given by

$$F(k) = \sum_{j=0}^{k} \Pr ob(total = j) \qquad k = 0,1,2,\ldots$$

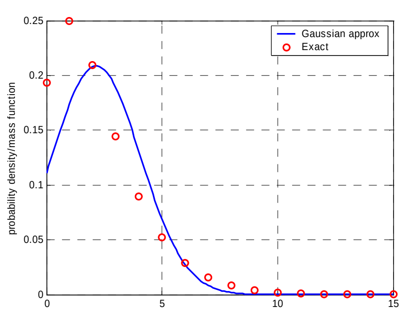
A Matlab code is used to find all the (k_1, k_2, k_3, k_4) that add to a particular j and then the k_is are substituted in the above expression to calculate the PMF.

Example: N-Node Open Jackson Networks – cont'd

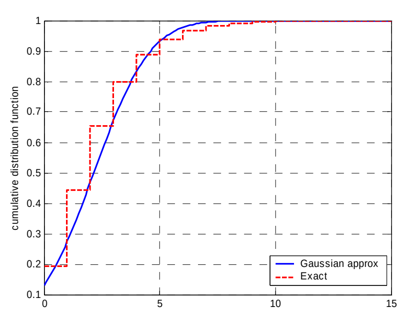
- **Solution-cont'd:**

d) The probability density/mass functions for the total number of customers in system are shown in figure (I)

The cumulative probability functions for the total number of customers in system are shown in figure (II)



(I)



(II)

Example: N-Node Open Jackson Networks – cont'd (Matlab Code)

- **Main Code: Example4_3_1.m**

```
%
% example 4.3 of Hayes
clear all
LineWidth = 2;
Lambda = [2.0, 1.0,  0.5,  0.3];
M      = 1./[0.1, 0.07, 0.03, 0.075];
Q = [0    1.0  0    0  ; ...
     0.2  0    0.5  0.3; ...
     0    0    0    0.6; ...
     0.4  0    0    0];

Omega = Lambda * inv(eye(4) - Q);
R     = Omega ./ M;

MeanTotal = 0;
VarTotal  = 0;
for i=1:4
    MeanTotal = MeanTotal + R(i)/(1-R(i));
    VarTotal  = VarTotal  + R(i)/(1-R(i))^2;
end
Nmax = 15; % max range for probability functions
%
% compute the Gaussian
nG  = 0:0.1:Nmax;
f1  = 1/sqrt(2*pi*VarTotal)*exp(-(nG-
    MeanTotal).^2/(2*VarTotal));
F1  = 0.5 + 0.5*erf((nG-MeanTotal)/(sqrt(2*VarTotal)));
%
% Compute the exact
f2  = GetExactDistribution(R, Nmax);
F2 = cumsum(f2);
[nE, FF2] = stairs(0:15, F2);
%
% plot results
figure(1);
h = plot(nG, f1, '-', 0:Nmax, f2,'or');
set(h, 'LineWidth', LineWidth);
ylabel('probability density/mass function');
xlabel('total number of customers');
legend('Gaussian approx', 'Exact');
grid

figure(2);
h = plot(nG, F1, '-', nE, FF2,'--r');
set(h, 'LineWidth', LineWidth);
ylabel('cumulative distribution function');
xlabel('total number of customers');
legend('Gaussian approx', 'Exact');
grid
```

- **GetExactDistribution function**

```
function P = GetExactDistribution(R, N);
%
% example 4.3 of Hayes
% computation of exact pmf for total
% number of customers, N
% R is a vector of loads

P = zeros(1,N+1);
for i=0:N;
    [AllKs, m] = GetAllKs( i );
    for j=1:m
        P(i+1) = P(i+1) +
  ComputeFromJoint(AllKs(j,:), R);
    end
end
```

- GetAllKs function

```
function [AllKs, m] = GetAllKs( N );
%
% function returns all possible 4 numbers that add
  to N
m     = 0;
AllKs = [];
for i1=0:N
    for i2=0:N
        for i3=0:N
            for i4=0:N
                if (N == (i1+i2+i3+i4)) % found one
                    % check if it is not aready
  included
                    %[i1 i2 i3 i4]
                    if (NotIncludedYet([i1 i2 i3
  i4], AllKs, m))
                        %
                        % include it
                        m = m + 1;
                        AllKs(m,:) = [i1 i2 i3 i4];
                    end
                end
            end
        end
    end
end
```


Example: N-Node Open Jackson Networks – cont'd (Matlab Code)

- ComputeFromJoint fuction

```
function P = ComputeFromJoint(Vector, R);
%
% use the product form to evaluate
P = 1;
for i=1:length(R)
    P = P*(1-R(i))*(R(i)^Vector(i));
end
```

- NotIncludedYet function

```
function Flag = NotIncludedYet(Vector, AllKs, m)
%
% check if Vector is already included in AllKs
Flag = 1;
if (m)
    for i=1:m
        if (sum(Vector == AllKs(i,:)) == 4)
            % i.e. vector found
            Flag = 0;
            break;
        end
    end
end
```

Average Message Delay

- **In an open network of N nodes, we have shown that the mean number of customers in network is given by** $E[k_1+k_2+\ldots+k_N]=\bar{\rho}=\sum_{i=1}^{N}\frac{R_i}{1-R_i}$
- **Therefore, using Little's formula we have** $\bar{\rho}=E[T]\sum_{i=1}^{N}\lambda_i$ **where E[T] is the mean delay for a customer**
- **We can also apply Little's formula on the individual nodes as in** $\bar{\rho}=\sum_{i=1}^{N}\lambda_i E[T_i]$

  where $E[T_i]$ is the mean delay of a customer originating at node i
- **However, obtaining $E[T_i]$ is not straight forward!!**

Average Message Delay – cont'd

- **Assume Λ_i is the total flow into node i and D_i is the customer delay at node i – then another application of Little's formula results in** $\bar{\rho}=\sum_{i=1}^{N}\Lambda_i E[D_i]$
- **Combining the previous results, we obtain**

$$E[T]=\frac{\sum_{i=1}^{N}\Lambda_i E[D_i]}{\sum_{i=1}^{N}\lambda_i}$$

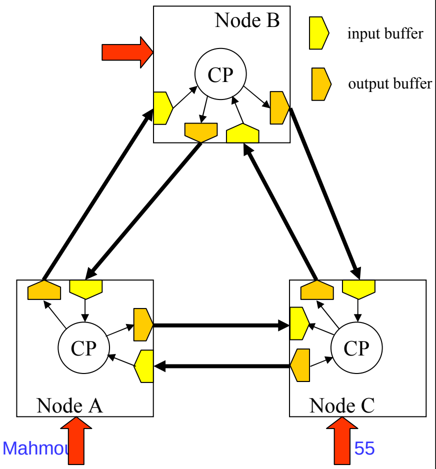
Remember
$\Lambda_i E[D_i] = R_i/(1-R_i)$

- **Therefore, the final result is given by** $E[T]=\frac{\sum_{i=1}^{N}R_i/(1-R_i)}{\sum_{i=1}^{N}\lambda_i}$

  which is the results used in previous examples to compute the mean end-to-end delay

