Basic Simulation Modeling

Chapter 1

Jiang Li, Ph.D.
Department of Computer Science

1.1 The Nature of Simulation (1)

- Simulation
  - Using computers to imitate (simulate) the operations of a real-world facility or process (the system)
- Model
  - Mathematical or logical representation of system behavior, possibly with approximation
  - If simple enough, get analytic solution
  - Otherwise, use simulation to get data for estimation
    - Model is evaluated numerically in a simulation

Importance of Simulation

- Know the results without really doing it
  - Saves the cost
- Simulation is one of the most widely used operations-research and management-science techniques
  - Along with "math-programming" and "statistics"
  - Numerous application areas
    - Manufacturing, computer networking, military operations, transportation, business processes and much more

Impediments

- Models to study large-scale systems tend to be complex and leads to complex computer programs
  - Eased by simulation packages
- Simulation of complex systems takes a lot of computer time
  - Eased by computer technology advancement
- Ignorance of simulation methodology
  - Need to compose efficient models, careful programming, proper data collection and result analysis
  - An integral part of this course

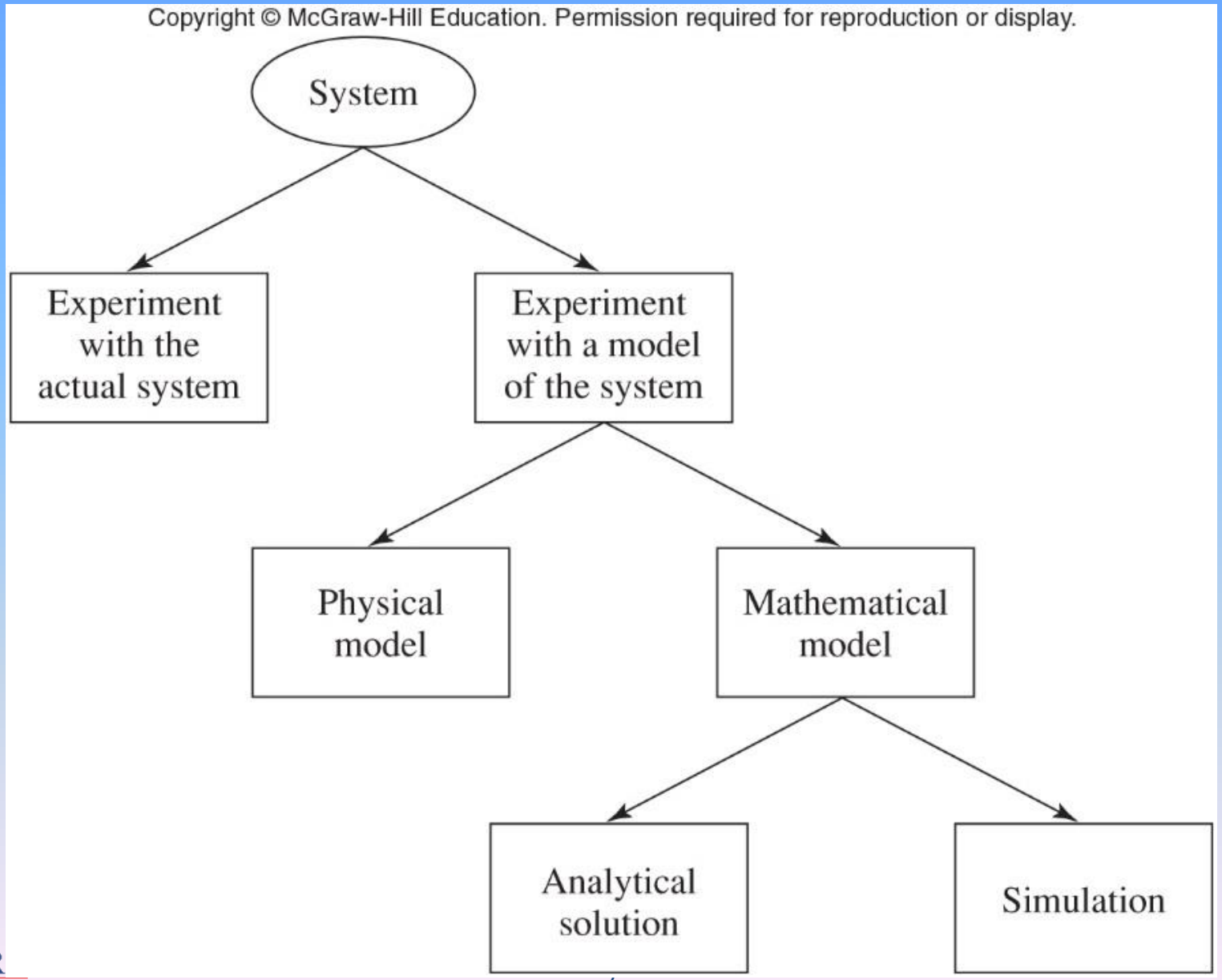
1.2 Systems, Models, and Simulation

- System
  - A collection of (interacting) entities
    - Each characterized by attributes
  - Could be a part of an overall system in the real world, depending on the study
- System state
  - The value of variables to describe a system at a particular time

Types of Systems

- Discrete
  - State variables change at separated points in time
  - E.g. a store (with the # customers as state variable)
- Continuous
  - State variables change continuously with time
  - E.g. a moving vehicle (with the position as state variable)
- Very few systems in practice are purely of one type, but one type of changes usually dominates

Ways to Study A System



Ways to Study A System (1)

- Experiment with the actual system vs. Experiment with a model of the system
  - First one preferred, if cost-effective
  - Usually it is not, or the system doesn't even exist
  - Model validity: does the model accurately reflects the system

Ways to Study A System (2)

- Physical (iconic) model vs. mathematical model
  - A mathematical model good for one purpose may not be good for others, e.g. $d = rt$
- Analytical solution vs. simulation
  - First one desirable if feasible
  - Simulation is needed in most situations

Simulation Models (1)

- Static vs. dynamic
  - Static: time is not a factor
  - Dynamic: the system evolves over time
- Deterministic vs. stochastic
  - Deterministic: output is decided once input is given
  - Stochastic: some components involves randomness
    - Only provides estimate
    - Must be treated carefully, e.g. one run does not tell much

Simulation Models (2)

- Continuous vs. discrete
  - A discrete model may be used to model a continuous system

1.3 Discrete-Event Simulation

- Event
  - An instantaneous occurrence that may change the system's state
  - Some may be used to schedule decisions
- Simulation clock
  - Variable that gives the current value of simulated time (not wall clock time)
  - Methods for advancing the simulation clock
    - Next-event time advance (used the most)
    - Fixed-increment time advance

Time-Advance Mechanisms

- Next-event time advance
  - Clock initialized to zero
  - Times of events are determined
  - Clock advanced to time of first event
    - State of system updated
  - Clock advanced to time of next event
    - State of system updated
  - Continues until stopping condition satisfied
  - Inactivity periods (of various length) are skipped
- Fixed-increment time advance
  - Does not skip over inactive periods
    - Can use up a lot of computer time

Next-Event Time-Advance Example (1)

- Single-server queueing system
  - t_i: arrival time of the ith customer ($t_0 = 0$)
  - $A_i = t_i - t_{i-1}$
  - S_i: Time to serve the ith customer
  - D_i: Delay in queue of the ith customer
  - $c_i = t_i + D_i + S_i$
  - e_i: occurrence time of ith event of any type

