System Modeling Using Markov Chain and MATLAB

Prof. Ying-Dar Lin

Department of Computer Science
National Chiao-Tung University
Motivation

- Systems become complicated
  - Multithreaded multiprocessor architecture
  - Communication networks
    - Bandwidth allocation
    - Scheduling
    - Routing
  - Control systems
- Too complex for simple queuing analysis
  - Multi-dimensional models are required
Markov Chain

- A mathematical tool to model a system
- Time-dependent variables/dimensions
  - Form a *state*
- Conditional independence of future evolution on the past
  - Past history *summarized* in current state
- An object in a discrete set of locations
  - Modeled over time
  - Next position depends *only* on current one
- Can be discrete or continuous
Discrete Time Markov Chains (DTMC)

- Assume a stochastic process \( \{X_0, X_1, X_2, \ldots, X_n, \ldots\} \)
- Transition probability matrix \( P \)

\[
P = \{p_{ij}, i, j \in S\} \quad p_{ij} \geq 0, \sum_{j \in S} p_{ij} = 1
\]

Let \( \pi_i(n) = \Pr\{X_n = i\}, i \in S \) be the unconditional distribution at time \( n \), and

\[
\pi(n) = [\pi_0(n), \pi_1(n), \ldots, \pi_n(n), \ldots]
\]

be the corresponding row vector
For \( n=1 \)

\[
\pi_i(1) = \sum_{j \in S} \Pr\{X_0 = j, X_1 = i\} = \sum_{j \in S} \pi_i(0)p_{ji}
\]

and \( \pi(1) = \pi(0)P \)

Iterate the equation below

\[
\pi(n+1) = \pi(n)P, \quad n \geq 0
\]

we have \( \pi(n) = \pi(0)P^n \) and the limiting state prob. as

\[
\tilde{\pi} = \lim_{n \to \infty} \pi(n) = \lim_{n \to \infty} \pi(0)P^n = \pi(0)\lim_{n \to \infty} P^n = \pi(0)\tilde{P}
\]
DTMC – Simple Examples

System 1:

\[
\begin{bmatrix}
0 & 1 \\
0 & 1 \\
\end{bmatrix}
\]

State transitions

Initial state

Transition probability matrix

\[
\begin{bmatrix}
0 \\
1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]

\[
[0 \ 1] \times \begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix} = [0 \ 1]
\]

(The state does not involve)

System 2:

\[
\begin{bmatrix}
0.5 & 0.5 \\
0.5 & 0.5 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 \\
0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
.5 & .5 \\
.5 & .5 \\
\end{bmatrix}
\]

\[
[0 \ 1] \times \begin{bmatrix}
.5 & .5 \\
.5 & .5 \\
\end{bmatrix} = [.5 \ .5]
\]

(After the 1st round)
Population Movement

Transition probability matrix

\[
\begin{bmatrix}
0.94 & 0.04 & 0.02 \\
0.02 & 0.97 & 0.01 \\
0.02 & 0.03 & 0.95
\end{bmatrix}
\]

Initial state \( S : [0.55, 0.20, 0.25] \)

\[
S(10) = S \times P^{10} = [0.38, 0.3733, 0.2464] \quad \text{(After 10 years)}
\]

\[
S(200) = S \times P^{200} = [0.25, 0.5417, 0.2083] \quad \text{(After 200 years)}
\]

\[
S(201) = S \times P^{201} = [0.25, 0.5417, 0.2083] \quad \text{(Steady state reached)}
\]
Continuous Time Markov Chains (CTMC)

- Similar to DTMC except
  - With infinitesimally small time units
  - Transitions are activated with *rates*, not *probabilities*

- Closely model a system
  - Over time
  - Job arrival and service times are considered

- Detailed definition is not mentioned here
  - We focus on the directions of using it
Properties of CTMC

- Two properties hold for the transition rate matrix
  - Conservative:
    \[ q_{ii} = -q_i = -\sum_{j \neq i} q_{ij} \], where
    \[ Q = \begin{pmatrix}
    -q_0 & q_{01} & q_{02} & q_{03} \\
    q_{10} & -q_1 & q_{12} & \vdots \\
    q_{20} & \vdots & \ddots & \vdots \\
    \vdots & \vdots & \ddots & \vdots 
    \end{pmatrix} \]
    \[ q_i \] denotes outgoing rate from state i
  - Stable: all \( q_{ij} \) are finite
Steady State Derivation

- Iterative multiplication: \( \pi_{k+1} = \pi_k \times Q \)
  - Does not converge ( \( \therefore |rate| > 1 \) )

- Steady-state derivation (assume NxN matrix)

\[
\begin{align*}
\pi \times Q &= 0 \quad \text{(1)} \\
\pi \times e &= 1 \quad \text{(2)}
\end{align*}
\]

\( \pi \): stationary prob. vector
\( e \): column matrix with all elements being 1

After some matrix manipulations with (1) and (2), we have

\[ \pi \times T = (0,0,0...,1) \], where \((0,0,0,...,1)\) is the initial state \( A \) and

\[
T_{n,m} = \begin{cases} 
Q_{n,m}, & m = 0,...,N-2 \\
1; & m = N - 1 
\end{cases}
\]

