CS 5224

Simulation

Dr. Chan Mun Choon
School of Computing, National University of Singapore

References

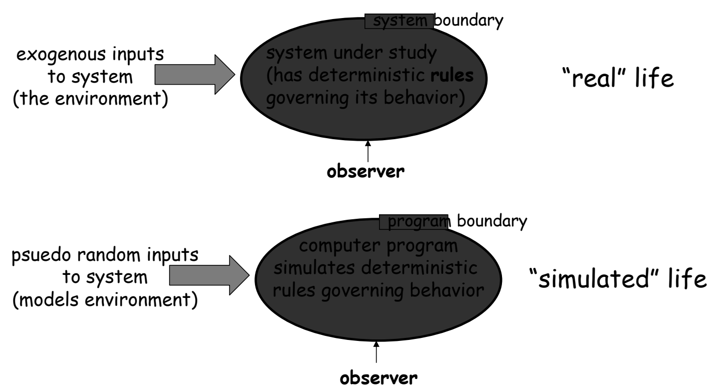
- Some slides are taken from the following source:
  - Jim Kurose's class in UMass (http://www-net.cs.umass.edu/cs653-2002/documents/network_simulation.ppt)
- References
  - Mahbub Hassan and Raj Jain, "High Performance TCP/IP Networking: Concepts, Issues and Solution," Chapter 4: TCP/IP Network Simulation.

Network Simulation

Overview:

- **fundamentals of discrete event simulation**
  - example of a simulation tools: ns-2
- analyzing simulation outputs
- random distribution generation

What is simulation?



Why Simulation?

- **Goal:** study system *performance, operation*
- Real-system not *available, is complex/costly or dangerous (*e.g.: nuclear explosion, weather forecast, space simulations, flight simulations)
- Quickly evaluate design *alternatives* (e.g.: different system configurations, parameters)
- Evaluate *complex functions* for which closed form formulas or numerical techniques not available
  - Simulation can incorporate more details than analytical modeling
- Validate analytical results
  - Provide more confidence to the analytical results

Simulation: advantages/drawbacks

- **Advantages:**
  - can discover/observe interesting behaviors not foreseen
  - can tune parameters (know which parameter values work well and when)
  - can control inputs: does system respond as expected
  - repeatable: can use for debugging, more detailed analysis after the fact
- Drawbacks/dangers:
  - simulated behavior may not equal real behavior (particularly if you have a bug in simulation)
  - can never 100% reproduce "real life" (e.g., does not include sufficient details or did not use correct inputs)
  - detailed simulations may not scale well (simulation not feasible)

Simulation: advantages/drawbacks

- advantages:
  - save lives, money
  - find bugs (in design) in advance
  - generality: over analytic/numerical techniques
  - detail: can simulate system details at arbitrary level
- drawbacks:
  - caution: does model reflect reality?
  - large scale systems: lots of resources to simulate (especially accurately simulate)
  - may be slow (computationally expensive – 1 min real time could be hours of simulated time)
  - art: determining right level of model complexity
  - statistical uncertainty in results

Steps/Check Lists

1. Define the objectives
2. Design the "environment"
   - E.g. network topology, link bandwidth, buffer size, traffic model, traffic load etc.
3. Select performance metrics (what to measure)
4. Select variable parameters
5. Construct model
6. Configure software to generate relevant performance data
7. Run simulation program and collect data
8. Present and **interpret** data

Types of Simulation

- Continuous vs. **Discrete Event (DES)**
- Terminating vs. Steady State
  - Study behavior of a system for a well-defined period of time or number of events
  - Terminate simulation only when the system has reached a "steady" state.
- Synthetic vs. Trace-Drive Simulation
  - Synthetically generate input traffic using random traffic generator
  - Use captured packet trace as input traffic

Programming a DES

- *simulated time:* internal (to simulation program) variable that keeps track of simulated time
- *system "state":* variables maintained by simulation program define system "state"
  - e.g., may track number (possibly order) of packets in queue, current value of retransmission timer
- *events:* points in time when system changes state
  - each event has associate *event time*
    - e.g., arrival of packet to queue, departure from queue
    - precisely at these points in time that simulation must take action (change state and may cause new future events)
  - model for time between events (probabilistic) caused by external environment