Store-and-Forward Message Switched Networks

- Consider the 3-node store-and-forward network depicted in figure
- The focus is on the queueing delay in the output buffers
- Arrival of messages to output buffers is rapid and can be modeled by Poisson arrivals
  - Prob of more than one arrival in an infinitesimal time period ≈ 0
  - Accumulation of traffic from multiple lines
  - Independence!!
- Is message service time independent of the arrival process?
- Independent Assumption: the service time of a message is chosen independently at each node
  - Many sources feed into one queue – valid approximation



Store-and-Forward Message Switched Networks – Delay Optimization

- **<u>Objective</u>: Determine link capacities (bit/sec) so that mean end-to-delay for network is minimum**
- **Assumptions and notations:**
  - Packet size = B bits
  - ith link/node/server capacity = C_i bit/sec ➔ mean service time is equal to $E[M_i] = E[B]/C_i$
- **Previously, we have derived the mean delay to be**

$$E[T] = \frac{1}{\alpha}\sum_{i=1}^{N}\frac{\Lambda_i E[B]}{C_i - \Lambda_i E[B]} = \frac{1}{\alpha}\sum_{i=1}^{N}\frac{I_i}{C_i - I_i}$$

Note that $R_i = \Lambda_i/\mu_i = \Lambda_i E[M_i] = \Lambda_i E[B]/C_i$ – $\alpha = \Sigma\lambda_i$ (i.e. sum of external arrivals) – $I_i = \Lambda_i E[B_i]$

Store-and-Forward Message Switched Networks – Delay Optimization – cont'd

- **One can refine the mean delay formula by including the link propagation time, P_i – The resulting formula is**

$$E[T] = \frac{1}{\alpha}\sum_{i=1}^{N}\left(\frac{I_i}{C_i - I_i} + \Lambda_i P_i\right)$$

Store-and-Forward Message Switched Networks – Delay Optimization – cont'd

- **Let us define the following performance figure**

$$E\left[T^k\right] = \frac{1}{\alpha}\left[\sum_{i=1}^{N}\left(\frac{I_i}{C_i - I_i}\right)^k\right]^{1/k}$$

- **Special cases:**
  - **For k=1 ➔ mean delay**
  - **For k=2 ➔ standard deviation of delay (assuming the delays are independent)**

Store-and-Forward Message Switched Networks – Delay Optimization – cont'd

- **Special cases (cont'd):**
  - **For k=∞ (refer to reference below)**

$$E\left[T^{\infty}\right] = \lim_{k\to\infty} \frac{1}{\alpha}\left[\sum_{i=1}^{N}\left(\frac{I_i}{C_i - I_i}\right)^k\right]^{1/k} = \frac{1}{\alpha}\max_i\left(\frac{I_i}{C_i - I_i}\right) = \frac{I_{k^*}}{\alpha\left(C_{k^*} - I_{k^*}\right)}$$

where k* is the value of i for which $I_i/(C_i-I_i)$ is maximum over i=1,2,…,N

- **Two applications:**
  - **Given I_i; i=1,2, …, N determine C_is such that mean delay is minimum – This will be tackled later**
  - **Given C_i; i=1,2, …, N determine the routing scheme (i.e. I_is) such that mean delay is minimum**

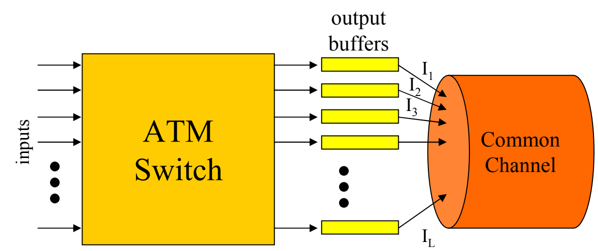
Meister, B.; Muller, H.; Rudin, H., Jr.; "New Optimization Criteria for Message-Switching Networks," IEEE Transactions on Communications, Volume: 19 , Issue: 3 , Jun 1971, Pages:256 - 26

Example: Store-and-Forward Message Switched Networks

- **<u>Problem</u>: Example 4.5 in textbook is missing information (average packet size for example) and network topology is unclear.**
- **Mere numerical substitution once the missing information is given**

Capacity Allocation

- Consider the following optimization problem that occurs in store-and-forward networks (switches for example)
- The switch hardware/software allocate capacities for the individual output links such that the sum does exceed the total available capacity
- How would the capacities be allocated?
  - How about minimizing the average delay?



ATM Multiplexing

Capacity Allocation – cont'd

- We have shown the average delay to be

$$E[T] = \frac{1}{\alpha}\sum_{i=1}^{L}\frac{I_i}{C_i - I_i}$$

assuming we have L links. We require the sum of the individual capacities be less than some upper bound C, i.e.

$$C \geq \sum_{i=1}^{L} C_i$$

- You can show that the individual C_ls should be give by

$$C_l = I_l + \frac{\left(C - \sum_{i=1}^{N} I_i\right)\sqrt{I_l}}{\sum_{i=1}^{N}\sqrt{I_i}}; \quad l = 1,2,\ldots,L$$

Capacity Allocation – cont'd

- **Note the allocated capacity is the minimum required (I_l) plus a fraction of the remaining capacity ($C-\Sigma I_l$)**

- **The minimum average is equal to**

$$E[T]_{\min} = \frac{1}{\alpha} \frac{\left(\sum_{i=1}^{N} \sqrt{I_i}\right)^2}{C - \sum_{i=1}^{N} I_i}$$

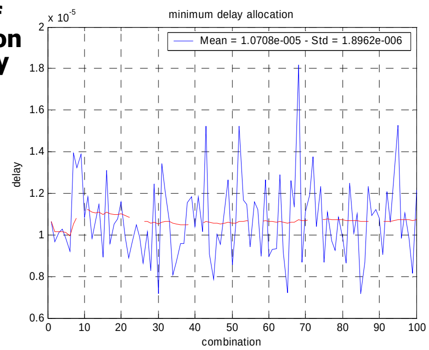
Example: Capacity Allocation

- **__Problem__: Assume 10 OC-1 (51.84 Mb/s) inputs are multiplexed on an output link who is total capacity is OC-12 (622.08 Mb/s) – If the volume of the input lines is chosen at random, determine the optimal allocation for each case and the average and standard deviation of the overall delay.**

For a nice table of the OC hierarchy please refer to http://www.linktionary.com/o/oc.html

Example: Capacity Allocation – cont'd

- **Solution: It can be seen that if the optimum capacity allocation is always used, the mean delay is 1.07e-5 while the standard deviation is 1.9e-6**




```
clear all
OC1  = 51.84e6;
OC12 = 622.08e6;
L    = 10; % L inputs
N    = 100;
for i=1:N
    Is   = OC1 * rand(1,L);
    RemC = OC12 - sum(Is);
    Cs   = Is + RemC * sqrt(Is)./sum(sqrt(Is));
    Alpha = sum(Is)/(53*8);
    T(i) = sum(sqrt(Is))^2/RemC/Alpha;
end
for i=1:N % compute mean
    TM(i) = mean(T(1:i));
end

figure(1);
h = plot(1:N, T, '-', 1:N, TM,'--r');
title('minimum delay allocation');
xlabel('combination');
ylabel('delay');
legend(['Mean = ' num2str(mean(T)) ' - Std = '
    num2str(std(T))]);
grid
```

What are the units of the delay in the above curve?

