Performance Modeling using Elementary Queuing Theory

- **Birth-Death Process**
  - $M/M/1$ queue
  - $M/M/m$ queue
  - $M/M/1/n$ queue
  - Cyclic Queuing Model
  - Response Time in a Client-Server System
  - Wireless Handoff Model

- **Non-Birth-Death Process**
  - $M/M/2$ Heterogeneous queue
  - $MMPP$ Arrival Process
  - $MMPP/M/1$ queue

Copyright © 2006 by K.S. Trivedi
Queueing Theory Pioneers

A.K. Erlang (1909—1920);
E.C. Molina (1908); W.H. Grinsted (1907);
T. Engset (1918)

C. Palm (1907-1951)
Continuous Time Birth-Death Process

• A CTMC \( \{X(t)|t \geq 0\} \) with \( I=\{0,1,2,\ldots\} \) forms a B-D process, if \( \lambda_i, i=0,1,2,\ldots \) and \( \mu_i, i=1,2,\ldots \) exist, such that the transition rates are given by

\[
q_{i,i+1} = \lambda_i,
\]

\[
q_{i,i-1} = \mu_i,
\]

\[
q_i = \lambda_i + \mu_i
\]

\[
q_{ij} = 0 \text{ for } |i-j| > 1.
\]

given Birth rate \( \lambda_i \geq 0 \) and Death rate \( \mu_i \geq 0 \)
Continuous Time Birth-Death Process (contd.)

The state diagram of the birth-death process is given as

\[
\begin{array}{cccccc}
0 & 1 & 2 & \cdots & k-1 & k \\
\lambda_0 & \lambda_1 & \lambda_{k-1} & \lambda_k & \mu_1 & \mu_2 & \cdots & \mu_k & \mu_{k+1} \\
\mu_1 & \mu_2 & \cdots & \cdots & \mu_{n-1} & \mu_n & \cdots & \mu_{n+1} & \cdots
\end{array}
\]

The generator matrix \( Q \) can be shown to be as

\[
Q = [q_{i,j}] = \\
\begin{bmatrix}
-\lambda_0 & \lambda_0 & 0 & 0 & 0 & \cdots & \cdots & 0 \\
\mu_1 & -(\mu_1 + \lambda_1) & \lambda_1 & 0 & 0 & \cdots & \cdots & 0 \\
0 & \mu_2 & -(\mu_2 + \lambda_2) & \lambda_2 & 0 & \cdots & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & \cdots & \mu_{n-2} & -(\mu_{n-2} + \lambda_{n-2}) & \lambda_{n-2} \\
0 & 0 & 0 & \cdots & \cdots & 0 & \mu_{n-1} & -(\mu_{n-1} + \lambda_{n-1}) & \lambda_{n-1} \\
0 & 0 & 0 & \cdots & \cdots & 0 & 0 & \mu_n & -(\mu_n + \lambda_n) & \lambda_n \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\]
Continuous Time Birth-Death Process (contd.)

\[ \frac{d\pi(t)}{dt} = \pi(t)Q \] (Need to solve this differential eq. to get \( \pi(t) \))

\[ \frac{d\pi_k(t)}{dt} = -(\lambda_k + \mu_k)\pi_k(t) + \lambda_{k-1}\pi_{k-1}(t) + \mu_{k+1}\pi_{k+1}(t), \quad k \geq 1 \]

\[ \frac{d\pi_0(t)}{dt} = -\lambda_0\pi_0(t) + \mu_1\pi_1(t), \quad k = 0, \]

We are interested for now in steady-state solution (we will consider transient solution in module 5). So we set the derivatives to zero,

\[ \frac{d\pi_k(t)}{dt} = 0 \]
Steady State Balance Equations

\[
0 = -(\lambda_k + \mu_k)\pi_k + \lambda_{k-1}\pi_{k-1} + \mu_{k+1}\pi_{k+1}, \quad k \geq 1, \\
0 = -\lambda_0\pi_0 + \mu_1\pi_1
\]

These are called balance eqns. Re-arranging above,

\[
\lambda_k\pi_k - \mu_k+1\pi_{k+1} = \lambda_{k-1}\pi_{k-1} - \mu_k\pi_k = \cdots = \lambda_0\pi_0 - \mu_1\pi_1
\]

Therefore, \( \lambda_{k-1}\pi_{k-1} - \mu_k\pi_k = 0 \Rightarrow \pi_k = \frac{\lambda_{k-1}}{\mu_k}\pi_{k-1}, \quad k \geq 0 \)

or, \( \pi_k = \frac{\lambda_0\lambda_1 \cdots \lambda_{k-1}}{\mu_1\mu_2 \cdots \mu_k}\pi_0 = \pi_o \prod_{i=0}^{k-1} \left( \frac{\lambda_i}{\mu_i+1} \right) \)

\[\checkmark \lambda_i \text{ and } \mu_i \text{ should be } > 0 \text{ for irreducibility condition, which means every state can be reached from every other state.}\]
Steady State Balance Equations (contd.)

Invoking total prob relation, \( \sum_{k \geq 0} \pi_k = 1 \) gives

\[
\pi_0 = \frac{1}{1 + \sum_{k \geq 1} \prod_{i=0}^{k-1} \left( \frac{\lambda_i}{\mu_i+1} \right)}
\]

- Note that limiting state probability vector \([\pi_0, \pi_1, \ldots]\) is now completely determined and are non zero provided \( \sum_{k \geq 1} \prod_{i=0}^{k-1} \left( \frac{\lambda_i}{\mu_i+1} \right) \) converges which also means that all states of the CTMC are recurrent non-null (also called positive recurrent).
CTMC State Classification

• State classification may be based on the distinction that:
  – Average number of visits to some states may be infinite
    while other states may be visited only a finite number of
    times (on average)

• **Transient state**: *if* there is non-zero probability that the
  system will **NOT** return to this state (or the average number
  of visits is finite).

