# Data Communications & Networks

## Session 7– Sub-Topic 1
## Discrete Event Simulation in PNSimulator

**Dr. Jean-Claude Franchitti**

*New York University*
*Computer Science Department*
*Courant Institute of Mathematical Sciences*

# Events

- Events are unique and represent an arrival of some new information into the system. For example:
  - Packet arrives from the link layer
  - Timer expires
  - Application generates a message
  - User closes an application

# Event Relationships

- Events have a defined position in time in relation to other events
  - *Causal Relationship*
    - One event cannot occur unless another event has already occurred
    - If Event A causes Event B, then A must precede B in time
    - If B, then A
  - *Random*
    - An event depends on nothing other than a probability distribution
      - A customer walks into a store
      - An application generates a request

# Delivery of Events

- All events are created as “future” events
- Based on event and perhaps current state, calculate a time at which the event will occur
- Insert event into a time-ordered queue
- Remove events from the head of the queue and deliver event:
  - Advance clock to event time
  - Call event handler method
  - Repeat until no more events on queue

# Event Generation

- After delivery of an event, call all random event generator:
  - If any random event is to happen, its generator will return an event and future time
- Event handlers can generate events. These are typically causal events
  - Example
    - your send() method is an event handler for the Event *SendApplicationMessage*
    - *Your send method starts a timer which causes a TimerEvent to be put on queue with future time = now + timeout value*

# PNSimulator Events are Atomic

- In PNSimulator, each event is fully handled before any new event handler is called
- This means  that all of your handlers are *Threadsafe*
- This does not insure that your code can be shared however!  That's your job
- What does this mean?

# Share Code / Separate Context

- All context dependent variables must be allocated such that any invocation of a handler L for user N can retrieve context(L,N)
- Usually in communications systems L is layer-specific
- For example, sequence numbers in TCP are distinct for each socket
- So you must keep your Node A variables distinct from your Node B variables