Closed Jackson Networks

- **Closed: fixed number of messages circulate within the network with neither arrivals to nor departures from the network**
- **Classic application – computer system**
  - **Over a short period it can be assumed that tasks/processes/customers neither enter nor leave the system**

Closed Jackson Networks – Traffic Equation

- **Since there are no external arrivals, the traffic equation reduces to**

$$\Lambda_i = \sum_{j=1}^{N} q_{ji}\Lambda_j$$

$$[\Lambda_1 \quad \Lambda_2 \quad \cdots \quad \Lambda_N] = [\Lambda_1 \quad \Lambda_2 \quad \cdots \quad \Lambda_N]\begin{bmatrix} 0 & q_{12} & \cdots & q_{1N} \\ q_{21} & 0 & \cdots & q_{2N} \\ \vdots & & 0 & \\ q_{N1} & q_{N2} & \cdots & 0 \end{bmatrix}$$

$$\Lambda = \Lambda Q$$

$$\Lambda^T = Q^T \Lambda^T$$

Note that Λ is the transpose of the eigenvector for the matrix Q^T corresponding to the eigenvalue 1!!

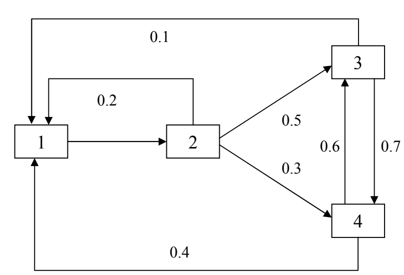
For a nice and very breif introduction to eigenvalues and eigenvectors please refer to http://www.sosmath.com/matrix/eigen0/eigen0.html or http://maths.ucd.ie/courses/math1200/algebra/algebranotes7-1.pdf
In Matlab do "help eig" to get help regarding eigenvalues/vectors calculations

Example: Traffic Equation for Closed Networks

- **<u>Problem</u>: For the closed network shown in figure,**

a) Find the routing matrix, Q?

b) Compute the total flows into each node?

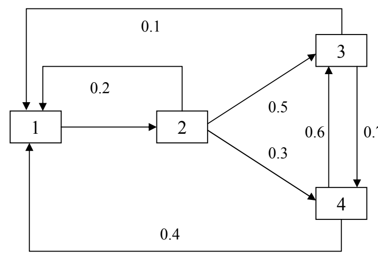


Example: Traffic Equation for Closed Networks

- **Solution:**

A) The routing matrix, Q:

$$Q = \begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0.2 & 0 & 0.5 & 0.3 \\ 0.1 & 0.2 & 0 & 0.7 \\ 0.4 & 0 & 0.6 & 0 \end{bmatrix}$$



Example: Traffic Equation for Closed Networks

- **Solution:**

B) The eigenvectos/values are calculated as shown

Hence, the *relative* flows are

Λ = [1.0 1.31 1.53 1.46]

```
>> [Vectors, Values] = eig(Q');
>> Vectors

Vectors =

   0.3728             0.0490 - 0.2601i   0.0490 + 0.2601i  -0.2709
   0.4870            -0.7672            -0.7672             0.5362
   0.5710             0.2160 + 0.1956i   0.2160 - 0.1956i  -0.6822
   0.5458             0.5022 + 0.0645i   0.5022 - 0.0645i   0.4169

>> Values

Values =

   1.0000                  0                  0                  0
        0            -0.1202 + 0.2880i        0                  0
        0                  0            -0.1202 - 0.2880i        0
        0                  0                  0            -0.7596

>> Vectors(:,1)./Vectors(1,1)

ans =

    1.0000
    1.3063
    1.5315
    1.4640
```

Closed Jackson Networks – Global Balance Equations

- **Same assumptions as before**
  - **Exponential and independent service time**
  - **Si servers at node i**
- **K – total number of customers**
- **An easy extension to the equations derived for open networks**

$$P(k_1,k_2,\ldots,k_N)\sum_{i=1}^{N}\mu_i d(k_i)=\sum_{i=1}^{N}\sum_{j=1}^{N}\mu_i d(k_i+1)q_{ij}P(k_1,k_2,\ldots,k_i-1,\ldots,k_j+1,\ldots,k_N)$$

$$\forall k_1,k_2,\ldots,k_N\geq 0$$

Closed Jackson Networks – Global Balance Equations – cont'd

- **It can be shown (following the same derivation process as that for open networks), the joint pmf is given by**

$$P(k_1,k_2,\ldots,k_N)=G(K,N)^{-1}\prod_{i=1}^{N}\left[R_i^{k_i}\Big/\prod_{j=1}^{k_i}d(j)\right]$$

$$=\begin{cases} G(K,N)^{-1}\prod_{i=1}^{N}R_i^{k_i} & \text{single server nodes}\\ G(K,N)^{-1}\dfrac{\prod_{i=1}^{N}R_i^{k_i}}{k_i!} & \text{infinite server nodes}\end{cases}$$

where Λ_1, Λ_2, …, Λ_N, is the solution to the traffic equation. $R_i = \Lambda_i/\mu_i$ and G(K,N) is a the normalization constant

Convolution Algorithm

- **How to calculate the normalization constant?**
- **Exhaustive method: find all (k_1, k_2, …, k_N) such that $\Sigma k_i = K$ – substitute in joint pmf and compute the constant G(K,N) such that the sum is equal to 1.**
  - **There are (N+K-1)!/(K!(N-1)!) ways - e.g. N = 4, K = 7 → 120 combinations!!**
  - **Prohibitive!!**
- **Use convolution algorithms**

Convolution Algorithm – Buzen – Simplified Version

- **Single server nodes → service rate is always μ – does not depend on number of customers at node**
- **Define S(k,n) = {k_1, k_2, …, k_n / $\Sigma k_i = k$; $0 \le k \le K$; $1 \le n \le N$}**
- **Define G(k,n) by summing over the set S(k,n)**

$$G(k,n) = \sum_{S(k,n)} \prod_{i=1}^{n} R_i^{k_i}$$

- **G(k, n) is the sum over all possible ways of dispersing k messages among n nodes.**

Convolution Algorithm – Buzen – Simplified Version – cont'd

- **How to compute G(k,n)? – consider splitting the summation into**
  - **kn = 0**
  - **kn > 0**
- **Therefore we can write G(k,n) as**

$$G(k,n)=\sum_{\substack{S(k,n)\\k_n=0}}\prod_{i=1}^{n}R_i^{k_i}+\sum_{\substack{S(k,n)\\k_n>0}}\prod_{i=1}^{n}R_i^{k_i}$$

- **But the first summation is just the sum over the first n-1 nodes since the nth node is empty. i.e. G(k,n-1)**
- **For the second summation – there is at least one message in node n – i.e. there are at most k-1 *other* messages in the total network. i.e. G(k-1,n**
- **Hence, the G(k,n) can be written as**

$$G(k,n)=G(k,n-1)+R_nG(k-1,n)$$

Convolution Algorithm – Buzen – Simplified Version – cont'd

- **We can show that**

$$G(k,n)=G(k,n-1)+R_nG(k-1,n)$$

- **The initiating values:**

$$G(k,1)=R_1^k\,;\quad k=1,2,\ldots,K$$

$$G(0,n)=1;\quad n\geq 1$$

- **What is G(1,n) equal to for n >0?**

Example: Convolution Algorithm

- **<u>Problem</u>: Assume a four node network with the routing matrix Q**

  Assume μ = [2.5 2.5 2.5 2.5] and finite population of K = 7.

