Availability of this Tangram II Tutorial

- For more info and source code of examples (useful to learn the syntax): www-net.cs.umass.edu/~sadoc/perf08

- This slides should be read after those that present the capabilities, structure and interface of Tangram II (available in the link above)

- Availability of Tangram II
  - Edlab: just run ./tangram2 in the command line
  - For download: www.land.ufrj.br
  - Self contained and bootable CD: on demand
Tangram II Tutorial: Overview

• How to create a model?
  – The key ingredient: the state variables
  – Events
  – Messages

• What to do with the model?
  – Obtain performance metrics
    * steady state results and/or transient results
    * e.g., expectations (mean number of customers in the system), distributions (distribution of the waiting time), ...
Tangram II Tutorial: Overview

- How to solve the model?
  - analytically
  - simulation

- Examples
  - M/M/1/K; M/M/1/K with failures
  - M/M/2/K; M/M/2/K with failures
  - ON-OFF/M/1/K
  - M/PH/1/K; PH/M/1/K; PH/PH/1/K
How to create a model?

- Identify:
  1. state variables
  2. objects
  3. events and messages
  4. actions to be taken when events trigger and when messages are received.
How to create a model? First Step

• First step: what are state variables?
  – what information do we need to store in order to take a snapshot of the system?
  – what is range of state variables?
  * Tangram II only works with finite state Markov Chains: all the state variables must have a finite pre-defined range
Identifying the state variables: Examples

- model: (state variables), description [range]
- M/M/1/K: \((X)\), number of customers in queue [\(range : (\{0, \ldots, K\})\)]
- M/M/1/K with server failures: \((X, S)\), number of customers in queue and state of server [\(range : (\{0, \ldots, K\}; \{ON, OFF\})\)]
Identifying the state variables: Examples

- **M/M/2/K**: \((X, S_1, S_2)\), number of customers in queue, number of customers in server 1, number of customers in server 2, \([\text{range} : (\{0, \ldots, K\}; \{0, 1\}; \{0, 1\}])\]

- **ON-OFF/M/1/K**: \((X, S)\), number of customers in queue and state of source; \([\text{range} : (\{0, \ldots, K\}; \{ON, OFF\}])\)
M/M/2/K with failures

- M/M/2/K with server failures:
  - option 1: \((X, B_1, B_2, SS_1, SS_2)\),
    - \(X\), number of customers in the queue, [range \(\{0, \ldots, K\}\)]
    - \(B_1\), is server 1 busy?, [range \(\{0, 1\}\)]
    - \(B_2\), is server 2 busy?, [range \(\{0, 1\}\)]
    - \(SS_1\), state of server 1, [range \(\{ON, OFF\}\)]
    - \(SS_2\), state of server 2, [range \(\{ON, OFF\}\)]
M/M/2/K with failures

- It is a good idea to always check number of states in model for debugging purposes: use combinatorial argument

- State variables: \((X, B_1, B_2, SS_1, SS_2)\)

- Total number of possibilities: \((Q + 1) \times 2 \times 2 \times 2 \times 2 = 176\) (where \(Q = 10\) is the buffer size)

- Number of reachable states: 44
Couting the number of states

• Unreachable states:
  
  – more than 1 user in queue, no one in service, one server ON
    * \((X > 0, B_1 = 0, B_2 = 0, SS_1 = ON, SS_2 = OFF)\): 10
    * \((X > 0, B_1 = 0, B_2 = 0, SS_1 = OFF, SS_2 = ON)\): 10
    * \((X > 0, B_1 = 0, B_2 = 0, SS_1 = ON, SS_2 = ON)\): 10
  
  – 0 users in queue: only four possible states
    * \((X = 0, B_1 = x, B_2 = x, SS_1 = x, SS_2 = x)\): 12
  
  – incompatible \(B\) and \(SS\)
    * \((X > 0, B_1 = 1, B_2 = x, SS_1 = OFF, SS_2 = ON)\): 20
    * \((X > 0, B_1 = x, B_2 = 1, SS_1 = ON, SS_2 = OFF)\): 20
\( X > 0, B_1 = 1, B_2 = 0, SS_1 = OFF, SS_2 = OFF \): 10
\( X > 0, B_1 = 0, B_2 = 1, SS_1 = OFF, SS_2 = OFF \): 10
\( X > 0, B_1 = 1, B_2 = 1, SS_1 = OFF, SS_2 = OFF \): 10

- priorities
  \( X = 1, B_1 = 0, B_2 = 1, SS_1 = ON, SS_2 = ON \): 1

- special case \( X = 1 \)
  \( X = 1, B_1 = 1, B_2 = 1, SS_1 = ON, SS_2 = ON \): 1