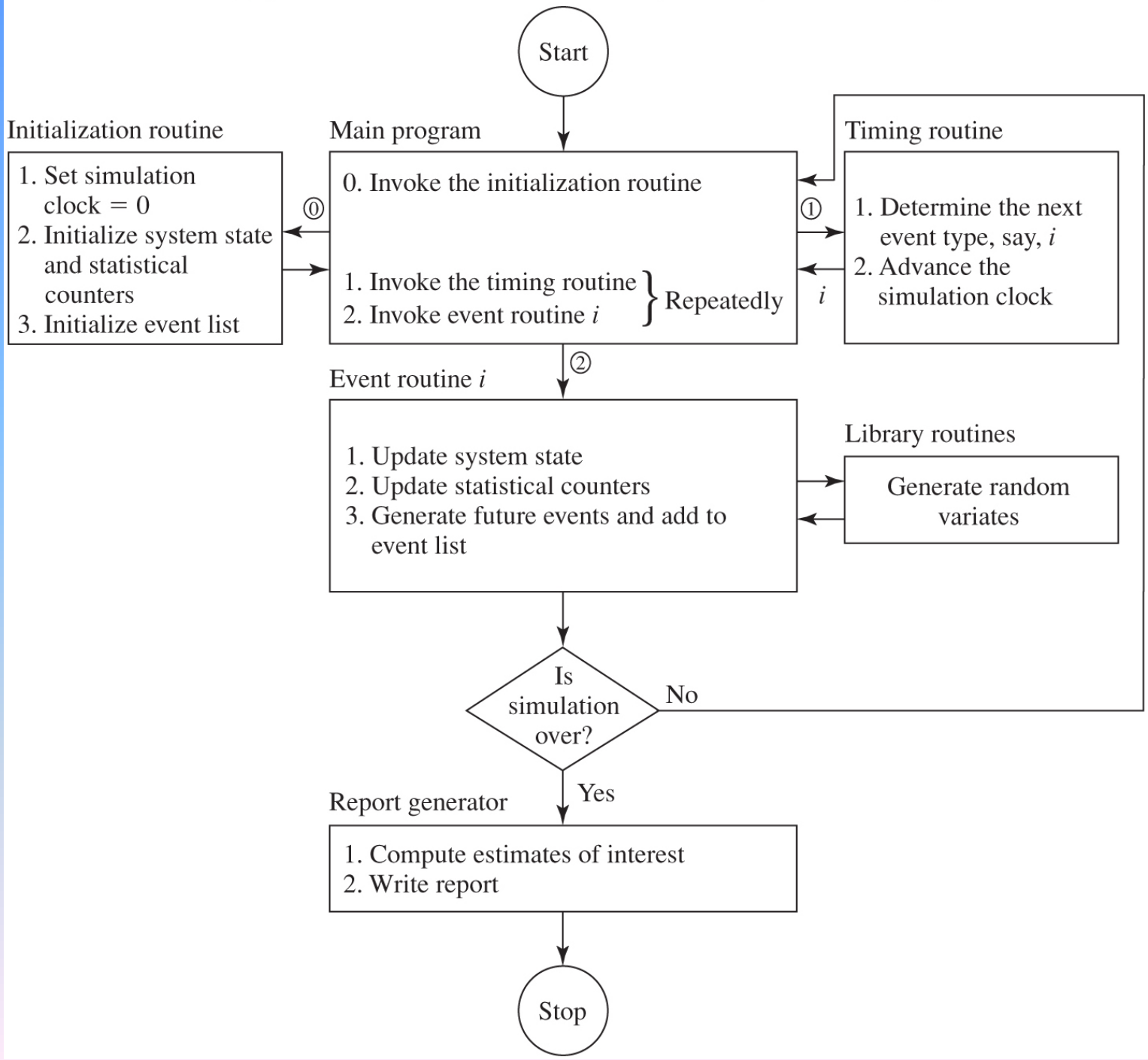
Next-Event Time-Advance Example (2)

- Known probability distribution for inter-arrival time and service time
  - Cumulative distribution functions known
    - F_A: CDF of inter-arrival time
    - F_S: CDF of service time
- Generate A_1 from F_A and get t_1
  - $e_1 = t_1$
- Generate S_1 from F_S and get S_1
- Generate A_2 from F_A and get t_2
- e_2 = ?

Components of a Discrete-Event Simulation Program

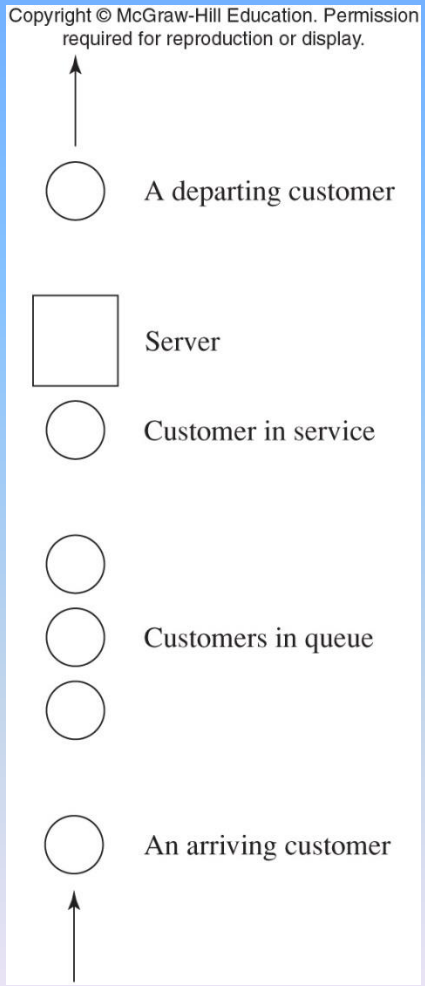
- System state
  - Lists of entity records
- Simulation clock
- Event list
- Statistical counters
- Routines
  - Initialization, timing, event, library, report generator, main

Components of a Discrete-Event Simulation Program



1.4 Simulation of a Single-Server Queuing System

- Example system: a one-operator barbershop
  - Interarrival times A_1, A_2…A_n are independent, identically distributed random variables
  - Customer service times are S_1, S_2…
  - FIFO
  - Measure the performance by looking at the estimates of three quantities
    - Expected average delay of customers in queue
    - Expected average number of customers in queue
    - Expected utilization of the server



