UCF Stands For Opportunity

CDA6530: Performance Models of Computers and Networks

Chapter 8: Discrete Event Simulation (DES)

SCHOOL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE

Simulation Studies

- Models with analytical formulas
  - Calculate the numerical solutions
    - Differential equations ---- Matlab Simulink
      - Or directly solve if has closed formula solutions
    - Discrete equations --- program code to solve
  - The mean value formulas for stochastic events
    - Solutions are only for the mean values
  - If you derive models in your paper, you must use real simulation to verify that your analytical formulas are accurate

Simulation Studies

- Models without analytical formulas
  - Monte Carlo simulation
    - Generate a large number of random samples
    - Aggregate all samples to generate final result
    - Example: use U(0,1) to compute integral
  - Discrete-time simulation
    - Divide time into many small steps
    - Update system states step-by-step
    - Approximate, assume system unchanged during a time step
  - Discrete event simulation (DES)
    - Accurate
    - Event-driven

Discrete-Time Simulation

- System is assumed to change only at each discrete time tick
  - Smaller time step, more accurate simulation
- Why use it?
  - Simpler than DES to code and understand
  - Fast, if system states change very quickly

Discrete-Time Simulation

While (simulation not complete){

1). Time tick: k ++;

2). For system's node i (i=1,2,⋯)

3). Simulate what could happen for node i during the last time step (k-1 → k)

4). Update the state of node i if something happens to it

5). Output time tick k's system's states (e.g., status of every node in the system)

}

Discrete-Time Simulation

- Note: when computing system node i's state at time tick k, it should be determined only by all other system nodes' states at time tick k-1
  - Be careful in step 4): not use node j's newly updated value at current round
    - Newly updated value represents state at the beginning of next round.
  - Example: worm propagation simulation

Discrete-Time Simulation

- Another example: one line of nodes
  - $X_i(t) = (U-0.5) + (X_{i-1}(t-1) + X_{i+1}(t-1)) / 2$

```
Simul_N = 1000; n=100; X = ones(n,1);
for k=1:Simul_N,
    U = rand(n,1);
    X(1) = (U(1) - 0.5) + X(2)/2;
    for i=2:n-1,
        X(i) = (U(i) - 0.5) + (X(i-1) + X(i+1)) / 2;
    end
    X(n) = (U(n) - 0.5) + X(n-1) / 2;
end
```

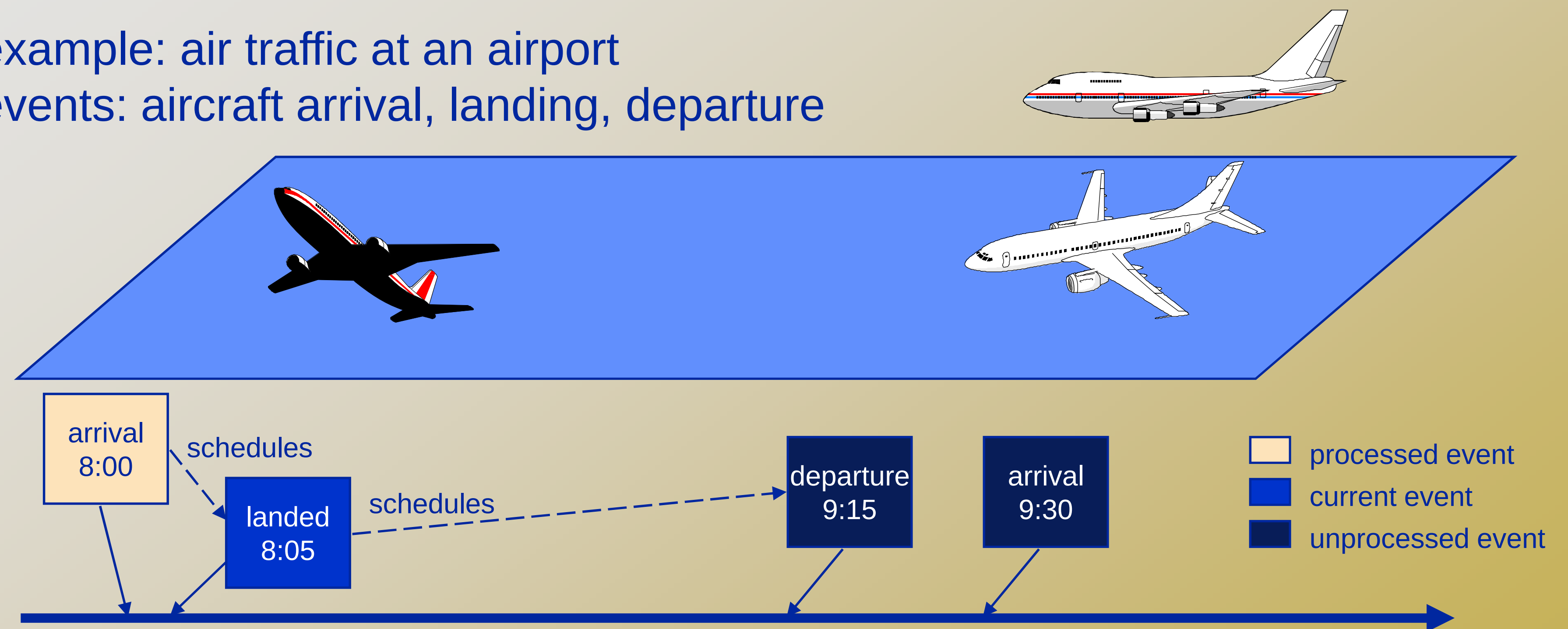
What's Wrong?