• State *i* is a said to be **recurrent** *if*, starting from state *i*, the
  process eventually returns to the state *i* with probability 1.

• If mean time to return to a recurrent state is finite then it is a
  recurrent non-null state; otherwise recurrent null.
Definitions

- A CTMC is said to be *ir reducible* if every state can be reached from every other state, with a non-zero probability. In a finite state irreducible CTMC all states are positive recurrent.
- A state is said to be *absorbing* if no other state can be reached from it with non-zero probability.
- Notion of *transient, recurrent non-null, recurrent null* are the same as in a DTMC. There is no notion of periodicity in a CTMC, however.
- Unless otherwise specified, when we say CTMC, we mean HCTMC.
Communicating classes

- **Communicating** states: \( i \) and \( j \) are said to be communicating if there exist directed paths from \( i \) and \( j \) and from \( j \) and \( i \).
- **Closed** set of states: A commutating set of states \( C \) forms a closed set, if no state outside of \( C \) can be reached from any state in \( C \).
State Classification of CTMC
M/M/1 Queue

- Arrivals process is Poisson, i.e., *interarrival* times are all *i.i.d*, \(\text{EXP}(\lambda)\).

- Service times are *i.i.d*, \(\text{EXP}(\mu)\).

- Define, \(\rho = \frac{\lambda}{\mu}\) (traffic intensity, in Erlangs)

\[
\begin{align*}
\lambda & \quad | & \lambda & \quad | & \lambda & \quad | & \lambda & \quad | & \lambda \\
0 & \quad \mu & \quad | & 1 & \quad \mu & \quad | & 2 & \quad \mu & \quad | & \cdots & \quad \mu \\
\end{align*}
\]
M/M/1 queue (contd.)

- From the balance equations, we get

\[ \pi_k = \left( \frac{\lambda}{\mu} \right)^k \pi_0 = \rho^k \pi_0 \]

Invoking total prob relation, \( \sum_{k \geq 0} \pi_k = 1 \) gives

and \( \pi_0 = \frac{1}{\sum_{k \geq 0} \rho^k} = 1 - \rho \), \( \rho < 1 \) for stable system

or, \( \pi_k = (1 - \rho)\rho^k, \ k \geq 0 \) (Modified Geometric pmf)

- If \( \lambda > \mu \) then all states of the CTMC are transient
- If \( \lambda = \mu \) then all states of the CTMC are null recurrent
- If \( \lambda < \mu \) then all states of the CTMC are recurrent non-null
M/M/1 queue (contd.)

• $U_o = 1 - \pi_o = \rho$ is known as server utilization and is interpreted as the proportion of time the server is busy

• Expected # of customers,

$$E[N] = \sum_{k=0}^{\infty} k \pi_k = \frac{\rho}{1 - \rho}$$

$$Var[N] = \sum_{k=0}^{\infty} k^2 \pi_k - (E[N])^2 = \sum_{k=0}^{\infty} (k^2 - (E[N])^2) \pi_k = \frac{\rho}{(1 - \rho)^2}$$
M/M/1 queue (contd.)

• This measure \( E[N] \) can be viewed as a weighted average,

\[
\sum_{k=0}^{\infty} r_k \pi_k
\]

• By choosing suitable weights for the states of a CTMC, we can get most measures of interest and the resulting model is known as an MRM (Markov Reward Model). Above weighted average is the expected reward rate in the steady state.

• Other measures:
  – Average queue length \( E[Q] \)
  – Average response time
  – Average waiting time (avg. time spent in the queue before service begins) etc.
**M/M/1 queue: Little’s formula**

- Let the random variable $R$ denote the response time (defined as the time elapsed from the instant of job arrival until its completion).

  Little’s law states
  \[
  E[R] = E[N]/\lambda = \sum_{k=0}^{\infty} \frac{k}{\lambda} \pi_k
  \]

- Here
  \[
  E[R] = \frac{E[N]}{\lambda} = \sum_{k=0}^{\infty} \frac{k}{\lambda} \pi_k = \lambda^{-1}(1 - \rho) \sum_{k=0}^{\infty} \frac{k\rho^k}{\rho/(1 - \rho)^2}
  \]
  
  \[
  = \lambda^{-1}(1 - \rho) \frac{\rho}{(1 - \rho)^2} = \frac{\rho}{\lambda(1 - \rho)} = \frac{\mu^{-1}}{1 - \rho} = \frac{\mu^{-1}}{\pi_0} = \text{av. service time} / \text{server idle prob.}
  \]

- Response time ($R$) = waiting time ($W$) + service time ($S$)
  
  \[
  E[W] = E[R] - E[S] = 1/(\mu(1-\rho)) - 1/\mu
  \]
M/M/1 queue: Thruput

- You may conclude by mistake that the thruput is $\mu$
- But in state 0 the thruput is 0; otherwise it is $\mu$
- Hence the expected thruput, $E[T]$ is

$$E[T] = \sum_{i=1}^{\infty} \mu \pi_i = \mu \sum_{i=1}^{\infty} \pi_i = \mu(1 - \pi_0) = \mu \rho = \lambda$$

- All the measures so far are examples of expected reward rate in steady state, $\sum_{i} r_i \pi_i$, with the appropriate reward assignment as in the table on next slide
Reward rate assignment for Measures for $M/M/1$ queue

