

# Chapter 24 Introduction to Simulation

- Why simulation ?
  - No physical system or difficult to use different workloads or environments
- Required skills for the whole process
  - Knowledge of the system under investigation
  - Model formulation - System analyst skills
  - *"Model programming - Model building skills"*
  - Data collection skills (monitoring)
  - Input data representation - statistical skills (workload characterization)
  - Output data analysis (pictorial/ratio games)
  - Experimental design
  - AND Team Management

# Key Questions on Simulation

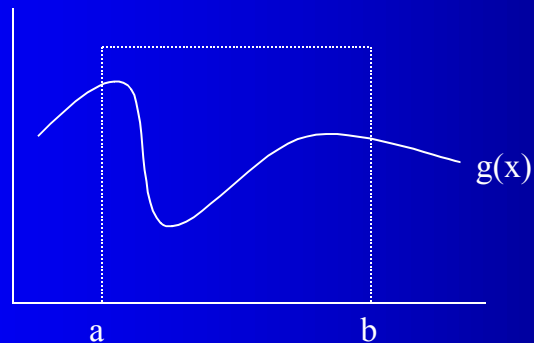
- What are the common mistakes in simulation and why most simulations fail?
- What language should be used for developing a simulation model? (24)
- What are different types of simulations? (24)
- How to schedule events in a simulation? (24)
- How to verify and validate a model? (25)
- How to determine that the simulation has reached a steady state? (25)
- How long to run a simulation? (25)
- How to generate uniform random numbers? (26)
- How to verify that a given random number generator is good? (27)
- How to select seeds for a random number generators? (27)
- How to generate random variables with a given distribution? (28)
- What distributions should be used and when? (29)

# Types of Simulation

- Static vs “Dynamic”
  - time is a variable or not
- Continuous vs “Discrete” Time (if dynamic)
  - the system state changes at all times or only at discrete times
- “Continuous” vs Discrete State
  - the state variables take continuous values or not
- Deterministic vs “Stochastic” (Probabilistic)
  - randomness in the inputs or in the model operations or not

# Static : Monte Carlo Simulation

- Named by John von Neumann (gambling similarity) 1961
- Problem-solving technique
- Example : Compute  $I = \int_a^b g(x)dx$



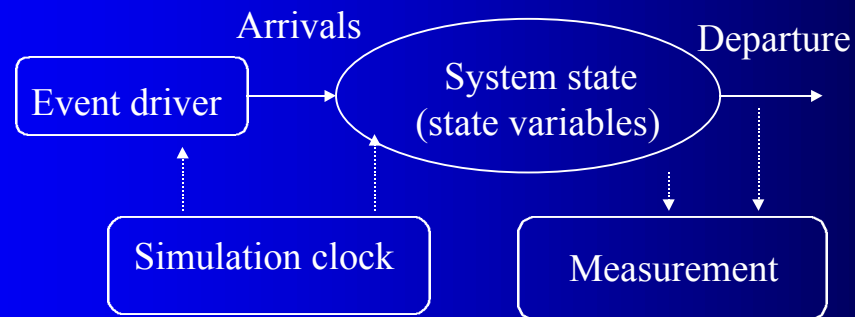
- Throw random points  $(x,y)$  in the bounding box : N
- and count the points below the curve  $g(x)$  : K
- $I = \text{Area of the bounding box} * K / N$

## Deterministic : Trace-Driven Simulation - OX

- Credibility
- Accurate Workload: Models correlation and interference
- Less Randomness (if not deterministic)
- Similarity to the Actual Implementation
  
- Complexity
- Representativeness: Workload changes with time, etc.
- Finiteness: Few minutes fill up a disk
- Single Point of Validation: One trace = one point
- Trade-Off: Difficult to change workload

# Dynamic, stochastic, discrete-time : Discrete-Event Simulation

- State Variables - define the state of the system
- Event - a change in the system state



# Components of Discrete-Event Simulations

1. Event Driver (Scheduler)
2. Simulation Clock and a Time Advancing Mechanism
3. System State Variables
4. Event Routines: One per event
5. Input Routines: Get model parameters
6. Report Generator
7. Initialization Routines: Set the initial state, Initialize seeds
8. Trace Routines: On/off feature

# The Able-Baker Carhop Problem

- Drive-in restaurant with two carhops, Able and Baker
- Car interarrival distribution
  - 1 min : 0.25
  - 2 min : 0.40
  - 3 min : 0.20
  - 4 min : 0.15
- Service distribution - Able & Baker

– 2 min : 0.30	0
– 3 min : 0.28	0.35
– 4 min : 0.25	0.25
– 5 min : 0.17	0.20
– 6 min : 0	0.20
- How long customers wait in average ?