  A) Find the relative total flows

  B) Compute the joint distribution $P(k_1, k_2, k_3, k_4)$

$$Q = \begin{bmatrix} 0 & 0.75 & 0.25 & 0 \\ 0.05 & 0 & 0.15 & 0.8 \\ 0.25 & 0.25 & 0 & 0.5 \\ 0.4 & 0.35 & 0.25 & 0 \end{bmatrix}$$

Example: Convolution Algorithm

- **<u>Solution</u>: Closed Network K = 7, N = 4**

  A) Relative flows (found in the same manner as previous example

  Λ = [1.0000 1.5844 0.9195 1.7273]

  B) To compute the joint distribution $P(k_1, k_2, k_3, k_4)$, we need to compute:

  R = Λ./μ

  = [0.4000 0.6338 0.3678 0.6909]

  Note R is the RELATIVE loading

  We also need to compute G(K,N) using Buzen's convolution algorithm

Example: Convolution Algorithm – cont'd

- **<u>Solution</u>: cont'd**

  Using the recursive algorithm (Refer to Matlab code) – G(7,4) = 1.7036

  Therefore the joint pmf is equal to

$$P(k_1,k_2,\ldots,k_N)=\prod_{i=1}^{N}R_i^{k_i}\Big/G(K,N)$$
$$=(0.4)^{k_1}(0.6338)^{k_2}(0.0.3678)^{k_3}(0.0.6910)^{k_4}/1.7036$$

Example: Convolution Algorithm – cont'd

- **<u>Solution</u>: cont'd**

  The following code implements the recursive alogorithm:

```
%
% Example 4.8

K = 7;
N = 4;
M = 2.5*ones(1,N);

Q = [0    0.75  0.25  0; ...
     0.05 0     0.15  0.8; ...
     0.25 0.25  0     0.5; ...
     0.4  0.35  0.25  0];

[Vectors, Values] = eig(Q');
RFlows = Vectors(:,1)'./Vectors(1,1); % relative flows
RR     = RFlows./M; % compute relative loads

ks = 0:K;
ns = 1:N;
G_K_N = zeros(K+1,N);
%
% fill initial values
G_K_N(1,:) = ones(1,N);
G_K_N(:,1) = RR(1).^ks';
%
% fill the remaining of the matrix
for n=2:N
    for k=1:K
        G_K_N(k+1, n) = G_K_N(k+1, n-1) + RR(n)*G_K_N(k, n);
    end
end
```

<u>Program output:</u>

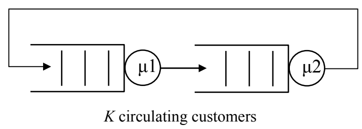
```
>> RFlows

RFlows =

    1.0000    1.5844    0.9195    1.7273

>> RR

RR =

    0.4000    0.6338    0.3678    0.6909

>> G_K_N

G_K_N =

    1.0000    1.0000    1.0000    1.0000
    0.4000    1.0338    1.4016    2.0925
    0.1600    0.8152    1.3306    2.7764
    0.0640    0.5806    1.0700    2.9882
    0.0256    0.3936    0.7871    2.8517
    0.0102    0.2597    0.5492    2.5195
    0.0041    0.1687    0.3707    2.1114
    0.0016    0.1085    0.2449    1.7036

>>
```

Example: Two-Node Network

- **<u>Problem</u>: Assume a network as shown in Figure with K total number of customers. The service rate for nodes 1 and 2 are μ1 and μ2, respectively. Using the theory of closed networks**

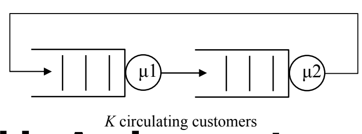
A) Derive the joint probability mass function

B) Derive the marginal distributions for each of the nodes



Example: Two-Node Network – cont'd

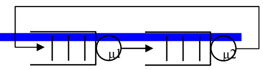
- **<u>Solution</u>:**



This problem was solved in Assignment #2 as an example of a birth-and-death process.

Apply the theory of closed networks and make sure get matching answers

Example: Two-Node Network – cont’d



K circulating customers

- **Solution:**

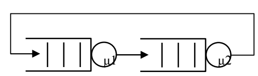
The routing matrix is given by $Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Q^T has two eigen values: 1 and -1. The vector corresponding to the eigen value 1 is √2$(1,1)^T$.

Therefore, the relative total flow is given by Λ = [1 1] and the relative loading is given by R = [1/μ1 1/μ2]

Example: Two-Node Network – cont’d

- **Solution:**



K circulating customers

From the closed network theory, the joint pmf is given by

$$P(k_1, k_2) = \frac{R_1^{k_1} R_2^{k_2}}{G(K,2)}$$

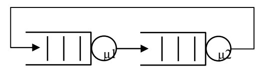
To find G(K,2) we either use the exhaustive method or follow the convolution algorithm explained in class

<u>The exhaustive method</u>: all (k_1, k_2) states such that k_1+k_2 = K can be written as (k_1, K-k_1) for k_1 = 0, 1, …, K

Therefore $G(K,2) = \sum_{i=0}^{K} R_1^{k_1} R_2^{K-k1}$

Example: Two-Node Network – cont'd

- **Solution:**



K circulating customers

The convolution method: is shown in table on the side – Again, G(K,2) is given by

$$G(K,2)=\sum_{i=0}^{K} R_1^{\,i} R_2^{\,K-i}$$

	1	2
0	1	1
1	R_1	R_1+R_2
2	R_1^2	$R_1^2+R_1R_2+R_2^2$
3	R_1^3	$R_1^3+R_1^2R_2+R_1R_2^2+R_2^3$
.		.
.		
K	R_1^K	$\sum_{i=0}^{K} R_1^{K-i} R_2^i$

Therefore the final joint PMF is given by

$$P(k_1,k_2)=\frac{R_1^{\,k_1} R_2^{\,k_2}}{\sum_{i=0}^{K} R_1^{\,i} R_2^{\,K-i}}$$

Example: Two-Node Network – cont'd

- **Solution:**

The marginal distribution for node 1 is given by

$$P(k_1)=P(k_1,K-k_1)=\frac{R_1^{\,k_1} R_2^{\,K-k_1}}{\sum_{i=0}^{K} R_1^{\,i} R_2^{\,K-i}}$$

$$=\frac{R_2^{\,K}\left(R_1/R_2\right)^{k_1}}{R_2^{\,K}\sum_{i=0}^{K}\left(R_1/R_2\right)^{i}}$$

$$=\frac{\left(\mu_2/\mu_1\right)^{k_1}}{\sum_{i=0}^{K}\left(\mu_2/\mu_1\right)^{i}}$$

The same result obtained before

$$=\frac{1-\mu_2/\mu_1}{1-\left(\mu_2/\mu_1\right)^{K+1}}\left(\frac{\mu_2}{\mu_1}\right)^{k_1}$$

