MODULE 9: Queueing Networks (contd...)

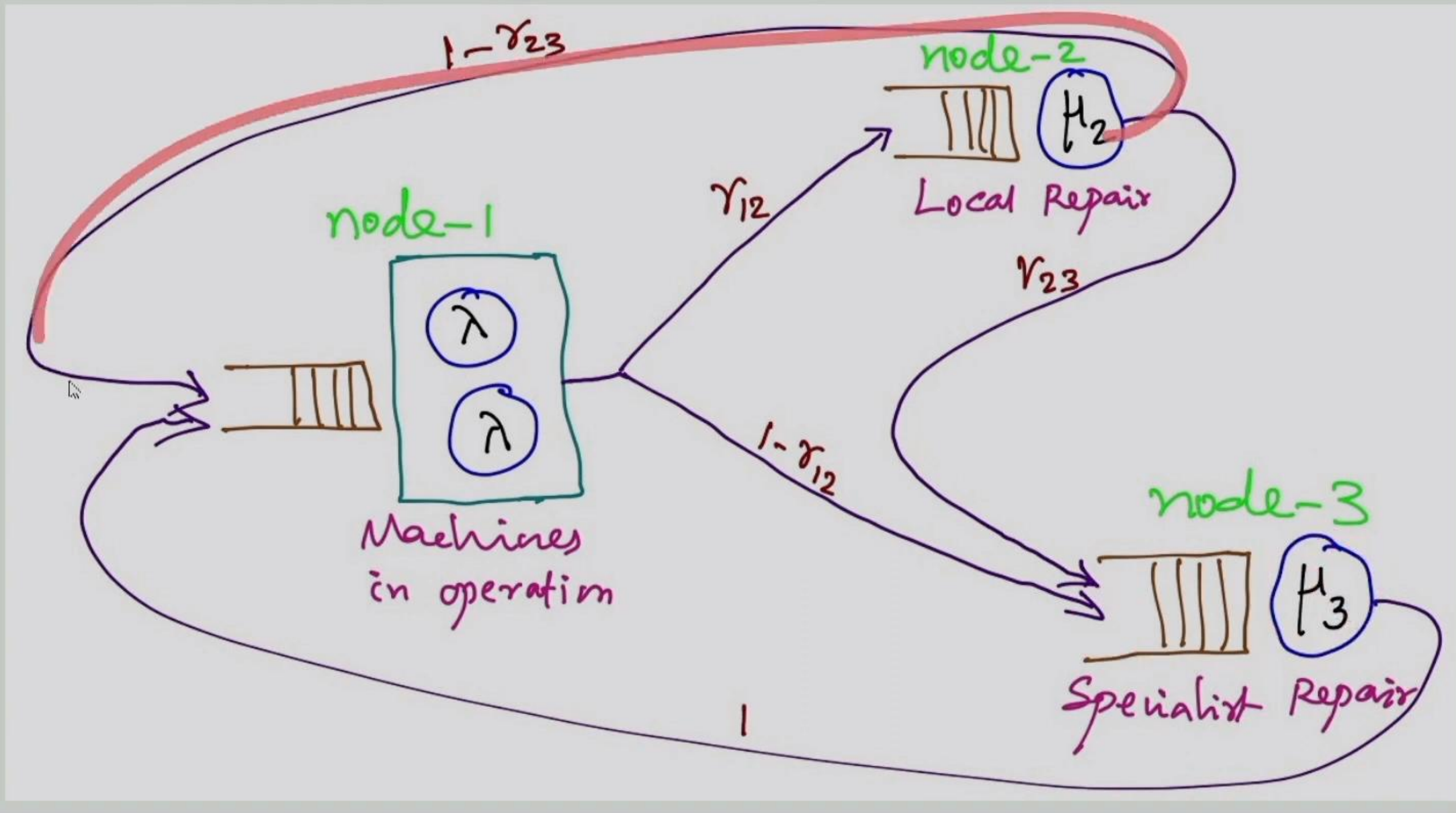
LECTURE 33

Closed Jackson Networks, Convolution Algorithm

Example (Two-Machine Three-Node Closed Network)

- Two special-purpose machines are in operating condition and they need to be maintained in that position at all times. The machines break down according to an $Exp(\lambda)$ distribution.
  - Call this operating node as node 1.
- Upon failure, a machine
  - has a probability of r_{12} being repaired locally (node 2) by a single repairman with repair times following an $Exp(\mu_2)$ distribution, or
  - must be repaired by the single specialist (node 3) with probability $1 - r_{12}$ who works according to an $Exp(\mu_3)$ distribution.
- After the service completion locally, the machine may require specialist attention with probability r_{23}, or return to operation with probability $1 - r_{23}$.
- After the special service (node 3), the unit always returns to operation $(r_{31} = 1)$.
- Here at node 1, the servers are machines, so $c_1 = 2$ with the mean service (or holding time) at node 1 is the mean time to failure of a machine. This means that $\mu_1 = \lambda$.