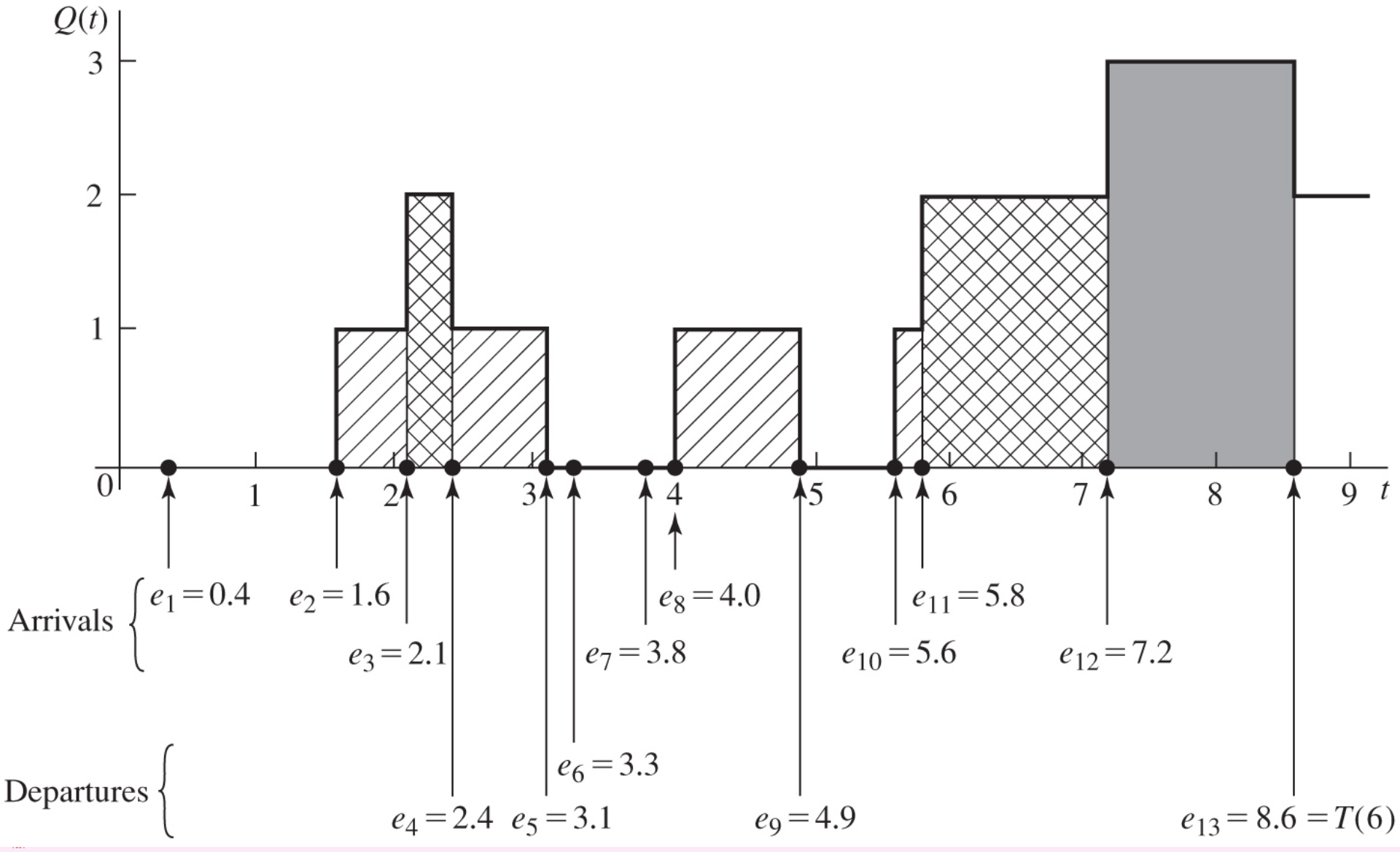
Estimate Average Queueing Delay

- $\hat{d}(n) = \frac{\sum_{i=1}^{n} D_i}{n}$
- $D_1 = 0$, $D_i >= 0$ $(i > 1)$
- $\hat{d}(n)$is an estimator based on size-1 sample

Estimate Average Queue Length

- $q(n) = \sum_{i=1}^{\infty} i\, p_i$
  - p_i: probability of the queue length Q(t) being i, or, the proportion of the time Q(t) = i
- $\hat{q}(n) = \sum_{i=1}^{\infty} i\, \hat{p}_i$
- $\hat{q}(n) = \frac{\sum_{i=1}^{\infty} iT_i}{T(n)}$

Example Realization



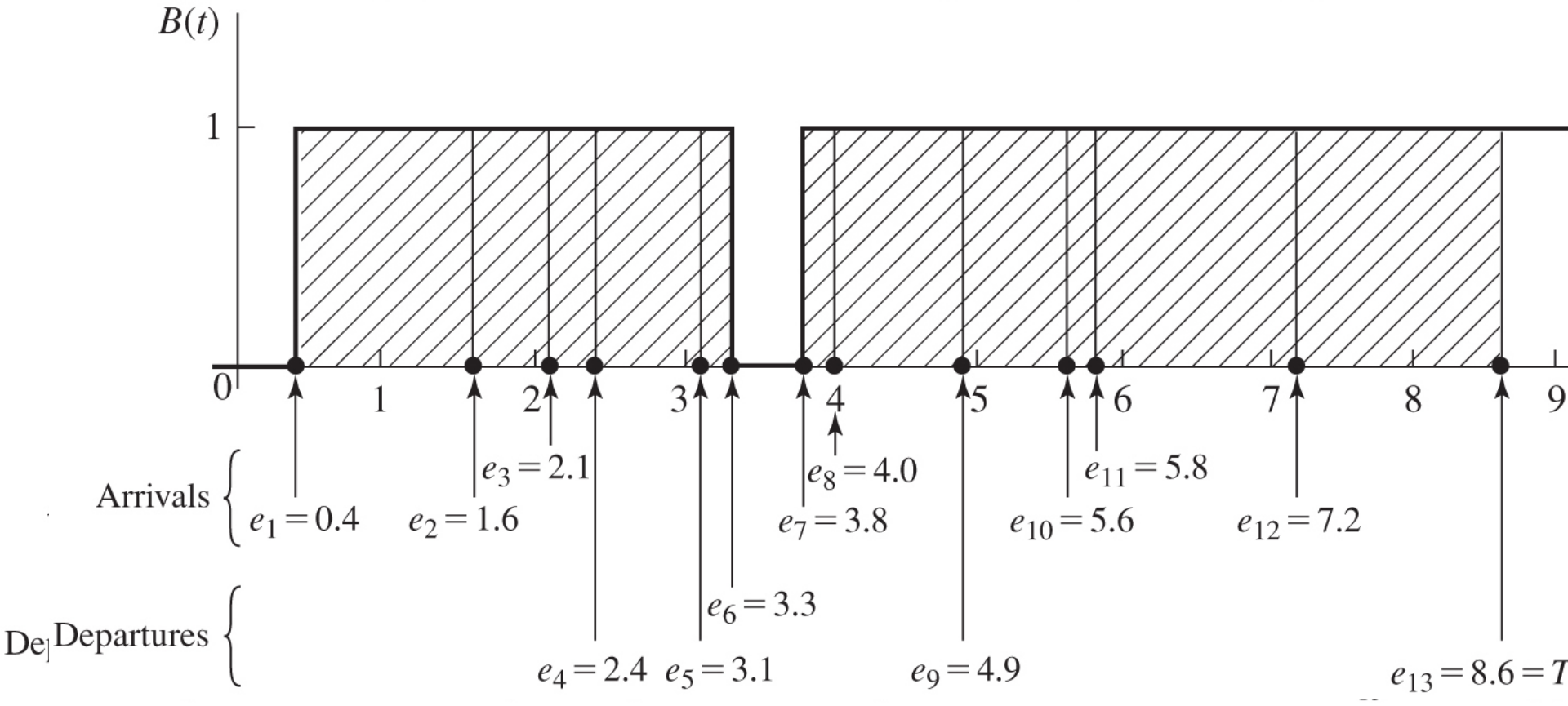
Estimate Average Queue Length

- $T_0 = (1.6 - 0.0) + (4.0 - 3.1) + (5.6 - 4.9) = 3.2$
- T_1, T_2, T_3?
- $\hat{q}(n) = \frac{\sum_{i=1}^{\infty} iT_i}{T(n)}$ = ?
- $\hat{q}(n) = \frac{\int_0^{T(n)} Q(t)dt}{T(n)}$ (preferable)

Estimate Server Utilization

- $B(t)=\begin{cases} 1 & \text{if the server is busy at time t} \\ 0 & \text{if the server is idle at time t} \end{cases}$
- $\hat{u}(n) = \frac{\int_0^{T(n)} B(t)dt}{T(n)}$