Mean Number of Message in Each Queue

- **Consider a closed network with K messages and N nodes**
- **Probability of node i having k customers or more (i.e. K-k or less are dispersed in the rest of N-1 nodes) is given by**

$$\Pr ob(Q_i \geq k) = \frac{\sum_{\substack{S(K,N)\\ k_i \geq k}} \prod_{i=1}^{N} R_i^{k_i}}{G(K,N)} = \frac{R_i^k \sum_{S(K-k,N)} \prod_{i=1}^{N} R_i^{k_i}}{G(K,N)}$$

$$= \frac{R_i^k G(K-k,N)}{G(K,N)}; \quad k \geq 1$$

Mean Number of Message in Each Queue – cont'd

- **Therefore, Probability of node i having exactly k customers is given by**

$$\Pr ob(Q_i = k) = \Pr ob(Q_i \geq k) - \Pr ob(Q_i \geq k+1)$$

$$= \frac{R_i^k G(K-k,N) - R_i^{k+1} G(K-k-1,N)}{G(K,N)}$$

- **Therefore, the mean number of customers in node i is given by**

$$E[K_i] = \sum_{k=0}^{K} k \Pr ob(Q_i = k)$$

$$= \frac{1}{G(K,N)} \sum_{k=0}^{K} k \left[R_i^k G(K-k,N) - R_i^{k+1} G(K-k-1,N) \right]$$

Mean Number of Message in Each Queue – cont'd

- **The previous formula can be simplified to be**

$$E[K_i] = \frac{1}{G(K,N)} \sum_{k=1}^{K} R_i^k G(K-k,N); \quad i = 1,2,\ldots,N$$

- **Can you do the above simplification?**

- **What is $\sum_{i=1}^{N} E[K_i]$ equal to? Prove?**

Absolute Flows

- **The derived quantities Λi and Ri were all relative**
- **Let us derive ρ_i or the ith node utilization**
- **From definition of ρ_i = Prob(Qi ≥ 1), therefore**

$$\Pr ob[Q_i \geq 1] = \rho_i = \frac{R_i G(K-1,N)}{G(K,N)}$$

- **The absolute flow is equal to ρ_i divided by the average service time, i.e.**

$$\Omega_i = \frac{\Lambda_i G(K-1,N)}{G(K,N)}; \quad i = 1,2,\ldots,N$$

Remember $R_i = \Lambda_i / \mu_i = \Lambda_i M_i$, where M_i is the average service time

Message Delays

- **The average delay of a message through the ith node can be derived from the mean number of message in node through the application of Little's formula**

$$E[D_i] = \frac{E[K_i]}{\Omega_i}$$

Example: Four-node Networks

- **<u>Problem</u>: Using the four node network specified on slide 77**

  a) Compute the mean number of customers in each of the four nodes

  b) Compute the absolute flow into each node

  c) Compute the mean delay through each of the four nodes

Example: Four-node Networks

- **Solution: Refer to Matlab code on next slide for implementation of previous formula**

a) The mean number of customers per node is given by [0.9071 2.3589 0.7858 2.9483]

b) Absolute flows: [1.2393 1.9636 1.1395 2.1407]

c) Mean delays: [0.7319 1.2013 0.6896 1.3773]

Note that sum of mean number of customers should be equal to K = 7!!

- **Regarding the Matlab code implementation: note that G_K_N matrix is of size K+1 by N – where 1st row corresponds to k=0, 2nd row to k=1, …, and K+1st row to k=K.**
    - **Therefore, G(K-1,N) in the previous formulas corresponds to G_K_N(K,N) in the Matlab code. Similary, G(K-2,N) in formula corresponds to G_K_N(K-1,N) in the Matlab code, and so on**

Example: Four-node Networks

Solution: Matlab code for example

```
%
% Example 4.8
K = 7;
N = 4;
M = 2.5*ones(1,N);

Q = [0    0.75  0.25  0; ...
     0.05 0     0.15  0.8; ...
     0.25 0.25  0     0.5; ...
     0.4  0.35  0.25  0];

[Vectors, Values] = eig(Q');
RFlows = Vectors(:,1)'./Vectors(1,1); % relative flows
RR     = RFlows./M; % compute relative loads

ks = 0:K;
ns = 1:N;
G_K_N = zeros(K+1,N);
%
% fill initial values
G_K_N(1,:) = ones(1,N);
G_K_N(:,1) = RR(1).^ks';
%
% fill the remaining of the matrix
for n=2:N
    for k=1:K
        G_K_N(k+1, n) = G_K_N(k+1, n-1) + RR(n)*G_K_N(k, n);
    end
end
%
% Mean numbers
for i=1:N
    Kmean(i) = sum(RR(i).^(1:K)' .* G_K_N(K:-1:1,N))/...
               G_K_N(K+1,N);
end
Omega = RFlows*G_K_N(K,N)/G_K_N(K+1,N);
Dmean = Kmean./Omega;
```

```
>> Example_4_8
>> RFlows

RFlows =

    1.0000    1.5844    0.9195    1.7273

>> RR

RR =

    0.4000    0.6338    0.3678    0.6909

>> Kmean

Kmean =

    0.9071    2.3589    0.7858    2.9483

>> Omega

Omega =

    1.2393    1.9636    1.1395    2.1407

>> Dmean

Dmean =

    0.7319    1.2013    0.6896    1.3773

>>
```

Infinite Server Case

- **How to compute the normalizing constant**

$$G(k,n)=\sum_{S(k,n)}\prod_{i=1}^{n}\frac{R_i^{k_i}}{k_i!}$$

for an infinite server case

- **The above is the multinomial expansion, i.e.**

$$G(k,n)=\frac{\left[\sum_{i=1}^{n}R_i\right]^k}{k!}$$

The binomial expansion is given by

$$(p+q)^k=\sum_{i=0}^{k}\binom{k}{i}p^i q^{k-i}$$

This is generalized by the multinomial expansion:

$$(x_1+x_2+\ldots+x_n)^k=\sum_{k_1+k_2+\ldots+k_n=k}\frac{k!}{k_1!k_2!\ldots k_n!}x_1^{k_1}x_2^{k_2}\ldots x_{k_n}^{k_n}$$

Infinite Server Case – Joint and Marginal Distributions

- **Therefore, the joint pmf for the closed network case with infinite servers case is given by**

$$P(k_1,k_2,\ldots,k_N)=\frac{K!\prod_{i=1}^{N}\left[R_i^{k_i}/k_i!\right]}{\left[\sum_{i=1}^{N}R_i\right]^K}$$

- **The marginal distribution of the Nth node can also be found as an application of the multinomial expansion:**

$$P(K_N=m)=\sum_{S(K-m,N-1)}\frac{K!\prod_{i=1}^{N}\left[R_i^{k_i}/k_i!\right]}{\left[\sum_{i=1}^{N}R_i\right]^K}=\binom{K}{m}R_N^m\frac{\left[\sum_{i=1}^{N-1}R_i\right]^{K-m}}{\left[\sum_{i=1}^{N}R_i\right]^K}$$

Finite Server Case – Joint and Marginal Distributions – cont'd

- **The previous expression can be applied to obtain the marginal distribution for any node**

Finite Server Case – Mean Number of Customers & Absolute Flows

- **Show that the mean number of customers in the i^{th} node is equal to**

$$E[K_i] = \frac{KR_i}{\sum_{i=1}^{N} R_i}; \quad i = 1,2,\ldots,N$$

Note that the sum of the mean is equal to K!!