Example


1 − r23
node-2
μ2
Local Repair
r12
node-1
λ
λ
Machines in operation
r23
1 − r12
node-3
μ3
Specialist Repair
1

Example

- The steady state joint probability distribution is given by

$$p_{n_1,n_2,n_3} = \frac{1}{G(2)} \frac{\rho_1^{n_1}}{a_1(n_1)} \rho_2^{n_2} \rho_3^{n_3}, \qquad n_i = 0, 1, 2; \quad i = 1, 2, 3,$$

where $a_1(n_1) = 1$ for $n_1 = 0, 1$ and $a_1(2) = 2$. We must find ρ_i from $\mu_i \rho_i = \sum_{j=1}^{k} \mu_j r_{ji} \rho_j$.

Here, the routing matrix R is

$$R = \begin{pmatrix} 0 & r_{12} & 1 - r_{12} \\ 1 - r_{23} & 0 & r_{23} \\ 1 & 0 & 0 \end{pmatrix}$$

and hence the traffic equations becomes

$$\begin{aligned} \lambda \rho_1 &= \mu_2 (1 - r_{23}) \rho_2 + \mu_3 \rho_3 \\ \mu_2 \rho_2 &= \lambda r_{12} \rho_1 \\ \mu_3 \rho_3 &= \lambda (1 - r_{12}) \rho_1 + \mu_2 r_{23} \rho_2 \end{aligned}$$

09_QNetworks2.pdf

14/ 28

Example

Since these equations are linearly dependent, as we know already, we can set one of the ρ_i's to 1 and solve for the rest.

Set $\rho_2 = 1$. Then, from the second equation, we get $\rho_1 = \frac{\mu_2}{r_{12}\lambda}$. Substituting these two into the third equation, we get

$$\rho_3 = \frac{\lambda(1-r_{12})}{\mu_3}\frac{\mu_2}{\lambda r_{12}} + \frac{\mu_2}{\mu_3}r_{23} = \frac{\mu_2(1-r_{12}+r_{12}r_{23})}{r_{12}\mu_3}.$$

We thus have the steady state solution for the closed network as

$$p_{n_1,n_2,n_3} = \frac{1}{G(N)}\left(\frac{\mu_2}{r_{12}\lambda}\right)^{n_1}\frac{1}{a_1(n_1)}\left(\frac{\mu_2(1-r_{12}+r_{12}r_{23})}{r_{12}\mu_3}\right)^{n_3}, \quad n_1, n_3 = 0, 1, 2.$$

The normalizing constant $G(N)$ can be obtained by summing p_{n_1,n_2,n_3} over all cases for which $n_1 + n_2 + n_3 = 2$.

♦ There are six cases in total: $(2,0,0), (0,2,0), (0,0,2), (1,1,0), (1,0,1), (0,1,1)$.

Example (Illustration)

- Assume $\lambda = 2,\ \mu_2 = 1,\ \mu_3 = 3,\ r_{12} = \frac{3}{4},\ r_{23} = \frac{1}{3}$. Then, the joint distribution is

$$p_{n_1,n_2,n_3} = \frac{1}{G(N)}\left(\frac{2}{3}\right)^{n_1}\frac{1}{a_1(n_1)}\left(\frac{2}{9}\right)^{n_3},$$

where $G(2)$ is computed to be

$$G(2) = \left(\frac{2}{3}\right)^2\frac{1}{2} + 1 + \left(\frac{2}{9}\right)^2 + \frac{2}{3} + \frac{2}{3}\frac{2}{9} + \frac{2}{9} = \frac{187}{81} = 2.3086.$$

$\bar{n}$	(2,0,0)	(0,2,0)	(0,0,2)	(1,1,0)	(1,0,1)	(0,1,1)
$p_{\bar{n}}$	0.0962	0.4332	0.0214	0.2888	0.0642	0.0962

- Only 9.62% of the time, both machines are operating.
- At least one machine available for 44.92% of the time.
- Performance not up to the mark. Decide how to improve!

Buzen's Algorithm or Convolution Algorithm

- In Gordon-Newell networks, the joint probability distribution is determined in terms of the normalization constant $G(N)$.
- In the examples considered so far, the 'naive computation' approach of calculating $G(N)$ was easy. But, this is not the case in general.
- For large N and k (i.e., for large networks), there are many possible ways to allocate N customers among the k nodes (it is actually $\binom{N+k-1}{k-1}$ ways which is of order N^{k-1}).
  - ▶ Calculations also become prone to numerical errors.
- Efficient algorithms are needed to make the computation of $G(N)$ easier (and less prone to numerical errors).
- Buzen (1973) developed an efficient algorithm to compute $G(N)$ recursively, using a total of Nk multiplications and Nk additions (single server case - which is a significant improvement).
  - ▶ Very useful for larger networks.

- We will now describe Buzen's Algorithm or Convolution Algorithm.
- Let $f_i(n_i) \doteq \frac{\rho_i^{n_i}}{a_i(n_i)}$ where $a_i(n_i) = \begin{cases} n_i! & n_i < c_i \\ c_i! c_i^{n_i - c_i} & n_i \geq c_i \end{cases}$.