Example Realization



$$\hat{u}(n) = \frac{\int_0^{T(n)} B(t)dt}{T(n)} = ?$$

Recap of Performance Measure

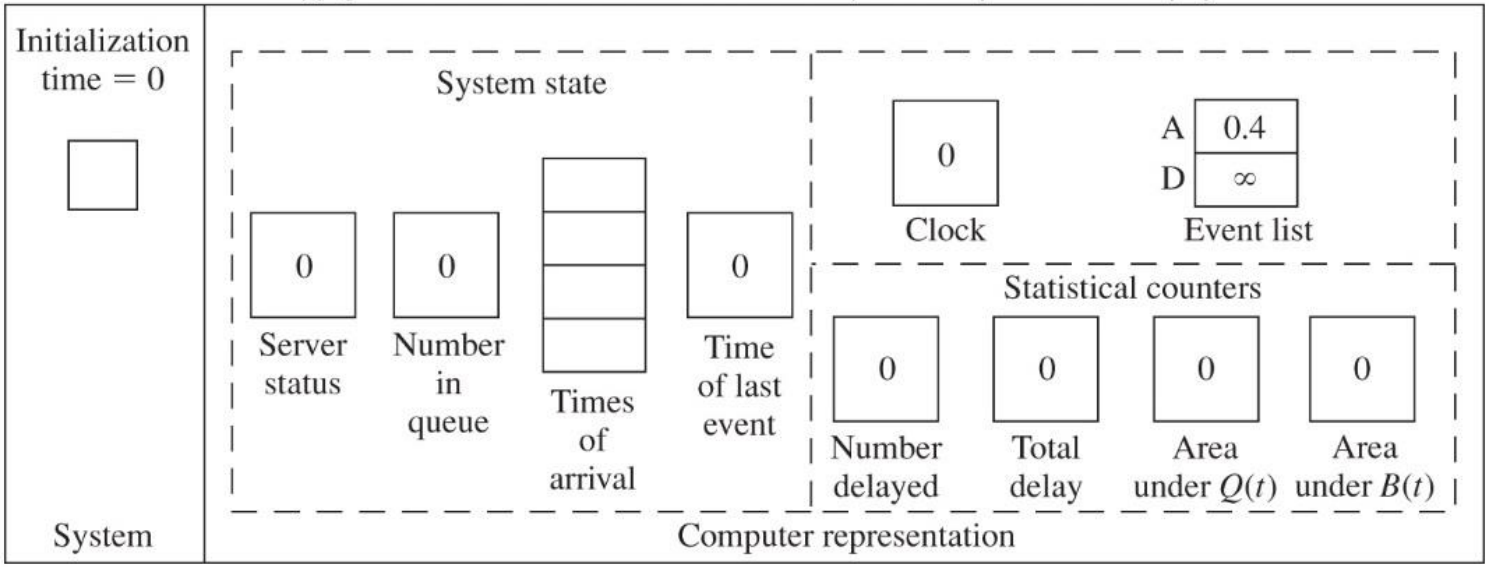
- Discrete-time statistic
  - $\hat{d}(n) = \frac{\sum_{i=1}^{n} D_i}{n}$
  - Another example: Max. delay
- Continuous-time statistic
  - $\hat{q}(n) = \frac{\int_0^{T(n)} Q(t)dt}{T(n)}$
  - $\hat{u}(n) = \frac{\int_0^{T(n)} B(t)dt}{T(n)}$
  - Another example: proportion of time with 5 customers in queue

Events and State Variables

- Events for the barbershop example
  - Arrival of a customer
  - Departure of a customer
- State variables
  - Status of the server
  - Number of customers in the queue
  - Arrival time of each customer currently in queue
  - Time of most recent event

Simulation of a Single-Server Queuing System

- Initialization
  - State of the system at $t = 0$
- Sequence of events
  - $t = 0.4$ arrival of customer 1
  - $t = 1.6$ arrival of customer 2
  - $t = 2.1$ arrival of customer 3
  - $t = 2.4$ departure of customer 1
  - Time units must be the same



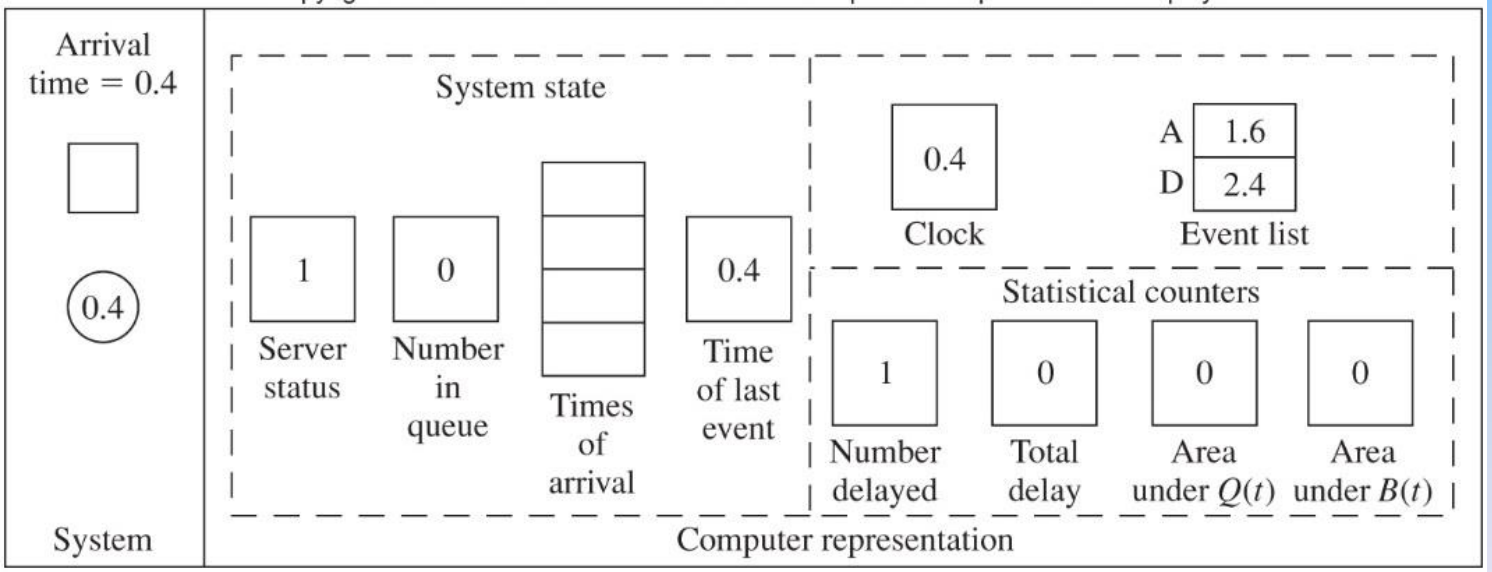
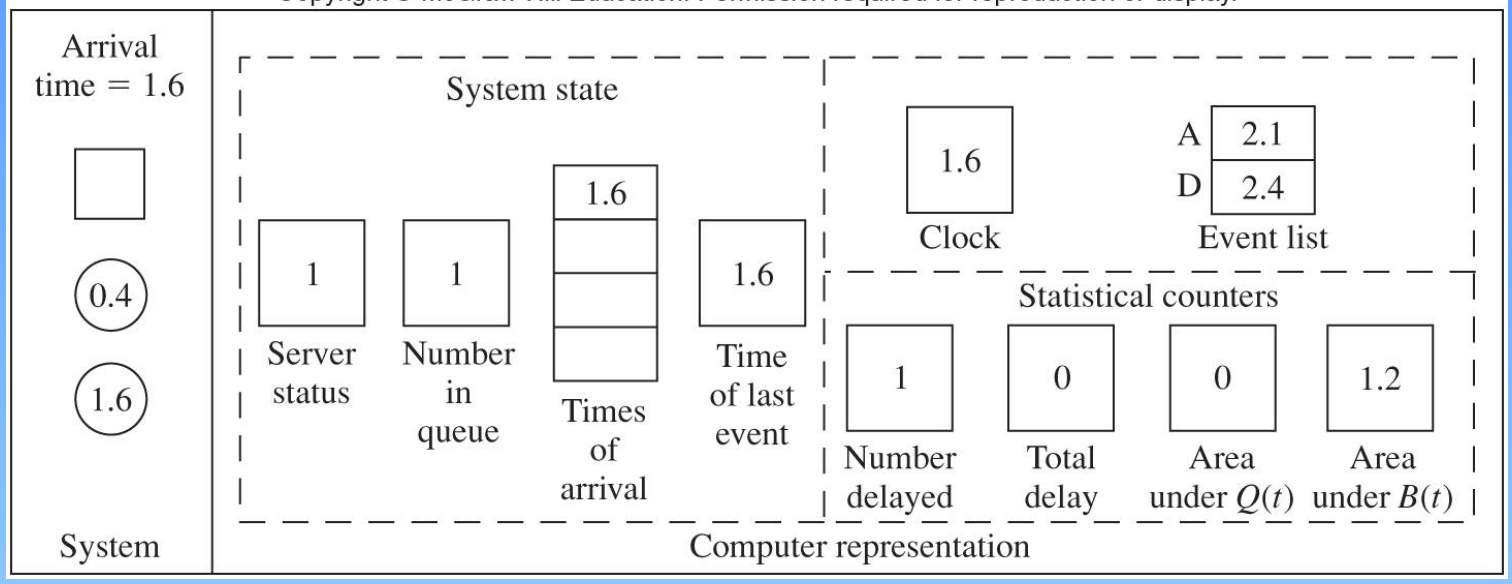