- **The i^{th} absolute flow is given by**

$$\Omega_i = \frac{K\Lambda_i}{\sum_{i=1}^{N} R_i}; \quad i = 1,2,\ldots,N$$

Example: Infinite Server Case

- **Problem: Consider the previous four node problem where the single servers are replaced with infinite server models.**

a) Calculate the average number of customers in each of the 4 nodes?

b) Calculate the absolute flows to each of the 4 nodes

Example: Infinite Server Case

- **Solution:**

a) K = 7, N = 4

In the previous example, relative flows and loadings were found to be:

Λ = [1.0000 1.5844 0.9195 1.7273]

$R = \Lambda./\mu$

= [0.4000 0.6338 0.3678 0.6909]

$E[K_i] = KR_i / \Sigma R_i$

$E[K_i]$ = [1.3381 2.1202 1.2304 2.3113]

Check $\Sigma K_i = K = 7$.

b) The absolute flows are given by:

Ω = [3.3453 5.3004 3.0760 5.7783]

Example: Infinite Server Case

- **Solution**:

```
%
% Example 4.8
K = 7;
N = 4;
M = 2.5*ones(1,N);

Q = [0    0.75  0.25  0; ...
     0.05 0     0.15  0.8; ...
     0.25 0.25  0     0.5; ...
     0.4  0.35  0.25  0];

[Vectors, Values] = eig(Q');
RFlows = Vectors(:,1)'./Vectors(1,1); % relative flows
RR     = RFlows./M; % compute relative loads

KMean = K*RR./sum(RR);
Omega = K*RFlows./sum(RR);
```

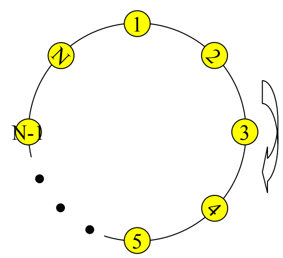
```
>> Example_4_9
>> KMean

KMean =

    1.3381    2.1202    1.2304    2.3113

>> sum(KMean)

ans =

     7

>> Omega

Omega =

    3.3453    5.3004    3.0760    5.7783

>>
```


Mean Value Analysis

- **Numerical problems arise when attempting to compute the normalization constant using the convolutional method**
- **Alternative – Mean value analysis**
  - **It yields averages rather than distributions**
  - **Usually sufficient for most applications.**
- **Based on the arrival theorem:**
  - **Within a closed chain containing k messages, the distribution of the number of messages of it own class seen by a message arriving at a node is the steady-state distribution for the case of one less message in the chain, k-1.**
  - **In contrast – for Poisson arrivals in an open network, the steady-state distribution and the distribution seen by an arriving message are identical**
- **For a simplified proof of the arrival theorem, refer to chapter 9 of Leon Garcia's textbook.**

Mean Value Analysis – Closed Chains

- **Consider a closed chain as shown in figure**
- **Assume:**
  - **no of circulating messages is k**
  - **Service time in node i is M_i ; i=1,2, …, N**



Mean Value Analysis – Closed Chains – cont'd

- **Delay at node i, is $d_i(k) = M_i[1+n_i(k-1)]$; k=1,2, …, K – where $n_i(k-1)$ is the average number of customers found in queue (or when there are k-1 customers circulating) by the k^{th} customer**
- **Throughput $\lambda(k) = k/\ \Sigma d_i(k)$; k=1,2, …, K – the sum is carried over all N nodes**
  - **Note $\Sigma d_i(k)$ is the total delay around the chain**
- **Applying Little's formula again $n_i(k) = \lambda(k)\ d_i(k)$; i=1,2, …, N; and k=1,2, …, K.**
- **The above procedure is used iteratively to find $d_i(K)$ and $n_i(K)$**
  - **Initially $n_i(0) = 0$; for i=1,2, …, N**

Example: Mean Value Analysis – Closed Chains

- **<u>Problem</u>: Consider the example of a closed chain with K = 14, and N = 6. Assume M = [2.5, 0.75, 0.03, 0.2, 0.5, 1.2].**

  a) Compute the mean number of message at each node.

  b) Use the detailed method outlined on slide 89 to calculate the mean number of message at each node

Example: Mean Value Analysis – Closed Chains

- **<u>Solution</u>:**

  Direct application of the iterative algorithm reveals that

$n_i(K)$ = [12.2999 0.4285 0.0121 0.0870 0.2500 0.9225]

$d_i(K)$ = [30.7514 1.0713 0.0304 0.2174 0.6250 2.3063], and

$\lambda(k)$ = 0.400

The following slide shows the intermediate solutions

Example: Mean Value Analysis – Closed Chains

- **Solution:**

```
%
% Example 4.10a
K = 14;
N = 6;
M = [2.5 0.75 0.03 0.2 0.5 1.2];

n_k_i = zeros(K+1,N);
d_k_i = zeros(K,N);
for k=1:K
    d_k_i(k,:) = M .* ( 1 + n_k_i(k,:) );
    Lambda(k)  = k./sum(d_k_i(k,:));
    n_k_i(k+1,:) = Lambda(k) .* d_k_i(k,:);
end
```


Example: Mean Value Analysis – Closed Chains

- **Solution:**

Output is as shown:

```
>> Example_4_10a
>> n_k_i
n_k_i =
```

$n_i(K)$ ⇒

k=0	0	0	0	0	0	0
k=1	0.4826	0.1448	0.0058	0.0386	0.0965	0.2317
k=2	1.0855	0.2514	0.0088	0.0608	0.1606	0.4328
	1.7990	0.3239	0.0104	0.0732	0.2002	0.5933
	2.6043	0.3695	0.0113	0.0799	0.2234	0.7116
	3.4791	0.3966	0.0117	0.0834	0.2362	0.7930
.	4.4022	0.4118	0.0119	0.0852	0.2430	0.8459
.	5.3568	0.4200	0.0120	0.0861	0.2465	0.8786
.	6.3308	0.4242	0.0121	0.0865	0.2483	0.8980
	7.3163	0.4264	0.0121	0.0868	0.2492	0.9092
	8.3083	0.4275	0.0121	0.0869	0.2496	0.9156
	9.3040	0.4281	0.0121	0.0869	0.2498	0.9190
	10.3018	0.4283	0.0121	0.0869	0.2499	0.9209
k=13	11.3006	0.4285	0.0121	0.0869	0.2500	0.9220
k=14	12.2999	0.4285	0.0121	0.0870	0.2500	0.9225
	i=1	i=2	i=3	i=4	i=5	i=6

Example: Mean Value Analysis – Closed Chains

- **Solution:**
  Output is as shown:

```
>> d_k_i

d_k_i =
```

	i=1	i=2	i=3	i=4	i=5	i=6
k=0	2.5000	0.7500	0.0300	0.2000	0.5000	1.2000
k=1	3.7066	0.8586	0.0302	0.2077	0.5483	1.4780
k=2	5.2137	0.9386	0.0303	0.2122	0.5803	1.7194
	6.9975	0.9929	0.0303	0.2146	0.6001	1.9119
	9.0109	1.0272	0.0303	0.2160	0.6117	2.0539
•	11.1978	1.0474	0.0304	0.2167	0.6181	2.1516
•	13.5056	1.0588	0.0304	0.2170	0.6215	2.2151
	15.8920	1.0650	0.0304	0.2172	0.6233	2.2543
•	18.3270	1.0682	0.0304	0.2173	0.6241	2.2776
	20.7907	1.0698	0.0304	0.2174	0.6246	2.2911
	23.2708	1.0706	0.0304	0.2174	0.6248	2.2987
	25.7601	1.0710	0.0304	0.2174	0.6249	2.3029
k=13	28.2544	1.0712	0.0304	0.2174	0.6250	2.3051
$d_i(K)$ k=14	30.7514	1.0713	0.0304	0.2174	0.6250	2.3063

$\lambda(K)$

```
>> Lambda

Lambda =
  Columns 1 through 7
    0.1931    0.2929    0.3450    0.3722    0.3861    0.3931    0.39[illegible]
  Columns 8 through 14
    0.3984    0.3992    0.3996    0.3998    0.3999    0.4000    0.4000
```


Example: Mean Value Analysis – Closed Chains

- **Solution:**

  b) Using the detailed method: The code on slide 96 is modified to solve for this particular network.