## The ABC Problem (cont'd)

- How to generate the arrival time and service time ?
- Interarrival distribution (w/cumulative probability)
  - 1 min : 0.25(0.25)
  - 2 min : 0.40(0.65)
  - 3 min : 0.20(0.85)
  - 4 min : 0.15(1.00)
- Service distribution - Able & Baker (w/cumulative prob.)

– 2 min : 0.30(0.30)	0(0.00)
– 3 min : 0.28(0.58)	0.35(0.35)
– 4 min : 0.25(0.83)	0.25(0.60)
– 5 min : 0.17(1.00)	0.20(0.80)
– 6 min : 0	0.20(1.00)

# DES - Event Driver

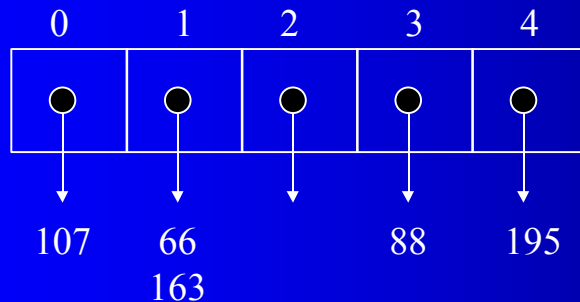
- Finds time of next event in system
  - arrival time
  - departure time (service demand)
- Updates time to that time
- Invokes event routines to update state variables
  - arrival event routine : place new customer in queue and schedule
  - departure event routine : free the occupied resource and schedule

## DES - Future Event List (FEL)

- Future Event List (Event Set) = Ordered linked list of future event notices
- Operations from the FEL
  - Enqueue new events
  - Find and dequeue earliest timestamp
- Data structure for the FEL - basically a priority queue
  1. Ordered Linked List: SIMULA, GPSS, and GASP IV
  2. Indexed Linear List: set of linked lists ( $t_{\sim}$ ,  $t+\Delta t_{\sim}$ , ...) - easier insertion
  3. Tree Structures: usually a binary tree

# FEL - Calendar Queues

- R.Brown, "Calendar Queues: A Fast  $O(1)$  Priority Queue Implementation for the Simulation Event Set Problem," CACM, Oct. 1988 (P30)
- All models do a lot of FEL operations - look for better data structure
- One variation of Indexed Linear List

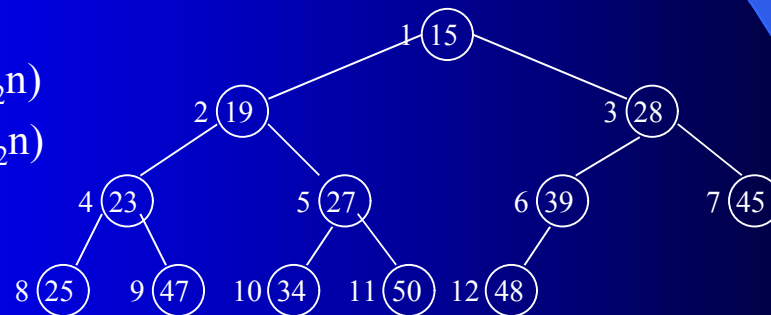


5 buckets (N)  
Bucket width (W) = 10  
Current time = 63  
Number of events (E) = 5

- Adjust bucket width and # buckets as simulation proceeds
  - when  $E > 2xN$ , twice the number of buckets
  - when  $E < N/2$ , half the number of buckets