- \( X \geq 2 \)
  \( X \geq 2, B_1 = 1, B_2 = 0, SS_1 = ON, SS_2 = ON \): 9
  \( X \geq 2, B_1 = 0, B_2 = 1, SS_1 = ON, SS_2 = ON \): 9

- Total number of states: 176 - 100 - 14 - 9 - 9 = 44
M/M/2/K with failures

- is it possible to model system using fewer state variables?
  - Yes! Tradeoff between clarity-flexibility versus space-performance
  - Clarity: readability of the code and facility to describe the measures of interest
  - Flexibility: regarding the ordering the states (Matrix Visualization tool): the higher the number of state variables, the higher the number of permutations of those
  - Space: more state variables require more space
  - Performance: if the model with less variables takes advantage of symmetries in the problem, it may be more efficient to solve
M/M/2/K with failures: removing redundancy

- option 2: \((X, SS_1, SS_2)\),
  - \(X\), number of customers in queue,
  - \(SS_1\), state of server 1,
  - \(SS_2\), state of server 2
- rationale: \(B_1\) and \(B_2\) are redundant. Given state of servers and size of queue we can infer number of busy servers
  - \((SS_1 = ON) \land (X > 0) \iff B_1 = 1\)
  - \((SS_2 = ON) \land (X > 1 \lor (X = 1 \land SS_1 = OFF)) \iff B_2 = 1\)
- Key assumption: work conserving system.
- Number of states in this model: equal to former. Why?
M/M/2/K with failures: compact description

- option 3: \((X, B, S)\),
  - \(X\), number of customers in the queue,
  - \(B\), number of busy servers,
  - \(S\), number of servers in state ON

- option 4: \((X, S)\),
  - \(X\), number of customers in the queue,
  - \(S\), number of servers in state ON
How to create a model? Second Step

• Second step: how do transitions between states occur?
  – In Tangram-II, user specifies CTMCs ($\pi Q = 0$)
  – Tangram-II internally works with discrete time models ($\pi = \pi P$)
    * continuous and discrete time models are equivalent: one can always convert one into the other, $P = Q/\Lambda + I$ (uniformization)
    * $\Lambda$ is uniformization rate (largest element in absolute value in diagonal of $Q$)
Identifying Rates

- \( \lambda \rightarrow \) \( \mu \rightarrow \) M/M/1/k
- \( \lambda \rightarrow \) \( \downarrow \) \( \rightarrow \) \( \downarrow \) \( \rightarrow \) M/M/1/k with failures
- \( \gamma_1 \rightarrow \) \( \gamma_2 \rightarrow \) ON-OFF/M/2/k
- \( \lambda \rightarrow \) \( \mu \rightarrow \) M/M/2/k
Identifying Rates: M/M/2/k with failures
How to create a model? Third Step

- Third step: thinking in terms of objects, events and messages
  - Objects: encapsulate state variables. A state variable can only be accessed inside the object where it is defined.
  - Objects react to reception of messages and can be associated with events
    - Events: scheduled according to rate specified by user.
    - Messages: used to establish communication between objects.
Identifying Objects

M/M/1/k

M/M/1/k with failures

ON-OFF/M/2/k

M/M/2/k
Identifying Objects: M/M/2/K with failures
Identifying Events

M/M/1/k

M/M/1/k with failures

ON-OFF/M/2/k

M/M/2/k
Identifying Events: M/M/2/K with failures

- 1. arrival
- 2. state change
- ON-OFF facility 1
- 3. state change
- OFF-ON facility 1
- 4. service facility 2
- 5. state change
- ON-OFF facility 2
- 6. state change
- OFF-ON facility 2

M/M/2/k with failures
Identifying Messages

• in all models considered up to now:
  – source, when its “arrival” event triggers, generates message to server;
  – server adds an element to its queue when it receives new message (if there is still space on the buffer)
How to create a model? Fourth Step

- Fourth step: identify actions that need to be taken when
  - event triggers
  - message is received
- Look at examples in http://www-net.cs.umass.edu/~sadoc/perf08
What to do with model?