Discrete Event Simulation

- Future Event List (FEL)
  - Mechanism for advancing simulation time and guaranteeing that all events occur in **correct chronological order**
  - Contains all event notices for events that have been scheduled to occur at a future time
  - **Event Notice**: a record of an event to occur at current or some future time, along with any associated data necessary to execute the event
    - At a minimum, the record includes (event type, event time)

Example

Consider a M/M/2/2 system with λ = 1 packet/sec, μ = 0.5 sec
Events of interest: arrivals and departures

Initially queue is empty. Current event is
At t = 0.42s, arrival of P1
T=0.42, P1 arrives with service time 0.29. Departs at 0.71s (0.42 + 0.29)

Current Events are
At t = 0.71s, P1 departs
At t = 0.54s (0.42 + 0.12), P2 arrives
T=0.54s, P2 arrives with service time 0.41. Departs at 0.92s (0.54+0.41)

Current Events are
At t = 0.71s, P1 departs
At t = 0.92s, P2 departs
At t = 1.21s (0.54 + 0.67), P3 arrives

Example (cont'd)

T=0.71s, P1 departs
T=0.92s, P2 departs
T=1.21s, P3 arrives with service time 0.87. Departs at 2.09s (1.21+0.87)

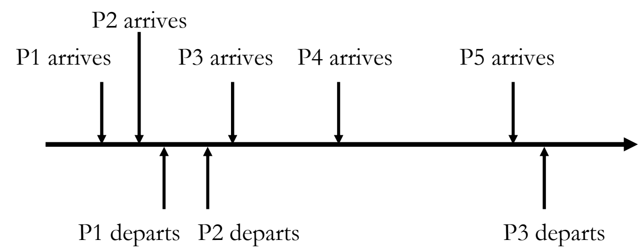
Current Events are
- At t = 2.09s, P3 departs
- At t = 1.54s, (1.21 + 0.33), P4 arrives

T=1.54s, P4 arrives with service time 0.48. Departs at 2.02s (1.54+0.48)

Current Events are
- At t = 2.09s, P3 departs
- At t = 2.02s, P4 departs
- At t = 1.94s (1.54+0.50), P5 arrives

At t = 1.94s, P5 arrives and is dropped since buffer is full !
The departure of P5 is not added to the event list

Example (cont'd)



Events are processed in chronological order in increasing sequence of the simulation time

NS-2

- What is ns-2?
  - **NS - Network Simulator**
  - Discrete event simulator targeted at networking research
    - supports simulation of TCP, routing and multicast protocols over wired and wireless (local and satellite) networks.
    - URL: http://www.isi.edu/nsnam/ns/index.html
- Brief History
  - 1989: REAL network simulator
  - 1995: DARPA VINT project
  - most recent release (ns-2.28 released Feb 3, 2005)

NS-2

- Why is it popular?
  - Its free and open source
  - Widely used, allow easy *comparison* of similar protocols
  - Has extensions and implementations of many versions of TCP, ad-hoc routing protocols and many others
- tutorial for ns-2
  - http://www.isi.edu/nsnam/ns/ns-tutorial/

Ns Models

- Traffic models and applications:
  - Web, FTP, telnet, constant-bit rate, real audio
- Transport protocols:
  - unicast: TCP (Reno, Vegas, etc.), UDP
  - Multicast: SRM
- Routing and queueing:
  - Wired routing, ad hoc rtg and directed diffusion
  - queueing protocols: RED, drop-tail, etc
- Physical media:
  - Wired (point-to-point, LANs), wireless (multiple propagation models), satellite

C++ and OTcl Separation

- "data" / control separation
  - C++ for "data":
    - per packet processing, core of *ns*, performs actual simulation
    - fast to run, detailed, complete control
  - OTcl for control:
    - Simulation scenario configurations
    - Periodic or triggered action
    - Manipulating existing C++ objects
    - fast to write and change

\+ running vs. writing speed

– Learning and debugging (two languages)

Hello World - Interactive Mode

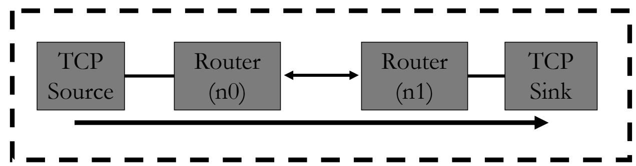
Interactive mode:

```
helium % ns
% set ns [new Simulator]
_o3
% $ns at 1 "puts \"Hello
  World!\""
1
% $ns at 1.5 "exit"
2
% $ns run
Hello World!
helium%
```

