

Introduction to Simulation

Refs: Chap1-3, Chap 24 of Raj Jain's book

Shiv Kalyanaraman

Rensselaer Polytechnic Institute

shivkuma@ecse.rpi.edu

<http://www.ecse.rpi.edu/Homepages/shivkuma>

GOOGLE: "Shiv RPI"

Why simulate ?

- ❑ **1.** Eg: real-system not *available, is complex/costly or dangerous* (eg: space simulations, flight simulations)
- ❑ **2.** Want to quickly evaluate (kick the tires!) and design *alternatives* (eg: networking protocols)
 - ❑ Can get cool graphs, and leverage lots of existing protocol models
- ❑ **3.** Want to evaluate *complex functions* for which closed forms not available (eg: Monte Carlo simulations, Matlab)
- ❑ **4.** Want to *animate or graphically visualize* complex behavior (Eg: protocol animations, performance graphs, Matlab plots)

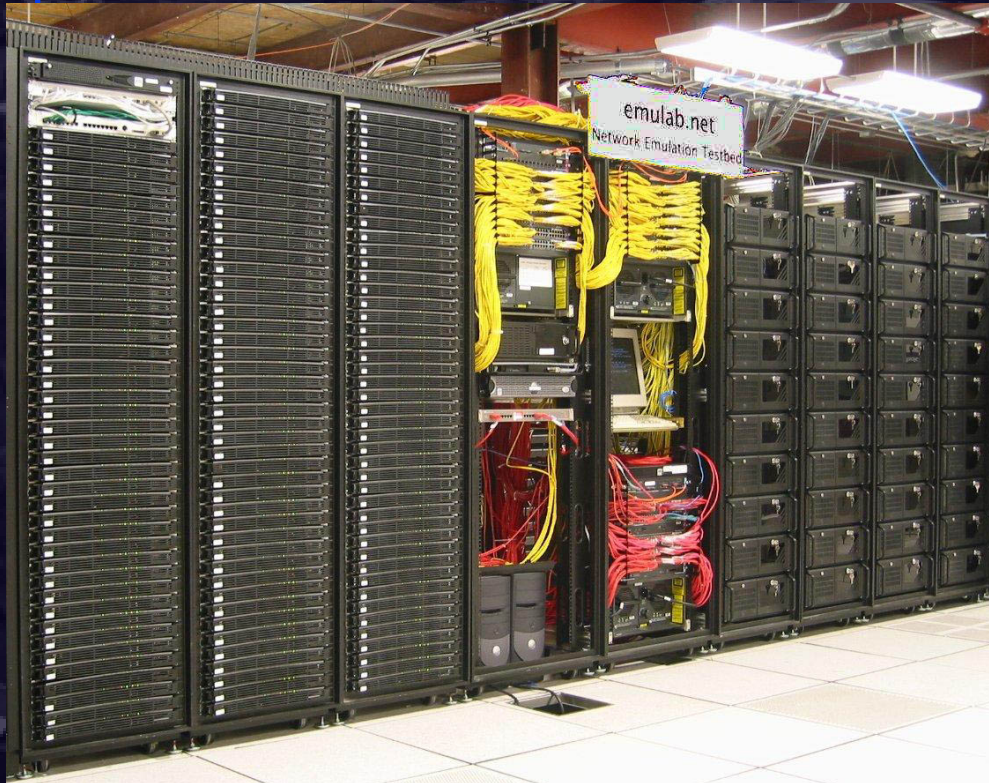
Simulators

- ❑ Networking stuff: ns-2/nam, SSFnet, Opnet
- ❑ Math stuff: Mathematica, Matlab, Maple or Monte-Carlo methods
- ❑ For more on simulation: CS has a detailed class on parallel and distributed simulation (PADS)

Simulation vs Measurement/Implementation

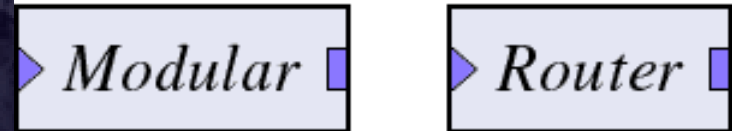
- ❑ **1.** Real system is more credible, but more complex – lot of auxiliary concerns & murphy's law strikes often!
 - ❑ But simulation must be thought of as a first step to real implementation
 - ❑ (I.e. to get a stable design that must be validated by implementation and/or analysis)
- ❑ **2.** Measurement of Internet traffic may not be the same as measurement tomorrow (real, but still random samples!)
 - ❑ Representative measurement traces can be used to drive simulation (I.e. trace-driven simulation)
- ❑ **3.** New emulation platforms: Utah's emulab (next slide)
 - ❑ Takes out the configuration complexity from small/medium sized real experiments!
- ❑ **4. Bottom line:** mix and match both tools depending upon the problem at hand

Utah Emulab and Click: Emulation and Modular Implementation Platforms



Utah's Emulab Testbed: control & interconnect the kernels of 100s of machines just by using ns-2 scripting!!!

MIT's **Click Modular Router**
On Linux:
Forwarding Plane Implns



Shivkumar Kalyanaraman

Simulation Lingo

- ❑ “State”: *variables* whose values define system state.
 - ❑ If a simulation is stopped & restarted – you need to “checkpoint” the state.
 - ❑ Hard problem in distributed simulations!
- ❑ “Event”: *Change* in system state = event
- ❑ Discrete-event model/simulation: events are not continuous but take on discrete values
 - ❑ All network simulations are of this type!

Deterministic vs Probabilistic Results

- ❑ “Deterministic”: results the same if you repeat the simulations
 - ❑ Usually the case if there is no randomness in the input or in the simulation process/protocols etc
- ❑ “Probabilistic”: results vary with every run!
 - ❑ Eg: similar to your RTT estimation exercise!
 - ❑ Random results => cannot make deterministic inferences EVER!
 - ❑ Multiple (truly random) simulation repetitions needed
 - ❑ Get averages, deviations, and to form a confidence interval around results.

Things to remember about Discrete Event Simulation

- ❑ The programming model revolves around “events” (eg: packet arrivals):
 - ❑ Events trigger particular sub-routines
 - ❑ Huge “switch” statement to classify events and call appropriate subroutine
 - ❑ The subroutine may schedule new events! (cannot schedule events for past, i.e., events are causal)
 - ❑ Rarely you might introduce new event *types*
 - ❑ Events have associated with them:
 - ❑ 1. Event type, event data structures (eg: packet)
 - ❑ 2. Simulation time when the event is scheduled
 - ❑ Key event operations: Enqueue (i.e. schedule a event).
 - ❑ Dequeue is handled by the simulation engine

Discrete Event Simulation: Scheduler

- ❑ Purpose: maintain a notion of simulation time, schedule events. A.k.a: “simulation engine”
- ❑ **Simulation time \neq Real time**
 - ❑ A simulation for 5 sec of video transmission *might* take 1 hour!
- ❑ Events are sorted by simulation time (not by type!): **priority queue or heap data structure**
 - ❑ After all subroutines for an event have been executed, control is transferred to the simulation engine
 - ❑ The simulation engine schedules the next event available at the same time (if any)
 - ❑ Once all the events for current time have been executed, simulation time is advanced and nearest future event is executed.
 - ❑ Simulation time = time of currently executing event