Common Simulation Mistakes

Prof. Michael Zink
Overview

- Potential mistakes:
  - Level of detail
  - Improper tool/language
  - Unverified/invalid models
  - Simulation execution
  - Random numbers
- Checklist for simulations
- Simulation tools and languages
- Simulation vs. Emulation
- Verification
- Discussion of paper
Inappropriate Level of Detail

- Level of detail often potentially unlimited
- But more detail requires more time to develop
  - And often to run!
- Can introduce more bugs, making more inaccurate not less!
- Often, more detailed viewed as “better” but may not be the case
  - More detail requires more knowledge of input parameters
  - Getting input parameters wrong may lead to more inaccuracy
    (Ex: disk service times exponential vs. simulating sector and arm movement)
- Start with less detail, study sensitivities and introduce detail in high impact areas
Improper Simulation Tool or Language

- Choice of language can have significant impact on time to develop
- Special-purpose languages can make implementation, verification and analysis easier
- C++Sim (http://cxxsim.ncl.ac.uk/), JavaSim (http://javasim.ncl.ac.uk/), SimPy(thon) (http://simpy.sourceforge.net/) ...
Unverified Models

- Simulations generally large computer programs
- Unless special steps taken, bugs or errors
- Techniques to verify simulation models ➔ we will get to this later
Invalid Models

- No errors, but does not represent real system
- Need to validate models by analytic, measurement or intuition
- Techniques to verify simulation models → we will get to this later
Improperly Handled Initial Conditions

- Often, initial trajectory not representative of steady state
  - Including can lead to inaccurate results
- Typically want to discard, but need method to do so effectively
- Techniques to select initial state
  - Transient removal
  - Long run
  - Proper initialization
  - ... more in lecture on “Statistical Analysis”
Too Short Simulation Run

- Attempt to save time
- Makes even more dependent upon initial conditions
- Correct length depends upon the accuracy desired (confidence intervals)
- Variance estimates ➔ Statistical Analysis
Random Number and Seed Issues

- “Home grown” are often not random enough
  - Makes artifacts
- Best to use well-known one
- Choose seeds that are different
- More in lecture #8 on 2/18
Checklist for Simulation (1)

1. Checks before developing a simulation:
   a) Is the goal of the simulation properly specified?
   b) Is the level of detail in the model appropriate for the goal?
   c) Does the simulation team include personnel with project leadership, modeling, programming, and computer systems backgrounds?
   d) Has sufficient time been planned for the project?

2. Checks during development:
   a) Has the random number generator used in the simulation been tested for uniformity and independence?
   b) Is the model reviewed regularly with the end user?
   c) Is the model documented?
3. Checks after the simulation is running:
   1. Is the simulation length appropriate?
   2. Are the initial transients removed before computation?
   3. Has the model been verified thoroughly?
   4. Has the model been validated before using its results?
   5. If there are any surprising results, have they been validated?
   6. Are all seeds such that the random number streams will not overlap?
Selecting a Simulation Tool/Language

- Simulation language
- General purpose
- Extension of a general purpose language
- Simulation package

- Making the right decision can save work, time, money, ...
Simulation Languages

- Save development time
- Built-in facilities for time advancing, event scheduling, entity manipulation, random variate generation, statistical data collection, and report generation
- More time for system specific issues
- Very readable modular code
General Purpose Language

- Analyst's familiarity
- Easy availability
- Quick startup
- Time for routines for event handling, random number generation
- Other Issues: Efficiency, flexibility, and portability
- Recommendation: Learn at least one simulation language
Simulation Packages

- MatLab  http://www.mathworks.com/
- NS-2   http://nsnam.isi.edu/nsnam/index.php/Main_Page
- Cadence http://www.cadence.com/
- Spice
- WXSIM  http://www.wxsim.com/
- CST    http://www.cst.com/
- ANSOFT http://www.ansoft.com/
Simulation Packages cont’d

- Other domains:
  - Water simulation packages
  - Chemical process simulator
  - Traffic simulator
  - Aerodynamics simulator
Simulation Languages

- Continuous Simulation Languages:
  - Continuous System Modeling Program (CSMP)
  - DYNAMO
  - Differential Equations

- Discrete Event Simulation Languages (Lecture 12):
  - Simula
  - General Purpose Simulation System (GPSS)  
    http://www.webgpss.com/
  - Simulation in Python (SimPy)  
    http://simpy.sourceforge.net/
  - Jist http://jist.ece.cornell.edu/
Simulation Languages cont’d

- Heterogeneous:
  - Simio http://www.simio.biz
  - Simulink http://www.mathworks.com/products/simulink/
  - Modelica http://www.modelica.org/
  - FlexSim http://www.flexsim.com/
Simulation vs. Emulation

- **Emulation**
  - Simulation that runs on a computer/device to make it appear to be something else

- **NISTNet**
  - Emulates network behavior:
    - Delay
    - Loss
    - Bandwidth limitation

- **Wireless Emulator**
  [http://www.cs.cmu.edu/~emulator/]
Verification Techniques I

- Good software engineering practices will result in fewer bugs
- Top-down, modular design
- Assertions (antibugging)
  - Say, total packets = packets sent + packets received
  - If not, can halt or warn
- Structured walk-through
- Simplified, deterministic cases
  - Even if end-simulation will be complicated and non-deterministic, use simple repeatable values (maybe fixed seeds) to debug
- Tracing (via print statements or debugger)
Verification Techniques II

- **Continuity tests**
  - Slight change in input should yield slight change in output, otherwise error

- **Degeneracy tests**
  - Try extremes (lowest and highest) since may reveal bugs

![Graphs showing throughput comparison between Debugged and Undebugged states.](image-url)
Verification Techniques

- Consistency tests – similar inputs produce similar outputs
  - Ex: 2 sources at 50 pkts/sec produce same total as 1 source at 100 pkts/sec
- Seed independence – random number generator starting value should not affect final conclusion (maybe individual output, but not overall conclusion)
Discussion of Paper

- “MANET Simulation Studies: The Incredibles”, S. Kurkowski, T. Camp, M. Colagrosso
Summary

- Overview on common simulation mistakes
- Simulation tool selection
- Overview of simulation tools
- Discussion of simulation mistakes in MANET papers