Figure 1.7 (a-b) Snapshots of the system and its computer representation at times t=0 and t=0.4



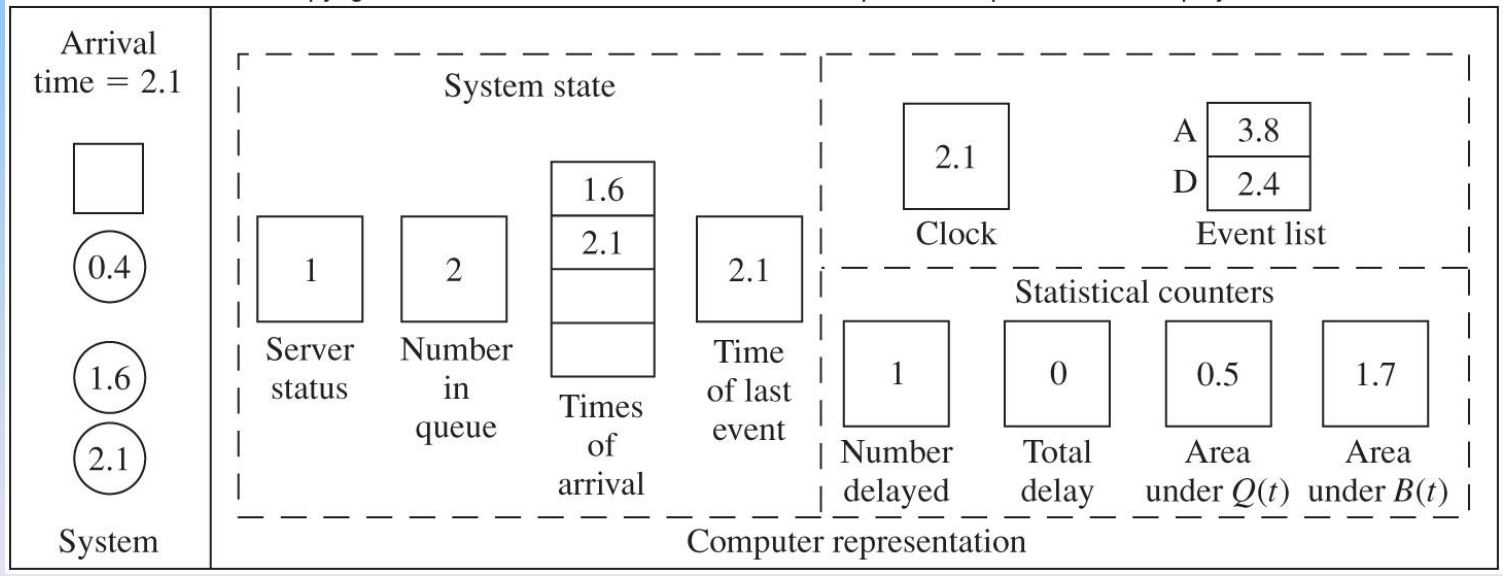
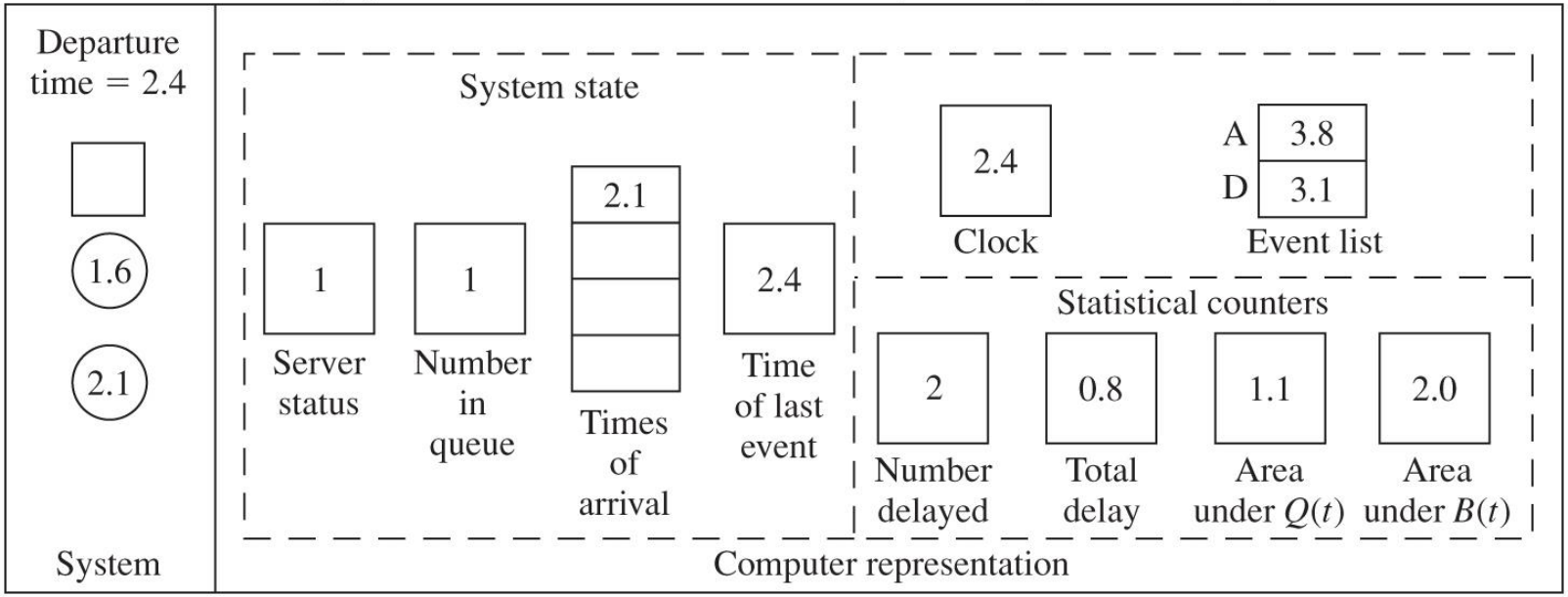


Figure 1.7 (c-d) Snapshots of the system and its computer representation at times t=1.6 and t=2.1