<table>
<thead>
<tr>
<th></th>
<th>$r_i = i$</th>
<th>$\frac{\rho}{1 - \rho}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean no. in the system</strong></td>
<td>$E[N]$</td>
<td></td>
</tr>
<tr>
<td><strong>Mean Response time $E[R]$</strong></td>
<td>$r_i = i/\lambda$</td>
<td>$\frac{1/\mu}{1 - \rho}$</td>
</tr>
<tr>
<td><strong>Throughput $E[T]$</strong></td>
<td>$r_0 = 0$</td>
<td>$\lambda$</td>
</tr>
<tr>
<td></td>
<td>$r_i = \mu$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$i &gt; 0$</td>
<td></td>
</tr>
</tbody>
</table>
Response time distribution (Tagged job approach)

• With FCFS scheduling and steady-state conditions
  – If there are already \( n \) jobs in the system, the next job \((n+1)^{st}\) will experience a response time \( R = S^* + S'_1 + S_2 + \ldots + S_n \)
  – \( S^* \): service time for the \((n+1)^{st}\) job; \( S'_1 \): residual service time for job currently undergoing service (#1).
  – Because of the memoryless property, these times are \( \text{EXP}(\mu) \).
  – Hence, for \( N=n \), the conditional LST of \( R \) is,

\[
L_{R|N}(s|n) = \left( \frac{\mu}{s + \mu} \right)^{n+1}
\]
Response time distribution (Tagged job approach)

- With FCFS scheduling and steady-state conditions

  - Note for a Poisson arrival process, it is known that, the state of the system as seen by an arriving customer is statistically the same as the state of the system at an arbitrary time. This Property is known as PASTA – Poisson Arrival See Time Average. Hence un-conditioning using

    \[ \pi_n = (1 - \rho)\rho^n, \quad (\pi_n: \text{prob. of exactly } n \text{ jobs in the system}). \]

    or, \[ L_R(s) = \sum_{n=0}^{\infty} \left( \frac{\mu}{s + \mu} \right)^{n+1} (1 - \rho)\rho^n = \frac{\mu(1 - \rho)}{s + \mu(1 - \rho)} \]

  - Then, taking inverse transform,

    Hence \( R \) is exponentially distributed:

    \[ R \sim \text{EXP}(\mu(1 - \rho)) \]
M/M/m queue

- *m*-servers service the queue.

\[ \lambda_k = \lambda, \quad k = 0, 1, 2, \ldots, \]

\[ \mu_k = \begin{cases} k\mu & k = 0, 1, 2, \ldots, m \\ m\mu & k > m \end{cases} \]
M/M/m Queue Solution

\[ \pi_k = \pi_0 \prod_{i=0}^{k-1} \left( \frac{\lambda_i}{\mu_i + 1} \right) \Rightarrow \pi_0 \prod_{i=0}^{k-1} \left( \frac{\lambda_i}{(i + 1) \mu} \right) = \frac{\pi_0}{k!} \left( \frac{\lambda}{\mu} \right)^k, \ k \leq m \]

\[ \pi_k = \pi_0 \prod_{i=0}^{m-1} \frac{\lambda_i}{(i + 1) \mu} \prod_{j=m}^{k-1} \frac{\lambda_j}{m \mu} = \frac{\pi_0}{m! m^{k-m}} \left( \frac{\lambda}{\mu} \right)^k, \ k \geq m \]

Letting, \( \rho = \lambda / m \mu \) (\( \rho < 1 \)), and \( \sum_{k=0}^{\infty} \pi_k = 1 \),

\[ \pi_0 = \left[ \sum_{k=0}^{m-1} \frac{(m \rho)^k}{k!} + \frac{(m \rho)^m}{m!} \cdot \frac{1}{1 - \rho} \right]^{-1} \]
M/M/m Queue performance measures

• Average number in system $E[N]$:

$$E[N] = \sum_{k=1}^{\infty} k \pi_k = m \rho + \frac{(m \rho)^m}{m!} \frac{\pi_0}{(1 - \rho)^2}$$

$$E[N] = \sum_{k=0}^{\infty} k \pi_k = \sum_{k=1}^{\infty} k \pi_k = \pi_0 \left[ \sum_{k=1}^{m} \frac{(m \rho)^k}{(k-1)!} + \sum_{k=m+1}^{\infty} \frac{(m \rho)^k k}{m! m^{k-m}} \right]$$

(Letting $k = m + j$)

$$E[N] = \pi_0 \left[ (m \rho) \sum_{k=0}^{m-1} \frac{(m \rho)^k}{k!} + \frac{(m \rho)^m}{m!} \sum_{j=1}^{\infty} \frac{(m \rho)^j (m + j)}{m^j} \right]$$

$$E[N] = \pi_0 \left[ m \rho \sum_{k=0}^{m-1} \frac{(m \rho)^k}{k!} + \frac{(m \rho)^m}{m!} \left( \sum_{j=1}^{\infty} m \rho^j + \sum_{j=1}^{\infty} j \rho^j \right) \right]$$

$$E[N] = \pi_0 \left[ m \rho \left( \sum_{k=0}^{m-1} \frac{(m \rho)^k}{k!} + \frac{(m \rho)^m}{m!} \frac{1}{1 - \rho} \right) + \frac{(m \rho)^m}{m!} \frac{\rho}{(1 - \rho)^2} \right]$$

$$E[N] = m \rho + \frac{(m \rho)^m}{m!} \frac{\pi_0}{(1 - \rho)^2}$$
M/M/m Queue performance measures

• Server utilization: \( rv M \) - number of busy servers. For number of customers \( 0 \leq k \leq m \), the number of busy servers = \( k \). Beyond that the number of busy servers = \( m \).