Batch mode:

```
simple.tcl
   set ns [new Simulator]
   $ns at 1 "puts \"Hello
     World!\""
   $ns at 1.5 "exit"
   $ns run
helium% ns simple.tcl
Hello World!
helium%
```


Simple Example




```
set ns [new Simulator]
    set n0 [$ns node]; set n1 [$ns node]
$ns duplex-link $n0 $n1 10M 10ms DropTail
set tcp [new Agent/TCP]; set tcpsink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp; $ns attach-agent $n1 $tcpsink
$ns connect $tcp $tcpsink
```

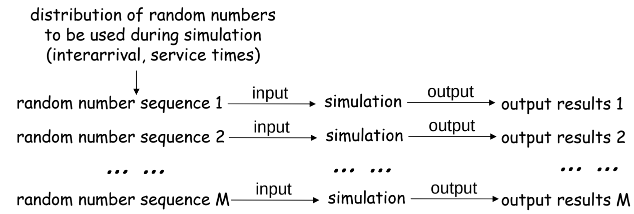
Network Simulation

Overview:

- fundamentals of discrete event simulation
- **analyzing simulation outputs**
- random distribution generation

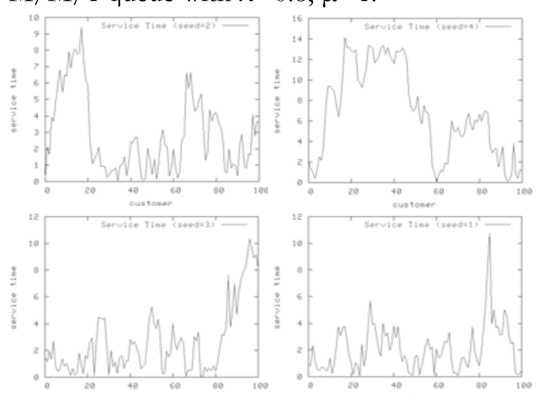
Analyzing Output Results

Each time we run a simulation, (using different random number streams/seeds), we will get different output results!



Analyzing Output Results

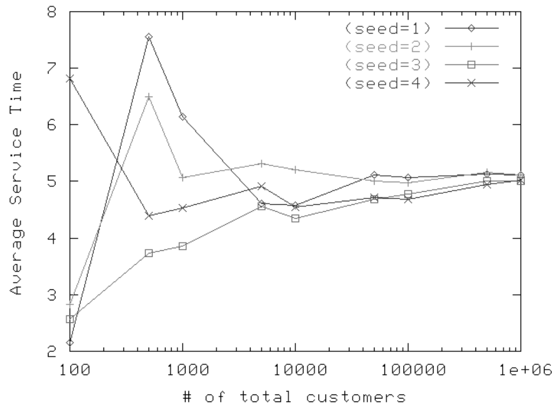
- M/M/1 queue with λ=0.8, μ=1.



Analyzing Output Results

- Besides the random sequence used, how long the experiment is run can also affects the result
  - Especially if the run is not sufficiently long
- Output results may converge to limiting “steady state” value if simulation run “long enough”

Analyzing Output Results



Confidence Intervals

- run simulation: get estimate V_1 as estimate of performance metrics of interest
- repeat simulation M times (each with new set of random numbers), get V_2, … V_M – *all different!*
- which of V_1, … V_M is "right"?
- intuitively, average of M samples should be "better" than choosing any one of M samples

$$\overline{V} = \frac{\sum_{j=1}^{M} V_j}{M}$$

How "confident" are we in $\overline{V}$?