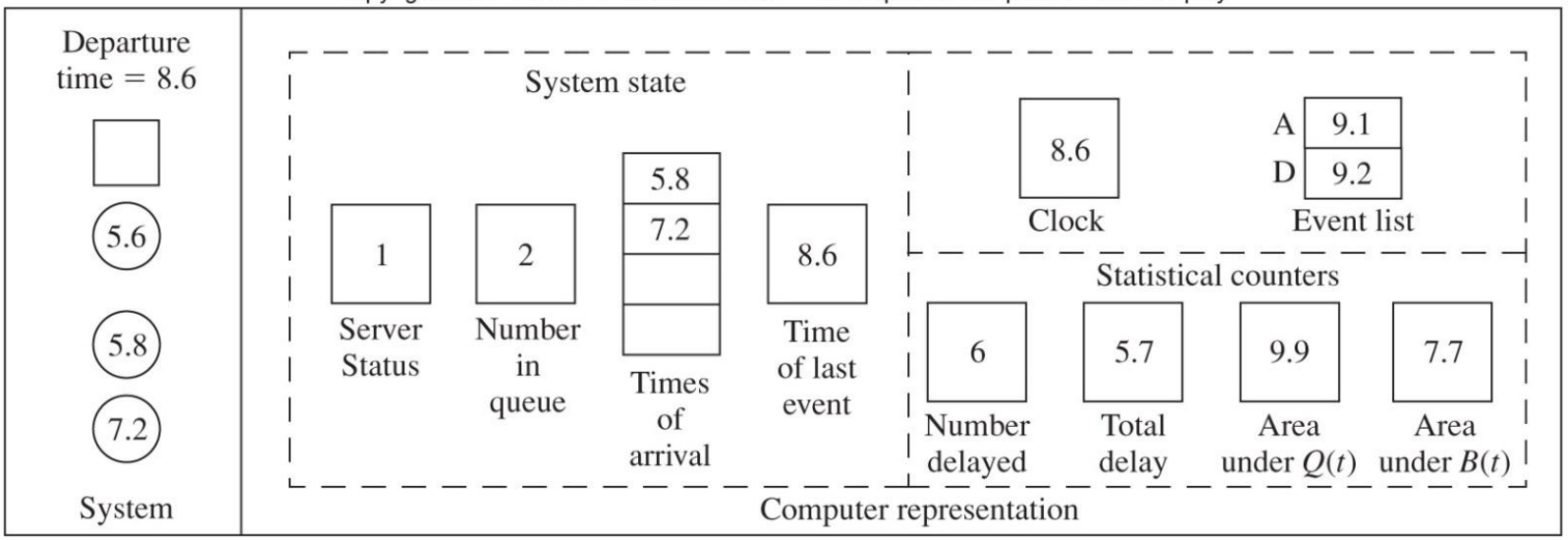
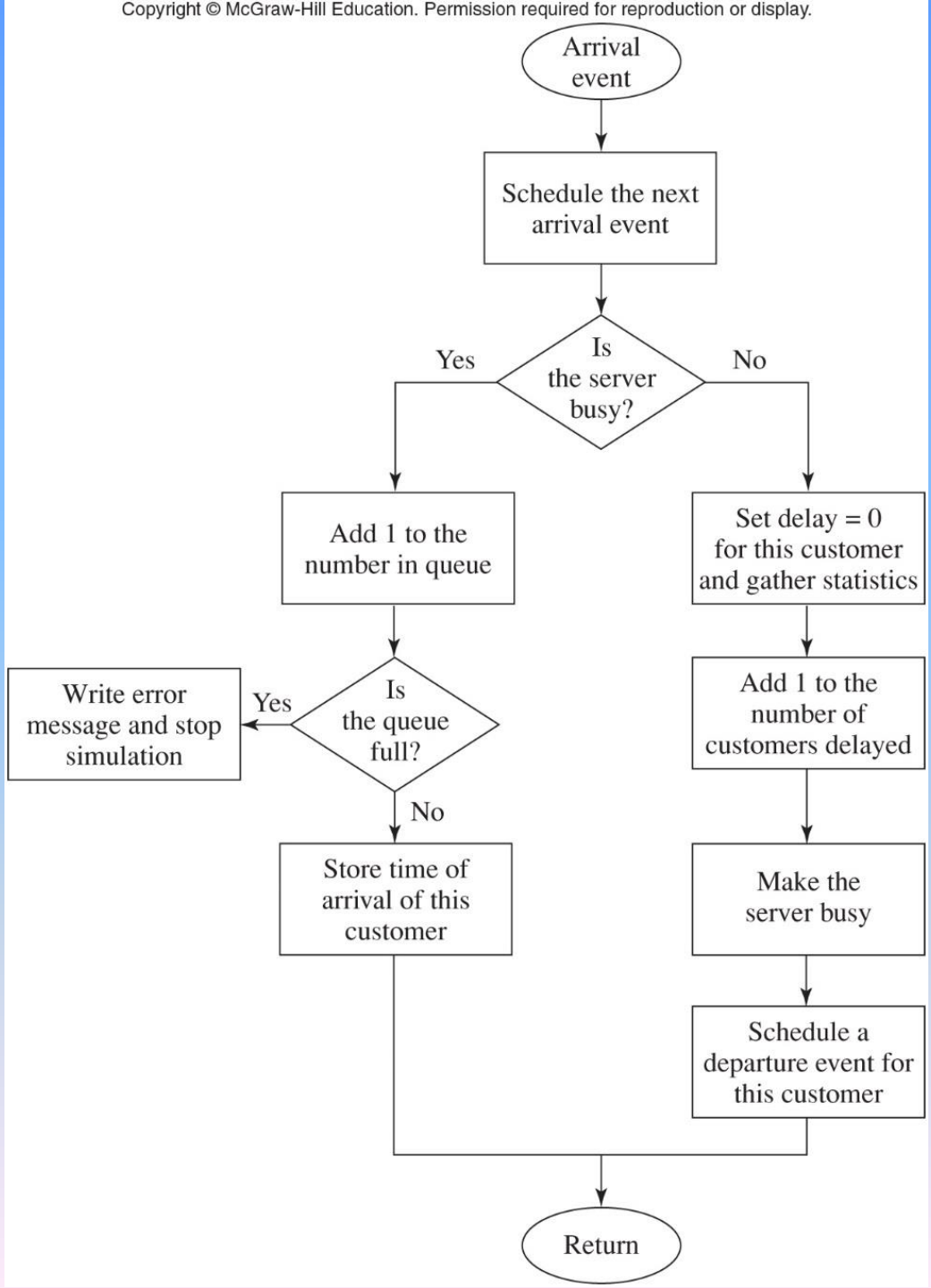


Figure 1.7 (e & n) Snapshots of the system and its computer representation at times t=2.4 and t=8.6