\[
P(M = k) = \begin{cases} 
P(N = k) = \pi_k, & 0 \leq k \leq m - 1, \\ 
P(N \geq m) = \sum_{k=m}^{\infty} \pi_k = \frac{\pi_m}{1-\rho}, & k = m. \\
\end{cases}
\]

Av. of busy servers = \( E[M] = \sum_{k=0}^{m-1} k\pi_k + \frac{m\pi_m}{1-\rho} = m\rho = \frac{\lambda}{\mu} \)

• A customer may have to join the queue if \( \# \) customers already in the system \( \geq m \)

\[
P(\text{queuing}) = \sum_{k=m}^{\infty} \pi_k = \frac{\pi_m}{1-\rho} = \frac{(m\rho)^m}{m!} \cdot \frac{\pi_0}{1-\rho}, \text{ (Erlang’s C formula)}
\]
Separate or common queue?

• M/M/2.
• 2 cases
  – Case 1: Two independent queues
    
    Two separate Poisson streams $\rightarrow$ 2 separate $M/M/1$ queues

  – Case 2: $M/M/2$ case
    
    Two separate Poisson streams $\rightarrow$ Combined Poisson streams

Copyright © 2006 by K.S. Trivedi
Comparative performance

• Case 1: For each $M/M/1$ queue,

$$\rho = \lambda / 2\mu \quad E[R_s] = \frac{\text{Av. service time}}{\text{server idle prob.}} = \frac{1/\mu}{1 - \lambda/2\mu} = \frac{2}{2\mu - \lambda}$$

• Case 2: Common queue $M/M/2$

$$E[R_c] = \frac{\text{Av. queue length}}{\lambda} = \frac{E[N_c]}{\lambda} \quad \text{(Little's formula)}$$

$$E[N_c] \quad (\text{using } E[N] \text{ for } m = 2) = 2\rho + 2\rho^3 \frac{1 - \rho}{(1 + \rho)(1 - \rho)^2} = \frac{2\rho}{1 - \rho^2}$$

or, $$E[R_c] = \frac{E[N_c]}{\lambda} = \frac{2\cdot\frac{1}{2\mu}}{1 - \left(\frac{\lambda}{2\mu}\right)^2} = \frac{1}{\mu(1 - \rho^2)} = \frac{4\mu}{4\mu^2 - \lambda^2}.$$  

comparing $R_s$ and $R_c$,

$$E[R_s] = \frac{2}{2\mu - \lambda} = \frac{4\mu + 2\lambda}{4\mu^2 - \lambda^2} > E[R_c].$$
**M/M/1/n Queue**

- Finite queue size, finite buffer space $\rightarrow$ finite state space.

$$\lambda_i = \lambda, \ 0 \leq i \leq n-1; \ \mu_i = \mu, \ 1 \leq i \leq n; \ \rho = \lambda / \mu.$$ 

\[
Q = \begin{bmatrix}
-\lambda & \lambda & & & \\
\mu & -(\lambda + \mu) & \lambda & & \\
\mu & -(\lambda + \mu) & \lambda & & \\
\mu & -(\lambda + \mu) & \lambda & & \\
\mu & -(\lambda + \mu) & \lambda & & \\
\end{bmatrix}
\]

- Steady State Solution:

$$\pi_i = \rho^i \pi_0, \ 0 \leq i \leq n$$

$$\pi_0 = \frac{1}{\sum_{i=0}^{n} \rho^i} = \begin{cases} 
\frac{1-\rho}{1-\rho^{n+1}}, & \rho \neq 1 \\
\frac{1}{n+1}, & \rho = 1 
\end{cases}$$

- Note that given irreducibility ($\lambda_i$ and $\mu_i$ should be $>0$), being a finite CTMC all states are recurrent non-null and hence there is no additional condition for stability.
\section*{M/M/1/n Queue Performance Measures}

- **Expected # of jobs in the system:**
  \[ E[N] = \sum_{k=0}^{n} k \pi_k = \frac{1 - \rho}{1 - \rho^{n+1}} \sum_{k=0}^{n} k \rho^k = \frac{\rho}{1 - \rho} \frac{n + 1}{1 - \rho^{n+1}} \rho^{n+1} \]

- **Loss probability**
  \[ r_n = 1, \quad r_k = 0, \quad k=0,1,..,n-1 \]
  \[ \Rightarrow \pi_n = \frac{1 - \rho}{1 - \rho^{n+1}} \rho^n \]

- **Throughput**
  \[ r_k = \mu, \quad k=1,2,..,n; \quad r_0 = 0 \quad (\text{or, } r_k = \lambda, \quad k=0,1,2,..,n-1; \quad r_n = 0) \]
  \[ \Rightarrow \mu (1 - \pi_0) = \lambda (1 - \pi_n) \]

- **Mean response time** \[ E[R] = E[N] / (\lambda (1 - \pi_n)) \]
\( M/M/1/n \): Response time distribution

- Response time distribution: Job may be rejected (or accepted)
  - Unconditional
  - Conditional (conditioned on the job being accepted):
- Reward assignment: for the \( k^{th} \) state, response time experienced by the tagged task is sum of \( k+1 \) service times, each of which is \( \text{EXP}(\mu) \), i.e., \( (k+1) \)-stage Erlang.
$M/M/1/n$: Response time distribution

– Unconditional (defective) distribution:

$$r_k = 1 - \sum_{i=0}^{k} \frac{\lambda \mu^i e^{-\lambda \mu}}{i!}; \quad r_n = 0 \Rightarrow F(t) = \sum_{k=0}^{n-1} \pi_k \left[ 1 - \sum_{i=0}^{k} \frac{\mu t^i e^{-\mu t}}{i!} \right]$$

– There is a defect (or mass at infinity) equal to $\pi_n$, the probability that the job is rejected.

