Simulation: Methodology and Statistics

Lecture #4

Contents

1 Introduction and Terminology

- There are systems that cannot be solved adequately with the analytical techniques. For example, it may be hard to analyze the interaction of different protocols over the various layers in the protocol stack
- Generally, for large and complex systems, analytic model formulation and/or solution may require making unrealistic assumptions and approximations.
- With simulation there are no fundamental restrictions towards what models can be solved. However, practical restrictions do exist since the amount of computer time or memory required for running a simulation can be prohibitively large.
  - For example, if we are simulating a model with rare events, we need a large number of simulation runs to get such a rare event at least once and many more replications may be needed to get statistically significant results.
  - Simulation can only show the existence of critical states but can never prove the absence of such states.
- The idea of the simulation is create an estimator of the the target KPI by gathering several measurements for such KPI.
- A deep understanding of statistical methods and necessary assumptions to assert the credibility of obtained results.

1.1 Terminology

- Simulation time (AKA, simulated time): the parameter that corresponds to the real-time clock.
- Run time is the time it takes to execute a simulation program.
- An event is a time point where the states of the system changes or other major things happen (such as the start and the end of the simulation).
- In queuing systems, the two major events are customer arrival and customer departure.
  - An arrival event increases the number of customer in queue by 1 (if the server is busy) or changes the state of the server to busy (if the server was idle).
  - A departure event decreases the number of customer in queue by 1 or changes the server state to idle

2 Simulation Classes

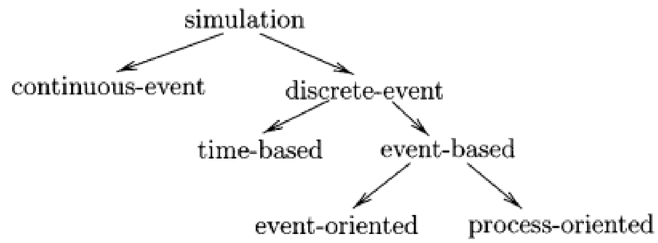

Figure 1: Simulation Classifications

- In *continuous-event* simulations, systems are studied in which the state continuously changes with time. Typically this type refers to systems described by differential equations that are solved numberically.
- In *discrete-event* simulations the state changes take place at discrete points in time.

2.1 Discrete Event Simulation Types

2.1.1 Time-based Simulation

- AKA, synchronous simulation
- the main control loop of the simulation controls the time progress in constant steps.
- easy to implement
- usually time-step so small that the resulting simulation becomes very inefficient.

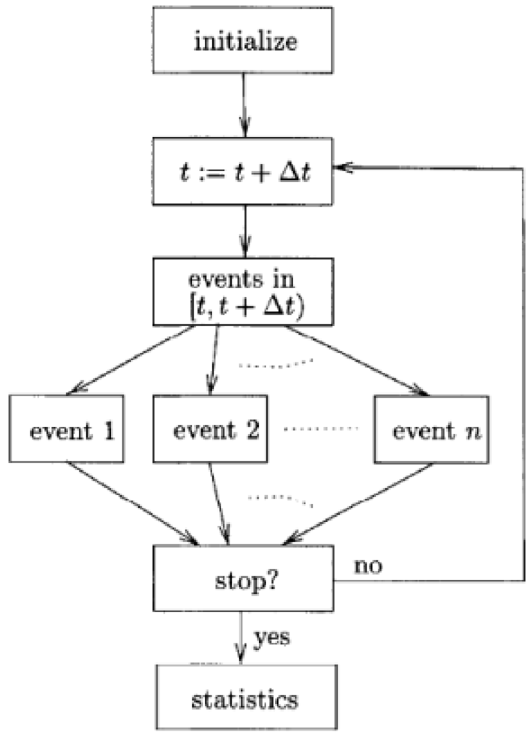

Figure 2: Time-based simulation

2.1.2 Event-based Simulation

- AKA, asynchronous simulation
- time steps of varying length
- easy to implement :)
- All future events are generally gathered in an ordered event list.
- Very common in computer and communication system simulation.
- May not be the best option for simulating systems with small time steps such as microprocessors