Confidence Intervals

- Can not get perfect estimate of true mean, μ, with finite # samples
- Look for bounds: find c1 and c2 such that:

  Probability(c1 < μ < c2) = 1 – α

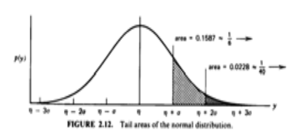
  [c1,c2]: *confidence interval*

  100(1-α): *confidence level*
- One approach for finding c1, c2 (suppose α=.1)
  - take k samples (e.g., k independent simulation runs)
  - sort
  - find largest value is smallest 5% -> c_1
  - find smallest value in largest 5% -> c_2

Central Limit Theorem

- Central Limit Theorem: If samples V_1, … V_M independent and from same population with population mean μ and standard deviation σ, then

sample mean: $\overline{V} = \frac{\sum_{j=1}^{M} V_j}{M}$



is approximately normally distributed with mean u and standard deviation $\frac{\sigma}{\sqrt{M}}$

Confidence Intervals .. more

- Still don't know population standard deviation. So we estimate it using sample (observed) standard deviation: $\sigma_v^2 = \frac{1}{M-1}\sum_{m=1}^{M}(V_m - \overline{V})^2$
- Given $\overline{V}, \sigma_v^2$ we can now find upper and lower tails of normal distributions containing α100% of the mass

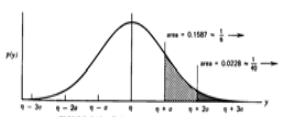


FIGURE 2.12. Tail areas of the normal distribution.

Confidence Intervals .. the recipe

- Given samples $V_1, \ldots V_M$, (e.g., having repeated simulation M times), compute

$$\overline{V} = \frac{\sum_{j=1}^{M} V_j}{M}$$

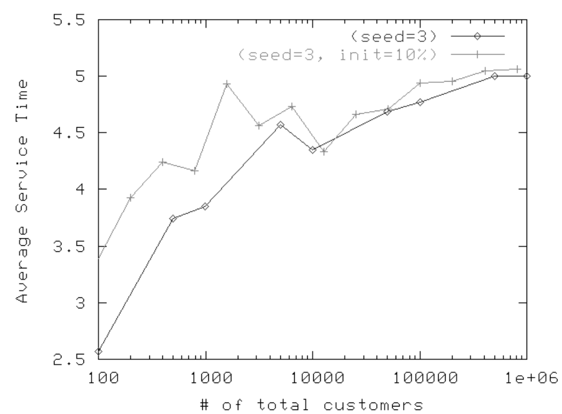
$$\sigma_v^2 = \frac{1}{M-1}\sum_{m=1}^{M}(V_m - \overline{V})^2$$

95% confidence interval: $\overline{V} \pm \frac{1.96\sigma_v}{\sqrt{M}}$

Effect of initial conditions

- Simulation outputs may change depending on initial condition
  - Ignore "early" part of simulation
  - Consider only later part of simulation that is less dependent on initial conditions
    - result may be more reliable

Effect of initial conditions



Comments

- In order to improve the accuracy of your simulation results:
  - Run your experiments a few times using different random seeds, find the average of the measurements
  - For each experiment, run it "long enough"
  - Ignore initial measurements and only used measurements from later part of the experiments
  - Make sure that the events you are measuring occurs often enough over the duration of measurement
    - E.g. if the expected blocking rate is 10^{-6}, it is not sufficient to simulate with only 10^6 events!

Network Simulation

Overview:

- **fundamentals of discrete event simulation**
- analyzing simulation outputs
- **random distribution generation**

Pseudo Number Generator (PNG)

- In order to generate input traffic using random traffic generator, a random number generator is needed.
- However, true random number sequences are difficult to generate. Often a pseudo number generator written in software is used in the simulation
- Example, K&R (The C programming language, pp46)
  - int rand(void) { next = next * 1103515245 + 12345;
    - return (unsigned int) (next/65536) % 32768; }
  - void srand(unsigned int seed) { next = seed; }
- The goal is to generate floating point or integer random numbers with uniform distribution, and random bits.

Output from K&R PNG

- Seed = 0
  - Sequence is 0, 21469, 9989, 22118, 3498, ….
- Seed = 5224
  - Sequence is 14000, 6652, 11479, 2806, 24788 ….
- In a simulation, if the same random seed is used, the outcome should always be the same
  - good for debugging
  - however, it is important to change the random seed if the goal is to study the system under different conditions

PNG

- **Long cycles**: The sequence does not cycle around and repeat itself for a very long period (the random(.) in UNIX has a cycle of $16(2^{31}-1)$)
- **Good numeric distribution**: If the formula is producing random numbers between 0 and N, the number of zeros, ones, twos, … Ns, that it produces should be roughly equal over a long period of time.
- **Lack of predictability**: You have no way to predict what the next number will be unless you know the formula and the **seed** (the initial value).
  - What is the default seed in your PNG?