- Conditioned on job being accepted, $R$ has a non-defective distribution:

$$r_k = \frac{1 - \sum_{i=0}^{k} \frac{\mu t^i e^{-\lambda \mu}}{i!}}{1 - \pi_n}; \quad r_n = 0 \Rightarrow F(t) = \sum_{k=0}^{n-1} \frac{\pi_k}{1 - \pi_n} \left[ 1 - \sum_{i=0}^{k} \frac{\mu t^i e^{-\mu t}}{i!} \right]$$
Measures for $M/M/1/n$ system

<table>
<thead>
<tr>
<th>Measure</th>
<th>Reward rate assignment</th>
<th>Expected steady-state reward rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean number in system</td>
<td>$r_j = j$</td>
<td>$\frac{\rho}{1-\rho} - \frac{n+1}{1-\rho^{n+1}} \rho^{n+1}$</td>
</tr>
<tr>
<td>Loss probability</td>
<td>$r_n = 1, r_j = 0 \ (j \neq n)$</td>
<td>$\pi_n = \frac{1-\rho}{1-\rho^{n+1}} \rho^n$</td>
</tr>
<tr>
<td>Throughput</td>
<td>$r_j = \mu \ (j \neq 0), r_0 = 0$ or $r_j = \lambda \ (j \neq n), r_n = 0$</td>
<td>$\mu(1 - \pi_0) = \lambda(1 - \pi_n)$</td>
</tr>
<tr>
<td>URTD$^a$</td>
<td>$r_j = 1 - \sum_{i=0}^{j} \frac{(\mu t)^i e^{-\mu t}}{i!}; r_n = 0$</td>
<td>Equation (8.55)</td>
</tr>
<tr>
<td>CRTD$^b$</td>
<td>$r_j = \frac{1-\sum_{i=0}^{j} \frac{(\mu t)^i e^{-\mu t}}{i!}}{1-\pi_n}; r_n = 0$</td>
<td>Equation (8.56)</td>
</tr>
</tbody>
</table>

$^a$Unconditional response time distribution.

$^b$Conditional response time distribution.
Model made in SHARPE GUI
Analysis frame showing output asked

Steady State prob. in all state asked

Throughput, Mean no. in system asked
Reward Rate Assignment

Copyright © 2006 by K.S. Trivedi
Output Generated by SHARPE

- Steady State prob. for all states
- Mean no. in the system
- Throughput
Cyclic Queuing model - Example

- Consider the cyclic queuing model shown on the next slide.

- Lengths of successive CPU execution bursts are independent exponentially distributed rv with mean $1/\mu$ and that the time to carry out an I/O operation is exponentially distributed with mean $1/\lambda$.

- At the end of a CPU burst, a program requests an I/O operation with probability $q_1$ or it completes execution with probability $q_0$; $q_0 + q_1 = 1$. 
Cyclic Queuing model – Example (contd.)

- At the end of a program completion, another statistically identical program enters the system, leaving the number of programs in the system at a constant level $n$ (known as the degree or level of multiprogramming).
Cyclic Queuing model – Example (contd.)

- Let the number of programs in the CPU queue including any being served at the CPU denote the state of the system, \( i \), where \( 0 \leq i \leq n \). Then state diagram becomes

- Steady State Probability is given by

\[
\pi_i = \left( \frac{\lambda}{\mu q_1} \right)^i \pi_0 = \rho^i \pi_0, \quad \text{and} \quad \pi_0 = \frac{1}{\sum_{i=0}^{n} \rho^i},
\]

where \( \rho = \frac{\lambda}{\mu q_1} \).
Cyclic Queuing model – Example (contd.)

• System throughput is given by

\[ E[T] = \mu q_0 U_0. \]

• \( E[T] \) is proportional to the CPU utilization, \( U_0 \) for fixed values of \( \mu \) and \( q_0 \).

• Let the random variable \( B_0 \) denote the total CPU time requirement of a tagged program. Then \( B_0 \) is \( \text{EXP}(\mu q_0) \).

• the average number of visits \( V_0 \) to the CPU is \( V_0 = 1/q_0 \), and thus \( E[B_0] = V_0 E[S_0] = 1/(\mu q_0) \), where \( E[S_0] = 1/\mu \) is the average CPU time per burst.

• If \( B_1 \) represents the total I/O service time per program, then

\[ E[B_1] = \frac{q_1}{q_0} \frac{1}{\lambda} = V_1 E[S_1] \]
Cyclic Queuing model – Example (contd.)

• Defining parameter \( \rho \) as

\[
\rho = \frac{\lambda}{\mu q_1} = \frac{q_0 \lambda}{q_1} \cdot \frac{1}{\mu q_0} = \frac{E[B_0]}{E[B_1]}. 
\]

• Thus \( \rho \) indicates the relative measure of the CPU versus I/O requirements of a program. If the CPU requirement \( E[B_0] \) is less than the I/O requirement \( E[B_1] \) (i.e., \( \rho < 1 \)), the program is said to be I/O-bound; if \( \rho > 1 \), then program is said to be CPU-bound; and otherwise it is called balanced.