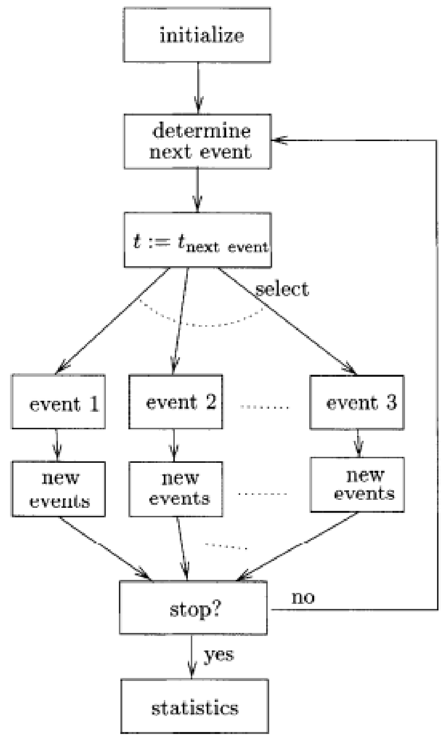

Figure 3: Event-based simulation

2.2 Event-based Simulation Implementations

- The system is divided into entities rather than trying to model it as one big finite state machine.
  - Temporary entities flow through the system
    * e.g. parts, customers or messages that arrive according to a stochastic distrubution
  - Permanent entities stay in the system during the simulation
    * e.g. machines, servers or routers, processing the temporary entities with stochastically distributed processing times
  - Attributes are used for defining the states and properties of individual entities.
- The scheduler (timer) maintains the simulation time and sends timer events to the entities
  - contains the simulation clock and
  - a list of scheduled future events
- Operation:
  - seeks the scheduled event that has the smallest time stamp
  - advances the simulation time
  - sends a timer event to the corresponding entity

2.2.1 Event-oriented Implementation

- In an event-oriented simulation a system is modeled by identifying its characteristic events and then writing a set of event routines that give a detailed description of the state changes taking place at the time of each event.
- The simulation evolves over time by executing the events in increasing order of time.
- the activity following each event is implemented as an eventroutine
- the event-routine may schedule new events and re-schedule existing event
- Control loop
  - event to happen is taken from the list
  - The simulation time is incremented to the corresponding value of the taken event
  - Corresponding eventroutine is invoked
  - simulated system state is changed
  - new events are generated and inserted in the event list
  - some statistics may be collected
  - loop until stopping criteria

2.2.2 Process-oriented Implementation

- The behavior of the system is represented by a set of interacting processes.
- A process is a time-ordered sequence of interrelated events which *describe the entire experience of an entity* as it flows through the system.
- Note that the internal structure of a process-oriented simulation is based on scheduling events as well.
- The logic behind process oriented simulation is similar to constructing a flowchart for the ***flow of an entity*** in the system.
- For example, imagine you are a customer in the bank example, then your experience will be as follows:
  - Create yourself (i.e. arrive to the system).
  - Note your arrival time.
  - Wait in queue if the server is busy.
  - Once the server is idle, move out of the queue and "seize" the server.
  - Note your start of service time and estimate your waiting time in queue.
  - Stay in service (i.e. "delay" your self) for an amount of time equal to your service time.
  - "Release" the server.
  - "Dispose" of your bank visit (leave the system).
- Processes exchange information to communicate state changes
- The simulation can be seen as an execution path of the communicating event-processes

2.3 Conducting Simulation

you can build your own code or use an existing package. The choice is not easy as it may involve one or more of the following factors

- development time of required models
- learning time of existing tool
- comparing your results with others
- level of simulated details