  The routing matrix Q is changed to reflect the new routing policy for this chain.

  Furthermore, the eigen vector corresponding to eigen value 1 is the 6th column.

Example: Mean Value Analysis – Closed Chains

- **Solution:**

```
%
% Example 4.10a
clear all
K = 14;
N = 6;
M = 1./[2.5 0.75 0.03 0.2 0.5 1.2];

Q = [0  1  0  0  0  0; ...
     0  0  1  0  0  0; ...
     0  0  0  1  0  0; ...
     0  0  0  0  1  0; ...
     0  0  0  0  0  1; ...
     1  0  0  0  0  0];

[Vectors, Values] = eig(Q');
RFlows = Vectors(:,6)'./Vectors(1,1); % relative flows
RR     = RFlows./M; % compute relative loads

ks = 0:K;
ns = 1:N;
G_K_N = zeros(K+1,N);
%
% fill initial values
G_K_N(1,:) = ones(1,N);
G_K_N(:,1) = RR(1).^ks';
%
% fill the remaining of the matrix
for n=2:N
    for k=1:K
        G_K_N(k+1, n) = G_K_N(k+1, n-1) + RR(n)*G_K_N(k, n);
    end
end
%
% Mean numbers
for i=1:N
    Kmean(i) = sum(RR(i).^(1:K)' .* G_K_N(K:-1:1,N))/...
               G_K_N(K+1,N);
end
Omega = RFlows*G_K_N(K,N)/G_K_N(K+1,N);
Dmean = Kmean./Omega;
```

```
>> G_K_N
G_K_N =

  1.0e+006 *

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0001
    0.0000    0.0001    0.0001    0.0001    0.0001    0.0001
    0.0001    0.0001    0.0001    0.0002    0.0002    0.0004
    0.0002    0.0003    0.0004    0.0004    0.0005    0.0009
    0.0006    0.0009    0.0009    0.0010    0.0012    0.0023
    0.0015    0.0022    0.0022    0.0024    0.0030    0.0057
    0.0038    0.0054    0.0055    0.0060    0.0075    0.0144
    0.0095    0.0136    0.0138    0.0150    0.0187    0.0360
    0.0238    0.0341    0.0345    0.0375    0.0468    0.0900
    0.0596    0.0851    0.0862    0.0937    0.1171    0.2251
    0.1490    0.2129    0.2155    0.2342    0.2927    0.5629
    0.3725    0.5322    0.5386    0.5855    0.7319    1.4074

>> Kmean
Kmean =

   12.2999    0.4285    0.0121    0.0870    0.2500    0.9225

>> Dmean
Dmean =

   30.7514    1.0713    0.0304    0.2174    0.6250    2.3063

>> Omega
Omega =

    0.4000    0.4000    0.4000    0.4000    0.4000    0.4000
```


Mean Value Analysis - Generalization

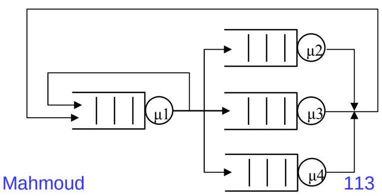
- **The previous iterative algorithm is a closed chain only**
- **The MVA algorithm is modified to accommodate general N-node closed networks:**
  - $d_i(k) = M_i[1+n_i(k-1)]$
  - $\lambda(k) = k / \Sigma[\ \Lambda_i\ d_i(k)\ ]$; $G(k) = G(k-1)/\ \lambda(k)$
  - $n_i(k) = \Lambda_i\ \lambda(k)\ d_i(k)$; $i=1,2, \ldots, N$
  - **Initially** $n_i(0) = 0$; $i=1,2, \ldots, N$; $G(0) = 1$;
  - **The iterations are carried over** $k = 0, 1, \ldots, K$
- **The above algorithm also computes the normalization constant required for the joint pmf distribution!**
- **The above is valid for one class of users – but can be generalized for C classes of users (refer to textbook)**

Example: Mean Value Analysis - Generalization

- **Problem: Consider the network shown in Figure for K = 6. The mean service times are given by M =[0.02 0.2 0.4 0.6]. Furthermore, the relative flows are given by Λ =[1 0.4 0.2 0.1].**

  a) Find the mean number of customers and mead delay for each node.

  b) Use the detailed method to verify your answer



Example: Mean Value Analysis - Generalization

- **Solution:**

  a) Applying the algorithm outlined for the MVA for closed networks, we find

  $n_i(K)$ = [0.2436 2.2610 2.2610 1.2343]
  $d_i(K)$ = [0.0246 0.5698 1.1397 1.2443], and
  $\lambda(k)$ = 9.9198

  Furthermore, the normalization constant G(K,N) is equal to 5.7562e-006

Example: Mean Value Analysis – Generalization – cont'd

- **Solution:**

```
%
% Example MVA for closed network
K = 6;
N = 4;
M = [0.02 0.2 0.4 0.6];
L = [1 0.4 0.2 0.1];
n_k_i = zeros(K+1,N);
d_k_i = zeros(K,N);
G     = ones(K+1,1);
for k=1:K
    d_k_i(k,:) = M .* ( 1 + n_k_i(k,:) );
    Lambda(k)  = k./sum(L.*d_k_i(k,:));
    G(k+1)     = G(k)/Lambda(k);
    n_k_i(k+1,:) = L .* Lambda(k) .* d_k_i(k,:);
end
```


Example: Mean Value Analysis – Generalization – cont'd

- **Solution:**

```
> n_k_i

n_k_i =

         0         0         0         0
    0.0833    0.3333    0.3333    0.2500
    0.1398    0.6882    0.6882    0.4839
    0.1791    1.0608    1.0608    0.6993
    0.2072    1.4485    1.4485    0.8958
    0.2279    1.8491    1.8491    1.0738
    0.2436    2.2610    2.2610    1.2343

>> d_k_i

d_k_i =

    0.0200    0.2000    0.4000    0.6000
    0.0217    0.2667    0.5333    0.7500
    0.0228    0.3376    0.6753    0.8903
    0.0236    0.4122    0.8243    1.0196
    0.0241    0.4897    0.9794    1.1375
    0.0246    0.5698    1.1397    1.2443
```

```
>> G

G =

    1.0000
    0.2400
    0.0372
    0.0047
    0.0005
    0.0001
    0.0000        5.7562e-006

>> Lambda

Lambda =

    4.1667    6.4516    7.8547
     8.7860    9.4401    9.9198
```