• CPU utilization as a function of the degree of multiprogramming

\( n \)
Comments on Cyclic Queue

- The monotonic increase in thruput comes at the increased main memory size
- For a fixed amount of memory, degree of multiprogramming can be increased in a paging system
- In a paging system, CPU burst length will decrease with $n$, probability $q_1$ will also increase with $n$
- Thruput as a function of $n$ may no longer be monotonic; so called **thrashing** phenomena may occur (see Problem 7 on p. 436)
More Comments on Cyclic Queue

• If the I/O device uses other than FCFS scheduling, service rate $\lambda$ may be queue size dependent

• SLTF (shortest latency time first) scheduling is one such example

• See Problem 6 on p. 436
Model made in SHARPE GUI
Analysis Frame

SS prob. and Utilization asked
Reward Rate assigned
Output generated by SHARPE

SS prob. and Utilization calculated

Copyright © 2006 by K.S. Trivedi
Response Time in a Client-Server System

- Consider a client–server system with $M$ clients in which individual think times are exponentially distributed with mean $1/\lambda$ seconds and the service rate is $\mu$
- The queuing model is shown on the right
Response Time in a Client-Server System

- State Diagram can be drawn as

- Service time per request, $B_0$, is exponentially distributed with mean $E[B_0] = 1/\mu$ seconds.
- The steady-state probability that there are $n$ requests executing or waiting on the CPU is given by

$$\pi_n = \pi_0 \rho^n \frac{M!}{(M-n)!}, \quad n = 0, 1, 2, \ldots, M,$$

Copyright © 2006 by K.S. Trivedi
Response Time in a Client-Server System (contd.)

- Probability that the server is idle is
  \[
  \pi_0 = \frac{1}{\sum_{n=0}^{M} \rho^n \frac{M!}{(M - n)!}}
  \]

- The server utilization \( U_0 \) is \( 1 - \pi_0 \), and the average rate of request completion is \( E[T] = \mu(1 - \pi_0) = U_0/E[B_0] \).

- If \( E[R] \) is average response time, then on the average a request is generated by a given client in \( E[R] + (1/\lambda) \) seconds.

- Thus the average request generation rate of the client subsystem is \( M/[E[R] + (1/\lambda)] \).
Response Time in a Client-Server System (contd.)

- In the steady state, the request generation and completion rates must be equal.

\[ \frac{M}{E[R] + \frac{1}{\lambda}} = \mu(1 - \pi_0) \]

\[ E[R] = \frac{M}{\mu(1 - \pi_0)} - \frac{1}{\lambda} = \frac{M \cdot E[B_0]}{U_0} - \frac{1}{\lambda} = \frac{M}{E[T]} - \frac{1}{\lambda} \]

- \( E[R] \) is plotted as a function of the number of clients, \( M \), assuming \( 1/\lambda = 15 \text{ s} \) and \( 1/\mu = 1 \text{ s} \).
Response Time in a Client-Server System (contd.)

• When the number of clients \( M = 1 \), there is no queuing and the response time \( E[R] \) equals the average service time \( E[B_0] \).
• As the number of clients increases, there is increased congestion as the server utilization \( U_0 \) approaches 1. In the limit \( M \to \infty, E[R] \) is a linear function \( [ME[B_0] - (1/\lambda)] \) of \( M \).
• In this limit, the installation of an additional client increases every other client’s response time by the new client’s service time \( E[B_0] \).
• The number of clients, \( M^* \), for which the heavy-load asymptote \( E[R] = ME[B_0] - (1/\lambda) \) intersects with the light-load asymptote \( E[R] = E[B_0] \) is therefore called the saturation number and is given by

\[
M^* = \frac{E[B_0] + 1/\lambda}{E[B_0]} = 1 + \frac{\mu}{\lambda}
\]
Handoffs in wireless cellular networks

- **Handoff**: When an MS moves across a cell boundary, the channel in the old BS is released and an idle channel is required in the new BS

- **Hard handoff**: the old radio link is broken before the new radio link is established (AMPS, GSM, DECT, D-AMPS, and PHS)
Wireless Cellular System
Traffic in a cell

- New Calls
- Handoff Calls From neighboring cells
- Call completion
- Handoff out To neighboring cells

A Cell

Common Channel Pool

Copyright © 2006 by K.S. Trivedi
Performance Measures: Loss formulas or probabilities

• When a new call (NC) is attempted in an cell covered by a base station (BS), the NC is connected if an idle channel is available in the cell. Otherwise, the call is blocked.

• If an idle channel exists in the target cell, the handoff call (HC) continues nearly transparently to the user. Otherwise, the HC is dropped.

• Loss Formulas
  – New call blocking probability, $P_b$ : Percentage of new calls rejected
  – Handoff call dropping probability, $P_d$ : Percentage of calls forcefully terminated while crossing cells
Guard Channel Scheme

Handoff dropping less desirable than new call blocking!

Handoff call has Higher Priority: Guard Channel Scheme

GCS: g channels are reserved for handoff calls.

↓ g trade-off between P_b & P_d ↑
Assumptions

- Poisson arrival stream of new calls $\lambda_1$
- Poisson stream of handoff arrivals $\lambda_2$
- Limited number of channels: $n$
- Exponentially distributed completion time of ongoing calls $\mu_1$
- Exponentially distributed cell departure time of ongoing calls $\mu_2$
Markov chain model of wireless hard handoff

\[ \Lambda(j) = \begin{cases} 
\lambda_1 + \lambda_2, & \text{if } i < n - g \\
\lambda_2, & \text{if } n - g \leq i < n
\end{cases} \]

\[ \lambda = \lambda_1 + \lambda_2, \quad \mu = \mu_1 + \mu_2, \quad A = \frac{\lambda}{\mu}, \quad A_1 = \frac{\lambda_2}{(\mu_1 + \mu_2)}. \]

\[ M(i) = i(\mu_1 + \mu_2), \quad i = 1, 2, \ldots, n \]

\[ \Pi_i = \lim_{t \to \infty} \text{Prob}(C(t) = i), \quad n \in \Omega = \{0, 1, 2, \ldots, n\} \]

\[ C(t): \text{the number of busy channels at time } t \]
Loss formulas for wireless network with hard handoff