2.3.1 Common known simulators

- General-Purpose Programming Language (GPPL): C, C++ , Java
  - total control over the software development process.
  - the model construction takes con- siderable time.
- Plain Simulation Language (PSL): SIMULA, SIMSCRIPT, SIMAN, GPSS, JSIM, SILK
  - supports DES
  - One also needs to get programming expertise in a new language before executing simulation models.
- Software libraries for simulation: CSIM-19, C++SIM and its Java counterpart JavaSim, baseSim
  - extensive libraries provide aid in simulation model development and execution
  - provide various monitoring and statistical gathering functions with very little or no need to learn any new language.
- Simulation Packages (SPs): OPNET Modeler, ns-2, OMNET++, Arena, Qualnet
  - cover several application domains like TCP/IP networks.

3 Random Number Generation

- True random numbers cannot be generated with a deterministic algorithm
  - We first generate pseudo-random numbers
  - Second, we compute (pseudo) uniformly distributed random numbers
  - Third, generate the required RV with different techniques
- An RNG can be classified as good if
  - can be computed with little cost
  - generated numbers are independent and uncorrelated
  - has a very long period

3.1 Pseudo-random numbers

- Only linear and additive congruential methods are considered

3.1.1 Linear congruential RNGs

$$z_{i+1} = (az_i + c)\, modulo\, m$$

- This algorithm will generate m different values
  - the values m and c are relative primes
  - all prime factors of m should divide a - 1
  - if 4 divides m, then 4 should also divide a – 1
- The number m is called the cycle length
- Cycles are relatively short :(

3.1.2 Pseudo-random numbers

$$z_i = \left(\sum_{j=1}^{k} a_j z_{i-j} \right) modulo\, m$$

- Starting values z_0 through z_{k-1} are generally derived by a linear congruential method
- With an appropriate selection of the factors aj cycles of length $m^k - 1$ are obtained.

USE a DIFFERENT RNG for each random number sequence to be used in the simulation

Have a control on your seed No random seeds

4 Generating non-uniform RV

- Uniform RV Generation: divide the obtained sequence with the largest number
- Various techniques for generating non-uniform RV
  - Inversion Method
  - Rejection method

4.1 Inversion Method

- Define $Z = F(y) \rightarrow Z\ U(0,1)$
- Hence, we generate Z and apply F^{-1} to get the corresponding Y value
- Example: generating exponential distribution

$$F_Y(y) = 1 - e^{-\lambda y} \rightarrow y = -ln(1-z)/\lambda$$

- Example: Generating Erlang-K RV Z
  - We generate k exponential distributions, then

$$z = \sum_{i=1}^{k} x_i$$

- Example: Hyper Exponential distribution

- Example: discrete distributions?

4.2 Rejection Method

- If we can not get the inverse
- Requirements:
  - $f_X(x)$is known,
  - limited domain $[a,b]$
- generate two uniformly distributed numbers u_1 and u_2
- derive the random numbers x,y
  $x = a + (b-a)u_1 \qquad y = cu_2$