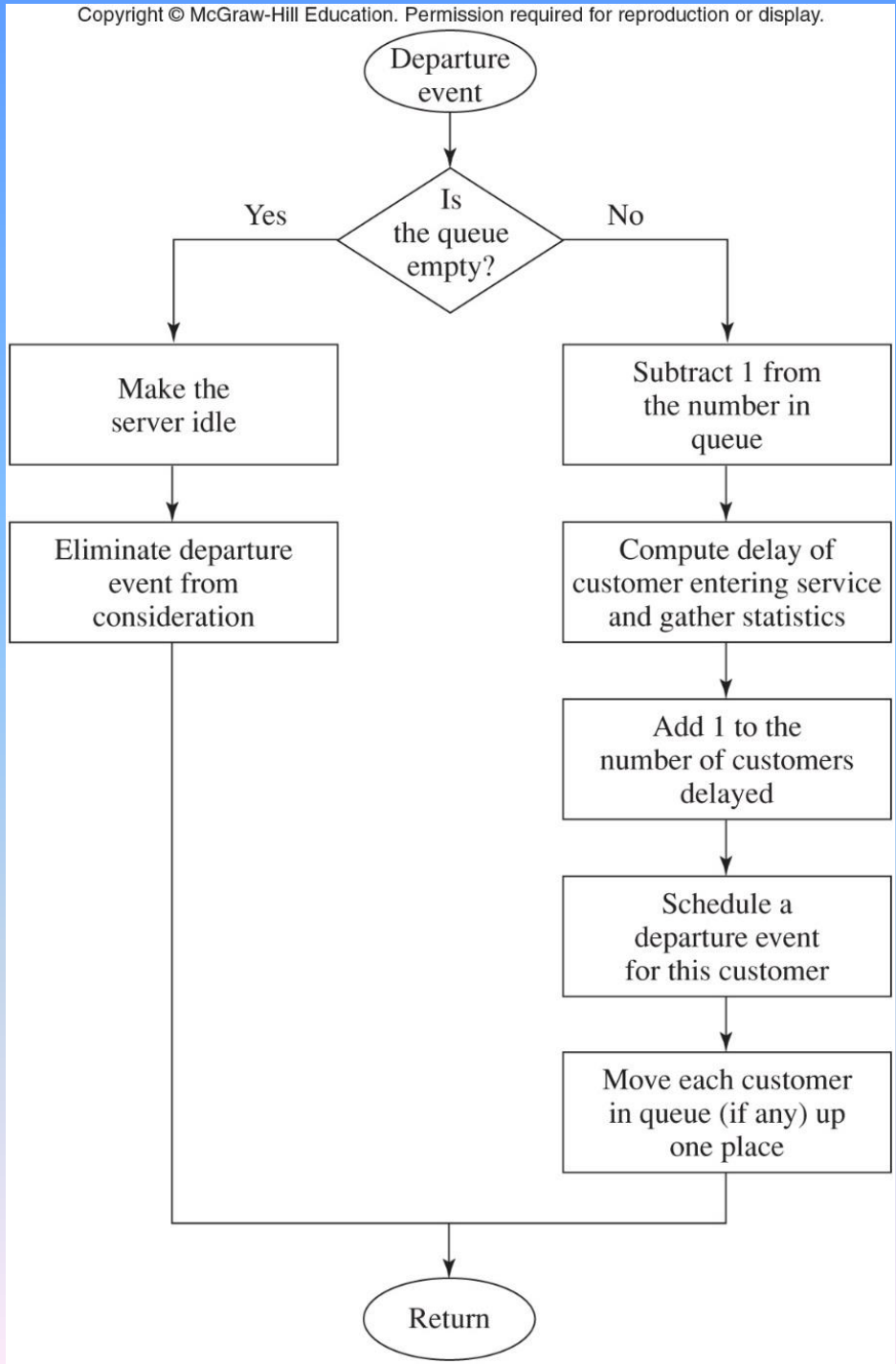
Writing a simulation program

- Our example: C, a general purpose language
  - Know every detail
  - Some simulations (modules) have to be written in GPL
- Program modules
  - Initialization
  - Timing
  - Arrive
  - Depart
  - Report

Flowchart for arrival routine



Flowchart for departure routine



Generate Random Variates from An Exponential Distribution

- $f(x) = \frac{1}{\beta} e^{-x/\beta}$ (x >= 0)
- $F(x) = \int_0^x \frac{1}{\beta} e^{-x/\beta} dx = 1 - e^{-x/\beta}$
- Pick U from (0,1), get $-\beta \ln U$
- $P(-\beta \ln U \leq x) = F(x)$

Simulation Code


```
/* External definitions for single-server queueing system. */

#include <stdio.h>
#include <math.h>
#include "lcgrand.h"  /* Header file for random-number generator. */

#define Q_LIMIT 100  /* Limit on queue length. */
#define BUSY      1  /* Mnemonics for server's being busy */
#define IDLE      0  /* and idle. */

int   next_event_type, num_custs_delayed, num_delays_required, num_events,
      num_in_q, server_status;
float area_num_in_q, area_server_status, mean_interarrival, mean_service,
      sim_time, time_arrival[Q_LIMIT + 1], time_last_event, time_next_event[3],
      total_of_delays;
FILE  *infile, *outfile;

void  initialize(void);
void  timing(void);
void  arrive(void);
void  depart(void);
void  report(void);
void  update_time_avg_stats(void);
float expon(float mean);
```

Figure 1.10 C code for the external definitions, queuing model

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mm1.in",  "r");
    outfile = fopen("mm1.out", "w");

    /* Specify the number of events for the timing function. */

    num_events = 2;

    /* Read input parameters. */

    fscanf(infile, "%f %f %d", &mean_interarrival, &mean_service,
           &num_delays_required);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Single-server queueing system\n\n");
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
    fprintf(outfile, "Number of customers%14d\n\n", num_delays_required);

    /* Initialize the simulation. */

    initialize();
```

Figure 1.11 C code for the main function, queuing model (continues)

```
    /* Run the simulation while more delays are still needed. */

    while (num_custs_delayed < num_delays_required) {

        /* Determine the next event. */

        timing();

        /* Update time-average statistical accumulators. */

        update_time_avg_stats();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case 1:
                arrive();
                break;
            case 2:
                depart();
                break;
        }
    }

    /* Invoke the report generator and end the simulation. */

    report();

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

Figure 1.11 C code for the main function, queuing model (cont'd.)

```
void initialize(void)  /* Initialization function. */
{
    /* Initialize the simulation clock. */

    sim_time = 0.0;

    /* Initialize the state variables. */

    server_status   = IDLE;
    num_in_q        = 0;
    time_last_event = 0.0;

    /* Initialize the statistical counters. */

    num_custs_delayed  = 0;
    total_of_delays    = 0.0;
    area_num_in_q      = 0.0;
    area_server_status = 0.0;

    /* Initialize event list.  Since no customers are present, the departure
       (service completion) event is eliminated from consideration. */

    time_next_event[1] = sim_time + expon(mean_interarrival);
    time_next_event[2] = 1.0e+30;
}
```

Figure 1.12 C code for function initialize, queuing model

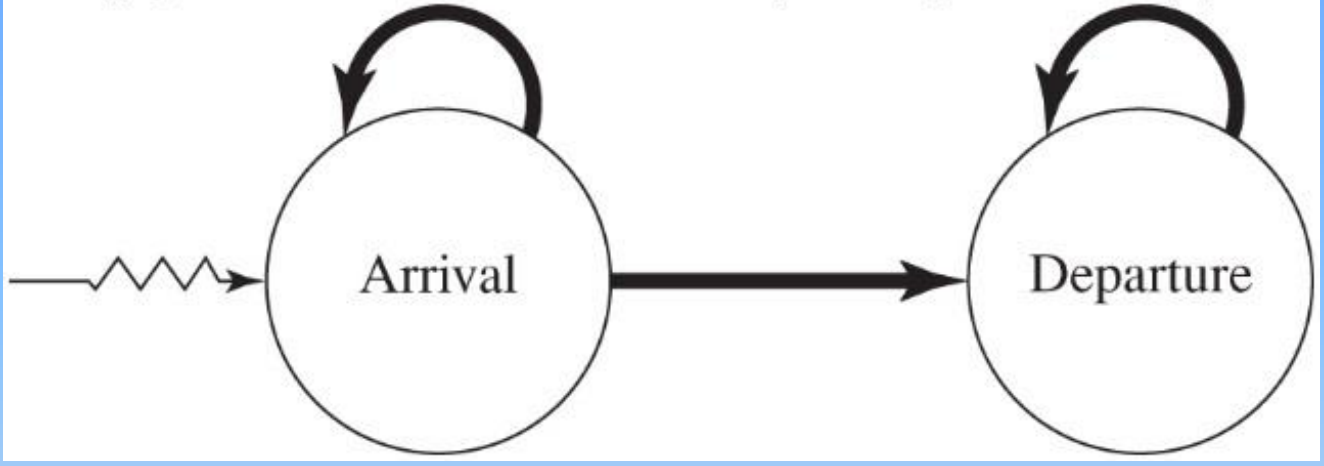
```
void timing(void)  /* Timing function. */
{
    int   i;
    float min_time_next_event = 1.0e+29;

    next_event_type = 0;

    /* Determine the event type of the next event to occur. */

    for (i = 1; i <= num_events; ++i)
        if (time_next_event[i] < min_time_next_event) {
            min_time_next_event = time_next_event[i];
            next_event_type     = i;
        }

    /* Check to see whether the event list is empty. */

    if (next_event_type == 0) {

        /* The event list is empty, so stop the simulation. */

        fprintf(outfile, "\nEvent list empty at time %f", sim_time);
        exit(1);
    }

    /* The event list is not empty, so advance the simulation clock. */

    sim_time = min_time_next_event;
}
```