Example: Mean Value Analysis – Generalization – cont'd

- **Solution:**

b) using the detailed method

```
%
% Example MVA for closed networks - b
clear all
K = 6;
N = 4;
M = 1./[0.02 0.2 0.4 0.6];
L = [1 0.4 0.2 0.1];
RFlows = L; % relative flows
RR     = RFlows./M; % compute relative loads

ks = 0:K;
ns = 1:N;
G_K_N = zeros(K+1,N);
%
% fill initial values
G_K_N(1,:) = ones(1,N);
G_K_N(:,1) = RR(1).^ks';
%
% fill the remaining of the matrix
for n=2:N
    for k=1:K
        G_K_N(k+1, n) = G_K_N(k+1, n-1) + RR(n)*G_K_N(k, n);
    end
end
%
% Mean numbers
for i=1:N
    Kmean(i) = sum(RR(i).^(1:K)' .* G_K_N(K:-1:1,N))/...
                   G_K_N(K+1,N);
end
Omega = RFlows*G_K_N(K,N)/G_K_N(K+1,N);
Dmean = Kmean./Omega;
```

Same as $G(k)$ computed by the MVA algorithm

```
>> Example_MVA_CNb
>> G_K_N

G_K_N =

    1.0000    1.0000    1.0000    1.0000
    0.0200    0.1000    0.1800    0.2400
    0.0004    0.0084    0.0228    0.0372
    0.0000    0.0007    0.0025    0.0047
    0.0000    0.0001    0.0003    0.0005
    0.0000    0.0000    0.0000    0.0001
    0.0000    0.0000    0.0000    0.0000

>> Kmean

Kmean =

    0.2436    2.2610    2.2610    1.2343

>> Dmean

Dmean =

    0.0246    0.5698    1.1397    1.2443
```


BCMP Networks

- **BCMP = Baskette, Chandy, Muntz, Palacios = 1975 paper**
- **Generalization of the product forms obtained for Jackson networks for FCFS**
- **The product form holds (in addition to FCFS) for**
  - **Infinite server model: a message immediately assigned a server as soon as it enters the system – all messages are simultaneously in service**
  - **Processor sharing: each message in the queue receives equal simultaneous service – all messages are simultaneously in service**
  - **Preemptive resume last-come first-served: newly arrived messages are served immediately – displaced messages are requeued and resume server only when the server is available again**
- **The product form holds also for arbitrary service times too!**

BCMP Networks – Single Node

- **Exponential Server**
- **C classes of users**
- **The joint pmf can be shown to be**

$$P\left(N_1 = n_1, N_2 = n_2, \ldots, N_C = n_C, \sum_{i=1}^{C} n_i = m\right) = (1-\rho)m! \prod_{i=1}^{C} \frac{\rho_i^{n_i}}{n_i!}$$

- **Note the product form holds for multiple classes in a single node network!!**

BCMP Networks – Single Node (Infinite Server)

- **For one class of customers, we have shown before that the marginal distribution is given by**

$$P(Q = m) = \frac{e^{-\rho}\rho^m}{m!}; \quad m = 0,1,2,\ldots$$

where $\rho = \Sigma\lambda_i/\mu$ - $\rho_i = \lambda_i/\mu$ for i=1,2, …, C

- **What would be the distribution for an arbitrary service time?**
  - **Use K-stage Cox network to model server**

BCMP Networks – Single Node (Infinite Server) – cont'd

- **We can show that the number of customers marginal distribution is still given by the previous expression even for the K-stage Cox network server**
- **This is referred to by the *insensitivity* property for the infinite-server case**

BCMP Networks – Single Node (Infinite Server) – cont'd

- **The previous result can be extended to C classes of user (similar to the exponential server case)**

$$P(N_1 = n_1, N_2 = n_2, \ldots, N_C = n_C) = e^{-\rho} \prod_{i=1}^{C} \frac{\rho_i^{n_i}}{n_i!}; \quad \forall n_i \geq 0$$

- **The probability distribution of the total number of customers of all classes in the node is**

$$P\left(\sum_{i=1}^{C} n_i = m\right) = \frac{e^{-\rho}\rho^m}{m!}; \quad m \geq 0$$

BCMP Networks – Single Node (Processor Sharing)

- **Server model – K-stage Cox network**
- **Using the same approach we learned in analysis (global balance equation ➔ detailed ➔ pmf), we can show that for C classes of users, the joint pmf is given by**

$$P(N_1 = n_1, N_2 = n_2, \ldots, N_C = n_C) = (1-\rho) m! \prod_{i=1}^{C} \frac{\rho_i^{n_i}}{n_i!}$$

where $\rho = \Sigma\lambda_i/\mu = \Sigma\, \rho_i$ – and $\Sigma n_i = m$

- **This is the same form for the single node exponential server with C classes of customers!!**

BCMP Networks – Single Node (Last-Come First-Served)

- **Server model – K-stage Cox network**
- **Again we can show that for C classes of users, the joint pmf is given by**

$$P(N_1 = n_1, N_2 = n_2, \ldots, N_C = n_C) = (1-\rho) m! \prod_{i=1}^{C} \frac{\rho_i^{n_i}}{n_i!}$$

where $\rho = \Sigma\lambda_i/\mu = \Sigma\, \rho_i$ – and $\Sigma n_i = m$

- **This is the same form for the single node exponential server and the processor-sharing server with C classes of customers!!**

BCMP Networks – Single Node - Summary

- **We can summarize the results for the single node and C classes of customers as**

$$P(N_1 = n_1, N_2 = n_2, \ldots, N_C = n_C) = \begin{cases} (1-\rho) m! \prod_{i=1}^{C} \rho_i^{n_i} / n_i! & \text{FCFS – exponential service} \\ (1-\rho) m! \prod_{i=1}^{C} \rho_i^{n_i} / n_i! & \text{PS – arbitrary service} \\ e^{-\rho} \prod_{i=1}^{C} \rho_i^{n_i} / n_i! & \text{Infinite servers – arbitrary service} \\ (1-\rho) m! \prod_{i=1}^{C} \rho_i^{n_i} / n_i! & \text{Preemptive LCFS – arbitrary service} \end{cases}$$

$\rho = \Sigma\lambda_i/\mu = \Sigma\, \rho_i$ – and $\Sigma n_i = m$

Network of BCMP Queues

- **N nodes**
  - **Probabilistic routing**
  - **Each is one of the four types considered by BCMP nodes**
- **The previous product forms for open and closed networks still apply**
- **Further more, let nodes 1,2, …, i belong to FCFS, PS, or LCFS while nodes i+1,i+2, …, N belong to the infinite server nodes, then the joint pmf is given by**

$$P(n_1, n_2, \ldots, n_N) = \prod_{j=1}^{i} (1 - R_j) R_j^{n_j} \prod_{k=i+1}^{N} \frac{e^{-R_k} R_k^{n_k}}{n_k!}$$

Network of BCMP Queues – cont'd

- **N nodes**
  - **Probabilistic routing**
  - **Each is one of the four types considered by BCMP nodes**
- **The previous product forms for open and closed networks still apply**
- **Further more, let nodes 1,2, …, i belong to FCFS, PS, or LCFS while nodes i+1,i+2, …, N belong to the infinite server nodes, then the joint pmf is given by**

$$P(n_1, n_2, \ldots, n_N) = \prod_{j=1}^{i} (1 - R_j) R_j^{n_j} \prod_{k=i+1}^{N} \frac{e^{-R_k} R_k^{n_k}}{n_k!}$$