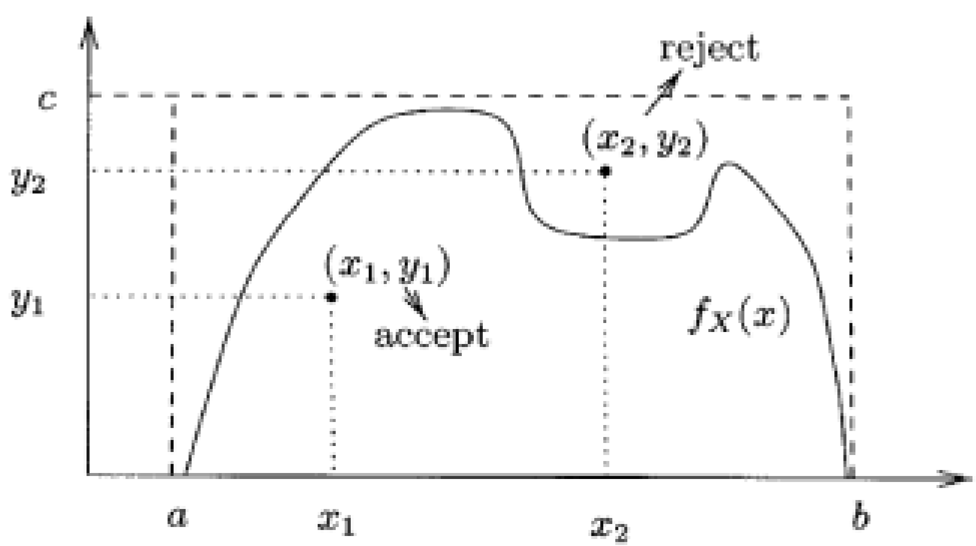

Figure 4: Rejecttion Method

- We repeat the procedure until we encounter a tuple for which the condition holds
- fairly efficient when the area under the density $f_X(x)$ is close to $c(b-a)$

$$\Pr\{x \le X \le x+dx | Y \le f_X(X)\} = \frac{\Pr\{x \le X \le x+dx, Y \le f_X(x)\}}{\Pr\{Y \le f_X(X)\}}$$
$$= \left(\frac{dx}{b-a}\right)\left(\frac{f_X(x)}{c}\right)\left(\frac{1}{(b-a)c}\right)^{-1} = f_X(x)dx.$$

5 Statistical Evaluation

5.1 Obtaining Measurements

- Typically, measurement could be
  - User-oriented, such as time spent by a user in the system
  - System-oriented, average number of users in the system.
- Initial transient removal
  - System initially empty!!
- How long should we remove??
  - Simulate so long (inefficient) such that the impact of the initial collected data is negligible.
    * setting the initial state by an approximate analytical model could help in avoiding the initialization step
  - Truncation method, remove the first l samples of the total n samples, where l is the smallest value such that

    $$\min\{x_{l+1}, ..., x_n\} \neq x_{l+1} \neq \max\{x_{l+1}, ..., x_n\};$$

    in other words the $(l+1)$-th sample is no longer the maximum, nor the minimum of the remaining samples.
- Truncation Method 2
  - Split samples into batches with k elements $k = [n/l]$
  - Start with batch size $l = 2$
  - For every batch, estimate the mean

    $$m_i = \frac{1}{l}\sum_{j=1}^{l} x_{(i-1)l+j} \quad i = 1, \ , k$$

    and the sample variance

    $$\sigma^2 = \frac{1}{k-1}\sum_{i=1}^{k}(m_i - m)^2$$

    where m is the sample mean

- Increase l stepwise until variance starts to decrease and drop first batch. Note that by increasing the batch size l, more and more samples of the initial transient period will become part of the first batch.

5.2 Mean Value and Confidence Intervals

5.2.1 Mean Value Estimation

- Estimator: we define a new stochastic variable, $\tilde{X}$
- If X_i are independent realization of the random variable $\tilde{X}$, the estimator

$$\tilde{X} = \frac{1}{n}\sum_{i-1}^{n} X_i$$

  will be unbiased and consistent

- Successive samples taken from the same simulation **are NOT independent!!!!**
  - e.g. queue response time of successive customers is correlated.

5.2.2 Guaranteeing Independence

- Method 1: Independent replicas
  - repeat the simulation n times with a different seed
  - Estimate the mean of every simulation
  - The obtained n means are considered independent
  - :(simulations has to be repeated several times
- Method 2: Batch means
  - Split measure samples into batches
  - Estimate the mean of each batch
  - Use the batch means as independent samples
  - Not completely independent but is a good approximation.
- Method 3: Regenerative method
  - Batches are defined between ***regenerative points***
  - Main concern is that regeneration points may not be sufficiently many
  - In queuing systems, regeneration points typically occurs when the queue is empty.