  Then, the normalization constant $G(N)$ can be written as

$$G(N) = \sum_{n_1+n_2+\cdots+n_k=N} \prod_{i=1}^{k} f_i(n_i).$$

- Now, define an auxiliary function

$$g_m(n) = \sum_{n_1+n_2+\cdots+n_m=n} \prod_{i=1}^{m} f_i(n_i). \qquad \text{(i.e., with } m \text{ nodes } \& \ n \text{ customers).}$$

- Observe that we have $G(N) = g_k(N)$. We now set up a recursive scheme to calculate $G(N)$.

- Take $g_m(n)$ and fix $n_m = i$. Then, we have

$$
\begin{aligned}
g_m(n) &= \sum_{i=0}^{n} \left(\sum_{n_1+\cdots+n_{m-1}+i=n} \prod_{j=1}^{m} f_j(n_j) \right) \\
&= \sum_{i=0}^{n} f_m(i) \left(\sum_{n_1+\cdots+n_{m-1}=n-i} \prod_{j=1}^{m-1} f_j(n_j) \right) \\
&= \sum_{i=0}^{n} f_m(i) g_{m-1}(n-i), \qquad n = 0, 1, \ldots, N.
\end{aligned}
$$

- Note from the above that $g_1(n) = f_1(n)$, for $n = 0, 1, 2, \ldots, N$, and $g_m(0) = 1$, for $m = 1, 2, \ldots, k$. These form the starting conditions.
- The above relationship of the auxiliary function can then be recursively used to calculate

$$G(N) = g_k(N).$$

- The algorithm is efficient for large networks.

- These functions also helps us in calculating the marginal distributions as well.
- Suppose that we want $p_i(n) = P\{N_i = n\}$.
  Let $S_i = n_1 + n_2 + \cdots + n_{i-1} + n_{i+1} + \cdots + n_k$. Then

$$p_i(n) = \sum_{S_i = N-n} p_{n_1, n_2, \ldots, n_k} = \sum_{S_i = N-n} \frac{1}{G(N)} \prod_{i=1}^{k} f_i(n_i)$$

$$= \frac{f_i(n)}{G(N)} \sum_{S_i = N-n} \prod_{j=1, j \neq i}^{k} f_j(n_j), \qquad n = 0, 1, 2, \ldots, N$$

In general, this may be cumbersome to compute. But for node k, the expression simplifies to

$$p_k(n) = \frac{f_k(n)}{G(N)} \sum_{S_k = N-n} \prod_{j=1}^{k-1} f_j(n_j) = \frac{f_k(n) g_{k-1}(N-n)}{G(N)}, \qquad n = 0, 1, 2, \ldots, N$$

To find other marginals, permute the node of interest with k (requires resolving some of the functions $g_m(n)$).

Example (Two-Machine Three-Node Closed Network - Illustration - Revisited)

- First, the factors $f_i(n_i)$ are

$$f_1(0) = 1, f_1(1) = \frac{2}{3}, f_1(2) = \frac{2}{9}, \;\; f_2(0) = f_2(1) = f_2(2) = 1, \;\; f_3(0) = 1, f_3(1) = \frac{2}{9}, f_3(2) = \frac{4}{81}.$$

- The $g_m(n)$'s are given by

$$G(2) = g_3(2) = \sum_{i=0}^{3} f_3(i) g_2(2-i) = f_3(0)g_2(2) + f_3(1)g_2(1) + f_3(2)g_2(0)$$

and

$$g_2(2) = f_2(0)g_1(2) + f_2(1)g_1(1) + f_2(2)g_1(0)$$
$$g_2(1) = f_2(0)g_1(1) + f_2(1)g_1(0)$$

with the starting conditions

$$g_1(0) = f_1(0) = 1, g_1(1) = f_1(1) = \frac{2}{3}, g_1(2) = f_1(2) = \frac{2}{9}, \;\; g_m(0) = 1, m = 1, 2, 3.$$

Example

- Calculations give us

$$g_2(1) = 1 \cdot \frac{2}{3} + 1 \cdot 1 = \frac{5}{3}$$

$$g_2(2) = 1 \cdot \frac{2}{9} + 1 \cdot \frac{2}{3} + 1 \cdot 1 = \frac{17}{9}$$

$$g_3(2) = 1 \cdot \frac{17}{9} + \frac{2}{9} \cdot \frac{5}{3} + \frac{4}{81} \cdot 1 = \frac{187}{81} = 2.3086. = G(2)$$

[Also, $g_3(1) = f_3(0)g_2(1) + f_3(1)g_2(0) = 1 \cdot \frac{5}{3} + \frac{2}{9} \cdot 1 = \frac{17}{9}$.]

$g_m(n)$	$m=1$	$m=2$	$m=3$
$n=0$	1	1	1
$n=1$	$\frac{2}{3}$	$\frac{5}{3}$	$\frac{17}{9}$
$n=2$	$\frac{2}{9}$	$\frac{17}{9}$	$\frac{187}{81}$

- This example (where the algorithm is not much simpler) is to only illustrate the algorithm. The efficiency will be evident when you consider larger networks.