Dropping probability for handoff:

\[ P_d(n, g) = \pi_n = \frac{A^{n-g}}{n!} A_1^g \frac{1}{A_1^{(n-g)}} \left( \sum_{i=0}^{n-g-1} \frac{A^i}{i!} + \sum_{i=n-g}^{n} \frac{A^{n-g}}{i!} A_1^{i-(n-g)} \right) \]

Blocking probability of new calls:

\[ P_b(n, g) = \sum_{i=n-g}^{n} \pi_i = \frac{A^{n-g}}{i!} A_1^{i-(n-g)} \left( \sum_{i=0}^{n-g-1} \frac{A^i}{i!} + \sum_{i=n-g}^{n} \frac{A^{n-g}}{i!} A_1^{i-(n-g)} \right) \]

**Notation**: if we set \( g=0 \), the above expressions reduces to the classical Erlang-B loss formula.
Computational aspects

- Overflow and underflow might occur if $n$ is large
- Numerically stable methods of computation are required
  - Recursive computation of dropping probability for wireless networks
  - Recursive computation of the blocking probability
  - For loss formula calculator, see webpage:
    http://www.ee.duke.edu/~kst/wireless.html
Non-birth-death examples

• We consider two example performance models which give rise to non-birth-death CTMC models
  – M/M/2 heterogeneous queue
  And
  – A queue with non-Poisson arrival stream; in fact with MMPP (Markov modulated Poisson process) arrivals
$M/M/2$ Queue with Heterogeneous Servers ($\mu_1 > \mu_2$).

- Variant of $M/M/2$ with homogenous rate.
- The state of the system is defined to be the tuple $(n_1, n_2)$ where $n_1 \geq 0$ denotes the number of jobs in the queue including any at the faster server, and $n_2 \in \{0, 1\}$ denotes the number of jobs at the slower server.
- When both servers are idle, the faster server is scheduled for service before the slower one.
- The block diagram of the system is given.
$M/M/2$ Queue with Heterogeneous Servers (contd.)

- State Diagram is given by

[Diagram of state transitions with labels for $\lambda$, $\mu_1$, $\mu_2$, and state transitions]

- Solving balance eqns. we have

$$\pi(0, 0) = \left[1 + \frac{\lambda(\lambda + \mu_2)}{\mu_1\mu_2(1 + 2\rho)(1 - \rho)}\right]^{-1}.$$
**$M/M/2$ Queue with Heterogeneous Servers (contd.).**

- The average number of jobs in the system may now be computed by assigning the reward rate equal $r_{n_1,n_2} = n_1 + n_2$ as the number of customers in the system in state $(n_1, n_2)$.
- Therefore, the average number of jobs is given by

\[
E[N] = \sum_{k \geq 0} k\pi(k, 0) + \sum_{k \geq 0} (k + 1)\pi(k, 1) \\
= \pi(1, 0) + \pi(0, 1) + \sum_{k \geq 1} (k + 1)\pi(k, 1) \\
= 1 - \pi(0, 0) + \frac{\pi(1, 1)}{(1 - \rho)^2}
\]
Example- $M/M/2$ server

- Assume that the job stream modeled as a Poisson process with rate $\lambda$ jobs/minute, and the service times $\mu_2$ jobs/minute.
- Thus, $\rho_2 = \frac{\lambda}{\mu_2}$, and the average response time is given by $E[R_2] = \frac{1/\mu_2}{(1-\rho_2)}$.
- Suppose this response time is considered intolerable by the users and an extra server is added to the system. Let the service rate $\mu_1$ of the new server be equal to $\alpha \mu_2$ for some $\alpha > 1$.
- Now for both machines we have

$$\rho = \frac{\lambda}{\mu_1 + \mu_2} = \frac{\rho_2}{1 + \alpha}$$
Example- $M/M/2$ server (contd.)

- Average no. of customer in the system is given by
  
  \[
  E[N] = \frac{1}{F(1-\rho)^2} = \frac{\rho(1+\alpha)(1+\rho+\rho\alpha)}{(1-\rho)(\alpha+\rho+\rho^2+2\alpha\rho+\rho^2\alpha^2)}
  \]

  \[
  F = \frac{\alpha\mu_2^2(1+2\rho)}{\lambda(\lambda+\mu_2)} + \frac{1}{1-\rho}
  \]

  \[
  = \frac{\alpha(1+2\rho)}{\rho_2(1+\rho_2)} + \frac{1}{1-\rho}.
  \]

- Now only using the faster server, response time becomes
  
  \[
  E[R_1] = \frac{1/\mu_1}{1-\rho_1} = \frac{1/\mu_2}{\alpha-\rho_2}
  \]

- The condition under which $E[R_1] \leq E[R]$ can be simplified to
  
  \[
  \rho^2(1+\alpha^2) - \rho(1+2\alpha^2) + (\alpha^2 - \alpha - 2) \geq 0
  \]
Example- \( M/M/2 \) server (contd.)

- Thus, for example, if \( \lambda = 0.2 \) and \( \mu_2 = 0.25 \) so that \( \rho_2 = 0.8 \), then if the new server is more than 3 times faster than the old one, the inequality shown above is satisfied, \textit{and surprisingly}, it is better to disconnect the slower machine altogether.