5.2.3 Confidence Intervals

- The mean estimator will have normal distribution $N(a, \sigma^2/n)$ according to the central limit theorem. σ^2 is the varience of X_i
- Hence, $Z = \frac{\tilde{X}-a}{\sigma/\sqrt{n}}$is $N(0, 1)$.
- However, since we do not know the variance of the random variable X, we have to estimate it as well. A typical estimator for σ^2 is defined as

$$\tilde{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \tilde{X})^2$$

- The stochastic variable $Z = \frac{\tilde{X}-a}{\tilde{\sigma}/\sqrt{n}}$ has a student distribution. The Student distribution with three or more degrees of freedom is a symmetric bell- shaped distribution, similar in form to the Normal distribution. As $n \to \infty$ student approaches a normal distribution and normal distribution tables can be used
- Note that

$$Pr\{|Z| \leq z\} = Pr\{\left|\frac{\tilde{X}-a}{\tilde{\sigma}/\sqrt{n}}\right| \leq z\} = \beta$$

- β indicates the probability that the mean estimate lies in $[a - z\tilde{\sigma}/\sqrt{n}, a + z\tilde{\sigma}/\sqrt{n}]$. In other words, β **represents the confidence level that the estimated KPI lies in the interval** $[a - z\tilde{\sigma}/\sqrt{n}, a + z\tilde{\sigma}/\sqrt{n}]$**.**

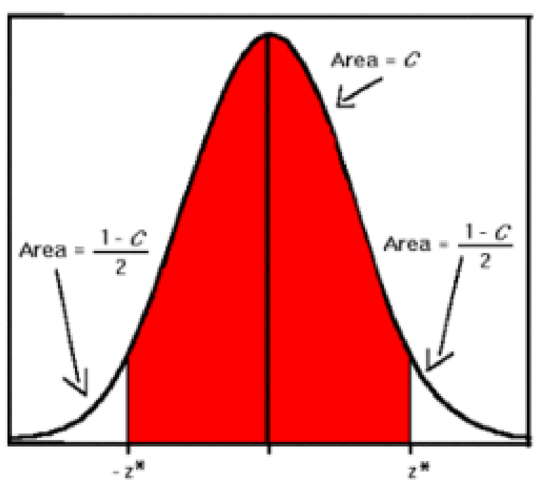

Figure 5: Confident interval

- Common choices for the confidence level are 0.90, 0.95, and 0.99
- For a 95% confidence interval, the area in each tail is equal to $0.05/2 = 0.025$
- A 95% confidence interval for the standard normal distribution, then, is the interval (-1.96, 1.96)

EXAMPLE
the sample mean of the boiling temperatures is calculated as101.82, with standard deviation 0.49. Determine the 95% and 90% confidence intervals.
Solution

- The critical value for a 95% confidence interval is 1.96. A 95% confidence interval for the unknown mean is ((101.82 - (1.96*0.49)), (101.82 + (1.96*0.49))) = (101.82 - 0.96, 101.82 + 0.96) = (100.86, 102.78).
- For a 90% confidence interval for the boiling temperature, the critical value z* for this level is equal to 1.645, so the 90% confidence interval is ((101.82 - (1.645*0.49)), (101.82 + (1.645*0.49))) = (101.82 - 0.81, 101.82 + 0.81) = (101.01, 102.63)

6 Homework

- The average and standard deviation of call durations in a telephony system is measured as 120 sec and 80 sec respectively. Estimate the 95% confidence interval of these measurement. [the corresponding interval to 95% confidence in normal distribution is (-1.96, 1.96)]
- Explain how to generate a uniform RV $U(a, b)$ using inverse method
- Explain how to a Pareto RV using inverse method
- Explain how to a Rayleigh RV using inverse method
- Explain how to a triangular RV using inverse method

References

BOUDEWIJN R. HAVERKORT, “Performance of Computer Communication Systems: A model based approach”