• Obtain performance metrics

• Performance metrics must be written in terms of state variables

• Examples: M/M/1/K
  – average queue size? $E[X]$
  – utilization = probability queue not empty: $P(X \neq 0)$
  – distribution of queue size $X$
    * for every value of $x$ ($x \in \{0, \ldots, K\}$), what is $P(X = x)$?
Performance Metrics

- M/M/1/k with server failures:
  * probability that server is on and queue is empty: $P(X = 0, S = 1)$
  * rate at which packets are lost: $P(X = K)\lambda$

- M/M/2/k:
  * probability that exactly one server is on and queue is empty: $P(S_1 + S_2 = 1, X = 0)$
  * probability that at least one server is on: $P(S_1 + S_2 \geq 1)$
  * expected number of servers on: $E[S_1 + S_2]$
Performance Metrics (cont)

- ON-OFF/M/1/k:
  * probability that source is on and queue is full: \( P(S = 1, X = K) \)
  * rate at which packets are lost: \( P(S = 1, X = K) \lambda \)

- M/M/2/k with failures:
  * probability that queue is empty given that at least one server is on: \( P(X = 0 | SS1 + SS2 \geq 1) \)
  * utilization of service facility 1: \( P(B1 > 0) \)
Transient Performance Metrics

- Up to now we considered only steady state metrics ($t \to \infty$).
- Tangram-II also allows for transient analysis, i.e. allows the user to answer questions about the system when $t < \infty$.
- Examples:
  - M/M/1/k
    - if queue initially full, what is average number of customers after 10 units of time?
    - what if queue initially empty?
Transient Analysis: Uniformization

- Tangram-II uses uniformization for transient analysis
- $\pi(t)$ probability distribution of the states of the system at time $t$
- $P$ state transition matrix
  - probability of transition from state $i$ to state $j$ in one step
- $\Lambda$ is uniformization rate (largest element of generator matrix obtained from continuous model description)
- Uniformization technique: $\pi(t) = \sum_{n=0}^{\infty} P_n e^{-\Lambda t} (\Lambda t)^n / n!$
Transient Analysis with Absorbing States

- Consider following problem:
  - in M/M/1 system, given a full buffer, what is mean time for buffer to empty for first time?

- In principle, Tangram-II does not have automatic support to answer such a question.

- However, user can always change the model in such a way that after the queue empties \((X = 0)\) the system returns back to the initial state \((X = K)\)
• Let $\pi = (\pi_0, \pi_1, \ldots, \pi_K)$ be steady state solution of modified system.

• If we pick $\pi' = (\pi_0/\pi_0, \pi_1/\pi_0, \ldots, \pi_K/\pi_0)$ then sum of elements of $\pi'$ characterizes measure of interest, since for every visit to state 0 we have $\sum_{i=1}^{K} \pi_i/\pi_0$ visits to other states.
How to solve the model?

• Once a model is constructed, it must be solved in order for the measures of interest to be obtained

• One of the powerful features of Tangram-II consists on the fact that once the model is specified the user is allowed to:
  – solve it analytically
  – simulate it

• When should each of those methods be considered?
- analytical solution:
  * all events are exponential (Markov Chain) or
  * at most one of them is not exponential (embedded Markov Chain)
- simulation:
  * debugging purposes;
  * events with arbitrary distributions;

• What does it mean to solve model analytically?

  - From continuous description of model, Tangram-II automatically generates
    * infinitesimal generator matrix $Q$
    * from $Q$, it generates matrix $P$ using uniformization
– To solve analytical model
  * in steady state: means to find distribution $\pi$ such that $\pi P = \pi$
  * in transient state at time $t$: means to find distribution $\pi(t)$ for states of system at time $t$

• What solution methods are available?
  – in steady state: direct methods (GTH = Gaussian Elimination) and indirect methods (Power Method, Gauss Seidel, Gauss Jacobi)
  – in transient state: uniformization (among others)
More examples: Phase Type Distribution

- Next we present some examples involving phase type distribution
- The process can be written in the form of a transition rate matrix:

\[
\begin{bmatrix}
S & S^0 \\
0 & 0
\end{bmatrix}
\]

- Continuous phase-type distribution is the distribution of time from the process starting until absorption
The Phase Type Distribution

• Why the phase type distribution is important?
  – because it can be used to approximate any distribution

• Special case that we are going to consider: Erlang, 2 identical phases in sequence

\[ \rightarrow \mu \rightarrow \mu \rightarrow \]
M/PH/1

source

events

1. arrival

server

events:

1. service 1

2. service 2

state variables: (S, X)

S: state of server facility

X: number of elements in the queue
**PH/M/1**

**at most one user**

\[ \mu \rightarrow \text{triangle} \rightarrow \mu \rightarrow \text{rectangle} \rightarrow \text{circle} \]

**symbology**

**standard symbology**

- **source**
- **events:**
  1. arrival 1
  2. arrival 2

- **server**
- **events**
  1. service

**state variables:** \((S, X)\)
- \(S\): state of source facility
- \(X\): number of elements in the queue
PH/PH/1

source

events:
1. arrival 1
2. arrival 2

server

events:
1. service 1
2. service 2

state variables: (S, X)
S1: state of source facility
S2: state of server facility
X: number of elements in the queue