- Average response times (in minutes) of the three configurations for different values of \( \alpha \) with \( \lambda = 0.2 \) and \( \mu_2 = 0.25 \).

<table>
<thead>
<tr>
<th></th>
<th>( \alpha = 1 )</th>
<th>( \alpha = 2 )</th>
<th>( \alpha = 3 )</th>
<th>( \alpha = 4 )</th>
<th>( \alpha = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E[R_1] )</td>
<td>20</td>
<td>3.33</td>
<td>1.818</td>
<td>1.25</td>
<td>0.95</td>
</tr>
<tr>
<td>( E[R_2] )</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>( E[R] )</td>
<td>4.7619</td>
<td>2.6616</td>
<td>1.875</td>
<td>1.459</td>
<td>1.20</td>
</tr>
</tbody>
</table>
M/M/2 Queue with Heterogeneous Servers (contd.)

- Also study perceived mean response time and perceived mean queue length idea (Problem 4 on p. 483)
- Study the optimization problem (Problem 2 on p. 483) and see the non-intuitive result
Homogeneous Poisson Process

- Consider a pure birth process \{N(t) | t \geq 0\} (i.e., all death rates are 0 or \(\mu_k = 0, k = 0,1,2,...\))
- Constant birth rate (i.e. \(\lambda_k = \lambda\) for \(k = 0,1,2,...\))
- \(N(t)\) is the number of arrivals (births) by time \(t\)

The state diagram can be shown as

```
0 \(\lambda\) 1 \(\lambda\) 2 \(\lambda\) ...... \(\lambda\) k \(\lambda\) k+1 \(\lambda\) ......```

Copyright © 2006 by K.S. Trivedi
Homogeneous Poisson Process (contd.)

\[
\frac{d\pi_0(t)}{dt} = -\lambda \pi_0(t), \quad k = 0,
\]

\[
\frac{d\pi_k(t)}{dt} = \lambda \pi_{k-1}(t) - \lambda \pi_k(t), \quad k \geq 1,
\]

• Solving the above (Kolmogorov) differential equations with the initial conditions that

\[
\pi_0(0) = 1, \quad \pi_k(0) = 0, \quad for \quad k \geq 1
\]

• We get (the Poisson pmf),

\[
\pi_k(t) = P(N(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, \quad k \geq 0; \quad t \geq 0
\]
Extensions to PP

- Non-Homogeneous Poisson Process (NHPP)
- Markov Modulate Poisson Process (MMPP)
- Markovian Arrival Process (MAP)
Markov Modulated Poisson Process

- An MMPP is a doubly stochastic Poisson process whose arrival rate is “modulated” by an irreducible continuous time Markov chain.
- Let $Q = [q_{ij}]_{m \times m}$ be the generator matrix of the CTMC with $m$ states. Each state $i$ is assigned a Poisson arrival rate, $\lambda_i, i=1, 2, \ldots, m$.
- The Poisson arrival rate is determined by the state of the CTMC; thus, when the Markov chain is in state $i$, arrivals occur according to a Poisson process of rate $\lambda_i$.
- Let $\lambda$ denote the arrival rate vector $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T$.
- Let $e = [1, 1, \ldots, 1]^T$. 

Copyright © 2006 by K.S. Trivedi
Markov Modulated Poisson Process (contd.)

• The reason for using this in traffic modeling is that MMPP has the capability of capturing some of the most important correlations between interarrival times and still remains analytically tractable.

• MMPP is a special case of the Markovian arrival process (MAP).
MMPP-the counting process

- Consider the counting process of MMPP.
- Let $N(t)$ be the number of arrivals in $(0, t]$ and $J(t)$ the state of the modulating CTMC.
- The bivariate process $\{J(t), N(t), t \geq 0\}$ is the counting process whose state space is $\{1, 2, \ldots, m\} \times \{0, 1, \ldots\}$.
- Transitions between states with the same number of arrivals—say, $(i, n)$ and $(j, n)$—are the same as transitions between state $i$ and $j$ of the CTMC with rate $q_{ij}$ and $q_{ji}$, respectively.
- An arrival may occur in any of the modulating CTMC states, resulting in the counter increasing by one.
- So a transition from state $(i, n)$ to state $(i, n+1)$ with rate $\lambda_i$ takes place.
- The state diagram of the bivariate process for a three-state MMPP is shown.

Copyright © 2006 by K.S. Trivedi
MMPP-the counting process (contd.)

• The counting process is also a homogeneous CTMC. Let $\pi$ be the steady-state vector of the modulating CTMC, which is the solution to
  
  \[ \pi Q = 0, \quad \pi e = 1 \]

• So we can see that the asymptotic average arrival rate:
  
  \[ \lim_{t \to \infty} \frac{E[N(t)]}{t} = \pi \lambda. \]

• The result shows that steady-state expected number of arrivals in an interval of length $t$ is the product of time duration $t$ and the average arrival rate, $\pi \lambda$. Where $\pi \lambda$ is the sum of rates weighted by steady-state probabilities of the modulating CTMC.
MMPP/M/1 Queue

• We now consider an MMPP/M/1 queue in which the arrival process is an MMPP characterized by the generator matrix $Q$ of the modulating CTMC and the arrival rate vector $\lambda$.

• The service time is exponentially distributed with mean $1/\mu$. The buffer size of the queue is assumed to be infinite.

• The state diagram for an MMPP/M/1 queue is shown next, in which the MMPP arrival process has three states.
MMPP/M/1 Queue (contd.)
Other arrival processes

• MAP, BMAP, PH type arrival processes have been studied [LUCA 1990, NEUT 1978, FISC 1993]

• Also periodic NHPP type arrivals have been studied [RIND 1995]