Therefore, we can have

\[ \pi = \pi \times T \times T^{-1} = (0,0,0...,1) \times T^{-1} \]
A Sample CTMC Model

A core processor with a number of coprocessors

<table>
<thead>
<tr>
<th>Task name</th>
<th>Processing time ($\mu$sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Receive</td>
<td>27.3</td>
</tr>
<tr>
<td>(2) Core_A</td>
<td>31</td>
</tr>
<tr>
<td>(3) Crypto</td>
<td>12.6</td>
</tr>
<tr>
<td>(4) Core_B</td>
<td>49</td>
</tr>
<tr>
<td>(5) Transmit</td>
<td>27.3</td>
</tr>
</tbody>
</table>
Model Description

**Rec**: A<bufA => rate=

**Core_A**: S=0, C<bufC => rate= 

**Cop**: B<bufB => rate= 

**Core_B**: S=1, T<bufT => rate= 

**Trans**: rate= 

\( \lambda \): Pkt arrival rate

\( \lambda_{s1}, \lambda_{s2} \): Switching rate

\( T_{s1}, T_{s2} \): Mean run length

\( \mu_X \): Service time of \( X \)

buf\( _X \): Buffer size of \( X \)

**State definition:**

\[(R, A, C, B, T, S)\]

**Conditions of the state transitions**

**PCS:**

\( \lambda_{s1} = \frac{1}{T_{s1}} \)

\( \lambda_{s2} = \frac{1}{T_{s2}} \)

Arrival: \( R<buf_R \) => rate= \( \lambda \)

Rec: \( A<buf_A \) => rate= \( \mu_R \)

Core_A: \( S=0, C<buf_C \) => rate= \( \mu_A \)

Cop: \( B<buf_B \) => rate= \( \mu_C \)

Core_B: \( S=1, T<buf_T \) => rate= \( \mu_B \)

Trans: rate= \( \mu_T \)

**Transition Matrix Generation**

**Conditions of the state transitions**

**PCS:**

\( \lambda_{s1} = \frac{1}{T_{s1}} \)

\( \lambda_{s2} = \frac{1}{T_{s2}} \)

rate= 0 (if \( A \geq 0, B=0 \))

rate= 0 (if \( B \geq 0, A=0 \))
Example Transition

Transition diagram => Q => T => T⁻¹ => π

Results to have:

• Utilization of components
  • Ex: Core_util = \sum \Pr\{(A \neq 0) \text{ or } (B \neq 0)\}

• Queue length:
  • Ex: Cop_len = \sum \Pr\{C > 0\} \times C

• Throughput = \sum \Pr\{T \neq 0\} \times \mu_T

State definition:
( R, A, C, B, T, S )
buf=3 for each component
Short Intro. to MATLAB

- Matrices manipulations
  - Calculate steady-state probabilities

- Commands used
  - "load": load matrices
  - "/": Ex: $\pi = AT^{-1} = A/T$ (A: initial state)
  - "dlmwrite": write results to a file
    - Ex: dlmwrite('ssresult',ans)
      (write results to the "ssresult" file)
User Interface

Matrix sizes do not exceed physical memory capacity

\[ \pi = A \times T^{-1} \]  (A: initial state)

Result file resides here!

Append ‘\n’ after each steady-state probability
Results Collection

- Steady-state prob.
  - Recorded in “ssresult”
- Core utilization
  - Prob. that Core_A and Core_B are not 0
- Utilizations of other components?
Requirements of the Mini-Project

- Install Matlab7
- Feed in the input matrices
  - A: initial state matrix
  - T: transition rate matrix
  - Both of them will be provided in the course page
- Statistics collector will also be provided
- Questions
  - Utilization of each component
    - Core (Core_A and Core_B), Rec, Cop, and Trans
  - System throughput?
  - Which one is the bottleneck?
  - What is the queue length for each component?
Requirement of the Term-Project (1/2)

- Write a program to generate the transition rate matrix
  - According to the sample CTMC model and “Steady-State Derivation”
  - Any language is fine (Perl is recommended)

- Adopt different input loads
  - By adjusting the arrival rate ($\lambda$) (packet size = 512 bytes)
  - Step of 5% (5%, 10%, 15%, ..., 40%)
  - Other parameters
    - PCS mean run length configured as $200\mu$sec
    - Buffer size ($buf_x$) = 2
    - Mean task processing time: refer to the p.11

\[
\begin{aligned}
T_{s_1} = T_{s_2} &= 200, \\
i.e. \quad \lambda_{s_1} = \lambda_{s_2} &= \frac{1}{200}
\end{aligned}
\]
Requirement of the Term-Project (2/2)

- Based on the mini-project
  - Feed A and T into the MATLAB
  - Obtaining statistics of the core
    - Use the statistics collector written previously

- Task: comparison against the implementation result (p.15)
  - How close is your model to the implementation?
  - Why is the difference?
  - What else do you observe?
References

- MATLAB forum, located at http://www.mathworks.com/matlabcentral/. (the software shall be available in the Departmental Computer Center)
- Help pages in the MATLAB. Examples are provided for commands.
- Any textbooks for the Markov chain.
  - J. Daigle, Queueing Theory for Telecommunications, Addison Wesley, 1992