"Randomness" of PNG

- The random(.) in UNIX generates a number uniformly distributed between 0 and RAND_MAX.
- On my system, RAND_MAX = 2^{31} - 1
- How should you generate a number that is uniformly distributed between 0 to 7 using the random(.) function ?

"Randomness" of PNG

- Take the 3 LSBs, run the simulations for 80K times
- 0: 9924
- 1: 9890
- 2: 9960
- 3: 10037
- 4: 10090
- 5: 9988
- 6: 9978
- 7: 10133
- MSE = 5908

(Seed = 5224)

- Take the 3 MSBs, run the simulations for 80K times
- 0: 10102
- 1: 9992
- 2: 10022
- 3: 9940
- 4: 9948
- 5: 10028
- 6: 9963
- 7: 10005
- MSE = 2429

Non-uniform Random Numbers

- So far, we have been talking only about generating random numbers that are uniformly distributed over some intervals
- How do we generate non-uniformly distributed random numbers?

Random Variate Generation

Goal: It is assumed that a distribution is completely specified and we wish to generate samples from this distribution as input to a simulation model.

Common Techniques:

- **Inverse Transformation**
- Convolution
- Acceptance-Rejection

Random Variate Generation

All these techniques assume that a source of uniform (0, 1) random numbers is available; R_1, R_2,..., where each R_i has:

$$\text{pdf: } f_R(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases} \qquad \text{and}$$

$$\text{cdf: } F_R(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$$

Note: The random variable may be either discrete or continuous.

Random Variate Generation

If the random variable is discrete, ==>

x take on a specific value, and F(x) is a step F^n

If F(x) is continuous over the domain x, ==>

$f(x) = dF(x) / dx$ and

the derivative f(x) is called the pdf.

Mathematically, the cdf is:

$F(x) = P(X \le x) = \int_{-\infty}^{x} f(t)dt$, where F(x) is defined over the range $0 \le F(x) \le 1$, and f(t) represents the value of the pdf of the variable x, when X = t.

Inverse Transformation

- The cdf fuunction, F(x) = u, 0<u<1
- $F^{-1}(u) = \{x: F(x) = u, 0<u<1\}$. If U is a uniform [0,1] r.v., then $F^{-1}(u)$ has distribution function F
- The Inverse transformation method can be used to generate random variates with an arbitrary continuous distribution function F provide F^{-1} is explicitly known.

Random Variate Generation

Example #1

Generate random variates x with density function f(x) = 2x, $0 \le x \le 1$

Solution:

$F(x) = \int_0^x 2t\,dt = x^2, \; 0 \le x \le 1$

Now set F(x) = R ==> $R = x^2$

Next, solve for x, ==> $x = F^{-1}(R) = \sqrt{R}, \; 0 \le r \le 1$

∴ Values of x with pdf f(x) = 2x can be generated by taking the square root of the random, R.

Exponential Distribution

Example #2

Generate random variates x with density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & 0 \le x \\ 0, & x < 0 \end{cases}$$

Solution:

$$F(x) = \int_{-\infty}^{x} f(t)\,dt = \text{Distribution function}$$

$$= \begin{cases} 1 - e^{-\lambda x}, & 0 \le x \\ 0, & x < 0 \end{cases}$$

Exponential Distribution

Now set F(x) = R

Next solve for x, ==>

$1 - e^{-\lambda x} = R$

$e^{-\lambda x} = 1 - R$

$-\lambda x = \ln(1 - R)$

$x = -\{\ln(1 - R)\} / \lambda$

or $= -\{\ln(R)\} / \lambda$

Exponential Distribution

Another way of writing this:

$F(x) = 1 - e^{-\lambda x} = R$

Because of symmetry, F(x) and 1 - F(x) are interchangeable, so,

$e^{-\lambda x} = R$

and $-\lambda x = \ln(R)$

$x = -\{\ln(R)\} / \lambda$

Note $\lambda = 1 / E(x)$, so

$x = -E(x)\ln(R)$