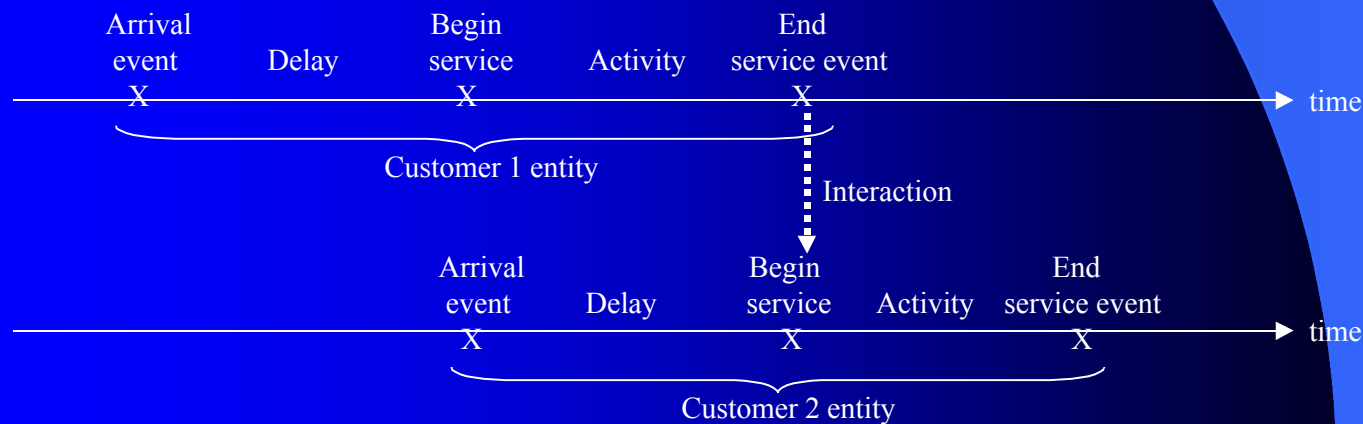
# FEL - Heap

- Special case of (Binary) Tree Structure
  - Event is a node in the binary tree (up to two children)
  - Event time for each node is smaller than that of its children
  - Root is the earliest event
  - Heap can be stored as arrays - parent in  $i$  -> children in  $2i$  and  $2i + 1$
- 
- Insertion ?  $O(\log_2 n)$
  - Removal ?  $O(\log_2 n)$ 
    - place at leftest
    - and swap



# Discrete-Event Simulation (Again)

- Event: a change in the system state
- Entity: an object or component of interest (customer, server, machine)
- Activity: a time duration of specified length (service time, interarrival time)
- Delay: a duration of time of unspecified indefinite length (queueing delay)
- Process: life cycle of one entity (sequence of events, activities and delays)



# Discrete-Event Simulation : World View

- Event-scheduling world view
- Process-interaction world view (US)
  - define simulation model in terms of entities and processes with their interactions
  - many processes are simultaneously active in a model
  - interactions between them are quite complex
- Activity scanning world view (Europe)
  - uses fixed time increment and scan to decide any activities can begin now
  - mixed approach : three-phase approach
  - Phase A : remove and process the imminent event from FEL
  - Phase B : execute B-type events (unconditional)
  - Phase C : scan the conditions and trigger C-type activities (conditional)

# Simulation Language

- “The language was originally invented because the author wanted to write some event-driven simulations for which Simular67 would have been ideal, except for efficiency considerations.”

by Bjarne Stroustrup

- “History of programming language is that of the simulation programming language.”

by R.E.Nance, “A History of Discrete Event Simulation Programming Languages,” ACM SIGPLAN Notices, Mar.1993

## Simulation Language (cont'd)

### ➤ Simulation language

- Save development time
- Built-in facilities for time advancing, event scheduling, entity manipulation, random variate generation, statistical data collection, and report generation
- GPSS, SLAM : Block-structured language (FORTRAN-based)  
SIMSCRIPT II.5 : English-like problem description language  
MODSIM III : Object-oriented language (Modular2-based)  
SIMULA : Problem description language (ALGOL-based)  
CSIM : Process-oriented language (C,C++ based)

## Simulation Language (cont'd)

- General purpose language
  - Analyst's familiarity with the language (C, FORTRAN)
- Extension of a general purpose language
  - compromise (SMPL, GASP)
- Simulation package
  - Input dialog, time saving, special purpose
  - NETWORK II.5 : Computer systems
  - OPNET : Communication networks including wireless networks
  - COMNET III : Communication networks
  - SIMFACTORY : manufacturing operations