Time Concept

- *physical time:* time in the physical system
  - Noon, Oct. 14, 2008 to noon Nov. 1, 2008
- *simulation time:* representation of physical time within the simulation
  - floating point values in interval [0.0, 17.0]
  - Example: 1.5 represents one and half hour after physical system begins simulation
- *wallclock time:* time during the execution of the simulation, usually output from a hardware clock
  - 8:00 to 10:23 AM on Oct. 14, 2008

Discrete Event Simulation Computation

example: air traffic at an airport
events: aircraft arrival, landing, departure



- Unprocessed events are stored in a pending event list
- Events are processed in time stamp order

From: http://www.cc.gatech.edu/classes/AY2004/cs4230_fall/lectures/02-DES.ppt

DES: No Time Loop

- Discrete event simulation has no time loop
  - There are events that are scheduled.
  - At each **run** step, the next scheduled event with the *lowest* time schedule gets processed.
    - The current time is then *that* time, the time when that event is supposed to occur.
- Accurate simulation compared to discrete-time simulation
- Key: We have to keep the list of scheduled events *sorted* (in order)

Variables

- Time variable t
  - Simulation time
  - Add time unit, can represent physical time
- Counter variables
  - Keep a count of times certain events have occurred by time t
- System state (SS) variables
- We focus on queuing systems in introducing DES

Interlude: Simulating non-homogeneous Poisson process for first T time

- Nonhomogeneous Poisson process:
  - Arrival rate is a variable $\lambda(t)$
  - Bounded: $\lambda(t) < \lambda$ for all $t<T$
- Thinning Method:
  1. $t=0$, $I=0$
  2. Generate a random number U
  3. $t=t-\ln(U)/\lambda$. If $t>T$, stop.
  4. Generate a random number U
  5. If $U \leq \lambda(t)/\lambda$, set $I=I+1$, $S(I)=t$
  6. Go to step 2
- Final I is the no. of events in time T
- $S(1), \cdots, S(I)$ are the event times
- Remove step 4 and condition in step 5 for homogeneous Poisson

Subroutine for Generating T_s

- Nonhomogeneous Poisson arrival
  - T_s: the time of the first arrival after time s.

1. Let t =s
2. Generate U
3. Let t=t-ln(U)/λ
4. Generate U
5. If U$\leq$ λ(t)/λ, set T_s=t and stop
6. Go to step 2

Subroutine for Generating T_s

- Homogeneous Poisson arrival
  - T_s: the time of the first arrival after time s.

1. Let t =s
2. Generate U
3. Let $t=t-\ln(U)/\lambda$
4. Set T_s=t and stop

M/G/1 Queue

- Variables:
  - Time: t
  - Counters:
    - N_A: no. of arrivals by time t
    - N_D: no. of departures by time t
  - System state: n – no. of customers in system at t
- Events:
  - Arrival, departure (cause state change)
  - Event list: EL = t_A, t_D
    - t_A: the time of the next arrival after time t
    - T_D: the service time of the customer presently being served
- Output:
  - A(i): arrival time of customer i
  - D(i): departure time of customer i

- Initialize:
  - Set $t=N_A=N_D=0$
  - Set SS n=0
  - Generate T_0, and set $t_A=T_0$, $t_D=\infty$
  - Service time is denoted as r.v. Y

- If ($t_A \leq t_D$)
  - $t=t_A$ (we move along to time t_A)
  - $N_A = N_A+1$ (one more arrival)
  - n= n + 1 (one more customer in system)
  - Generate T_t, reset $t_A = T_t$ (time of next arrival)
  - If (n=1) generate Y and reset $t_D=t+Y$ (system had been empty and so we need to generate the service time of the new customer)
  - Collect output data $A(N_A)=t$ (customer N_A arrived at time t)

- If ($t_D<t_A$) (Departure happens next)
  - $t = t_D$
  - n = n-1 (one customer leaves)
  - $N_D = N_D+1$ (departure number increases 1)
  - If (n=0) $t_D=\infty$; (empty system, no next departure time)

    else, generate Y and $t_D=t+Y$ (why?)
  - Collect output $D(N_D)=t$

Summary

- Analyzing physical system description
- Represent system states
- What events?
- Define variables, outputs

- Manage event list
- Deal with each top event one by one