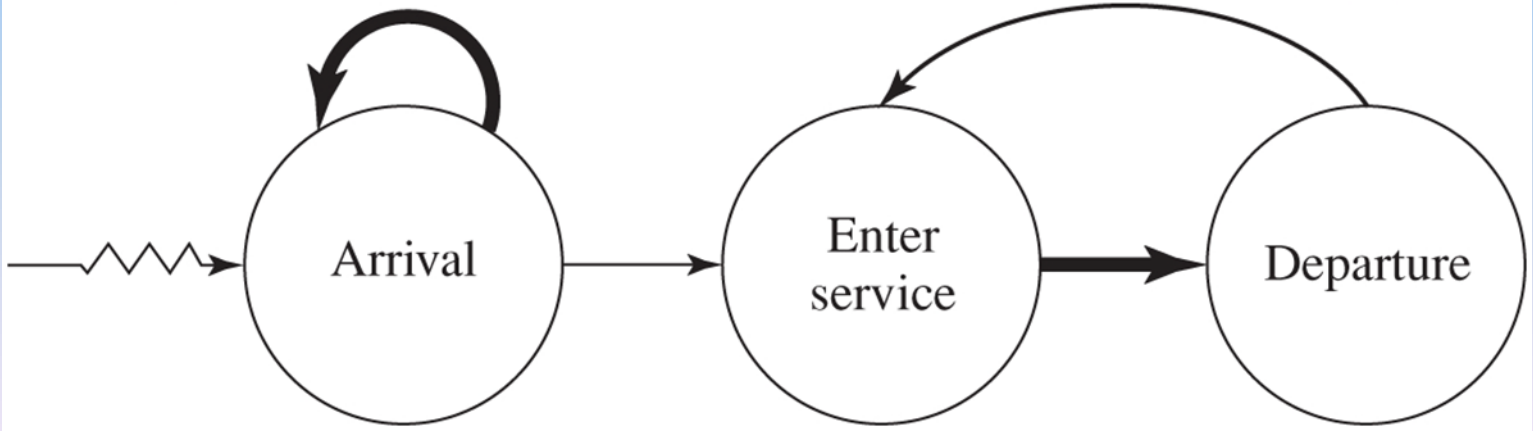
Figure 1.13 C code for function timing, queuing model

Simulation Output Discussion

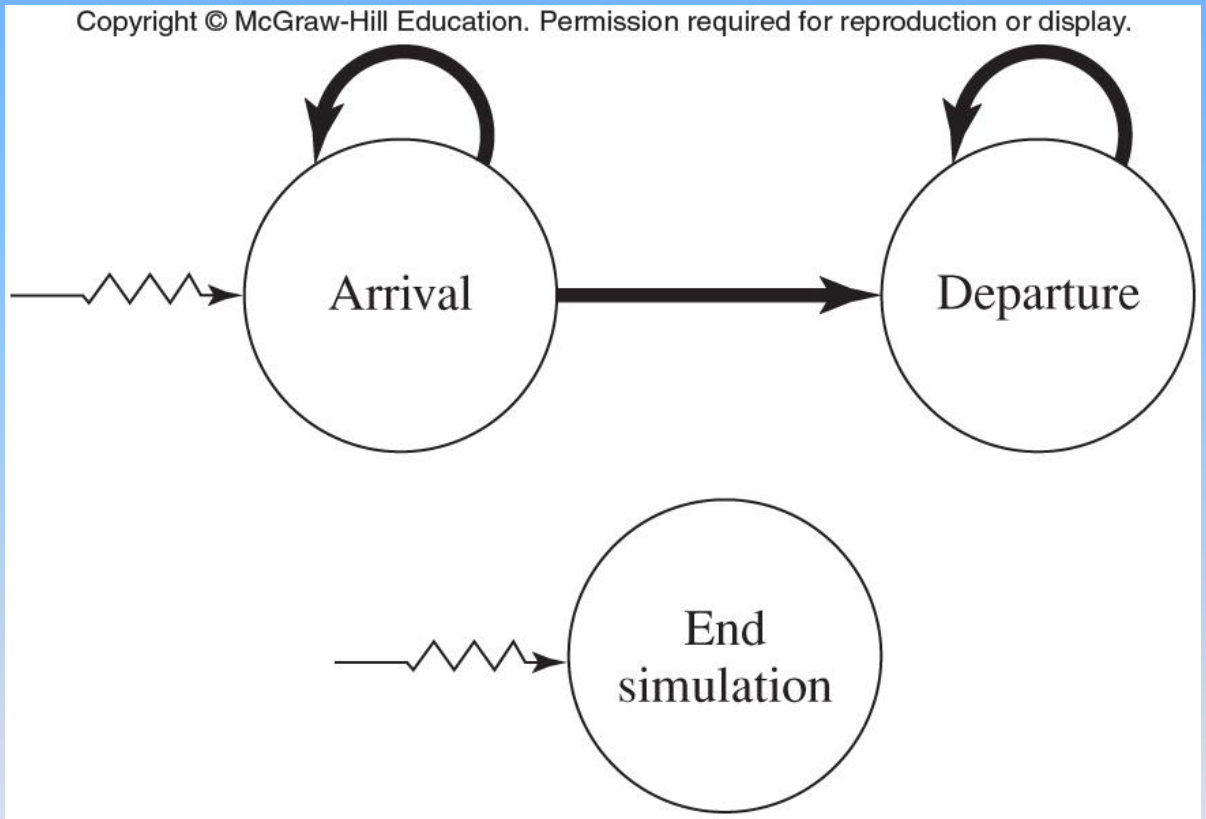
- Numbers will vary each time the simulation is run
  - Not explicit answers but estimates of quantities
- Results are functions of the input parameters, and the way system is initialized
- Might want to study steady state characteristics of the system
  - The characteristics after running for an infinite amount of time (in theory)
  - How long is long enough?
  - More in Ch. 9
- Alternative stopping rules could have been defined
  - E.g. using simulation time. Schedule an end-simulation event.
  - Pay attention to partial service, e.g. when a shop closes at 5PM.

Event Graph





Event Graph



1.5 Simulation of an Inventory System

- Problem: compare various ordering policies for an inventory system
  - Given: initial inventory level, demands, times between demands
  - Costs: setup cost, incremental cost, holding and shortage costs
  - State variables: inventory level, amount of an outstanding order from company to supplier, and time of last event

Event Graphs of An Inventory System

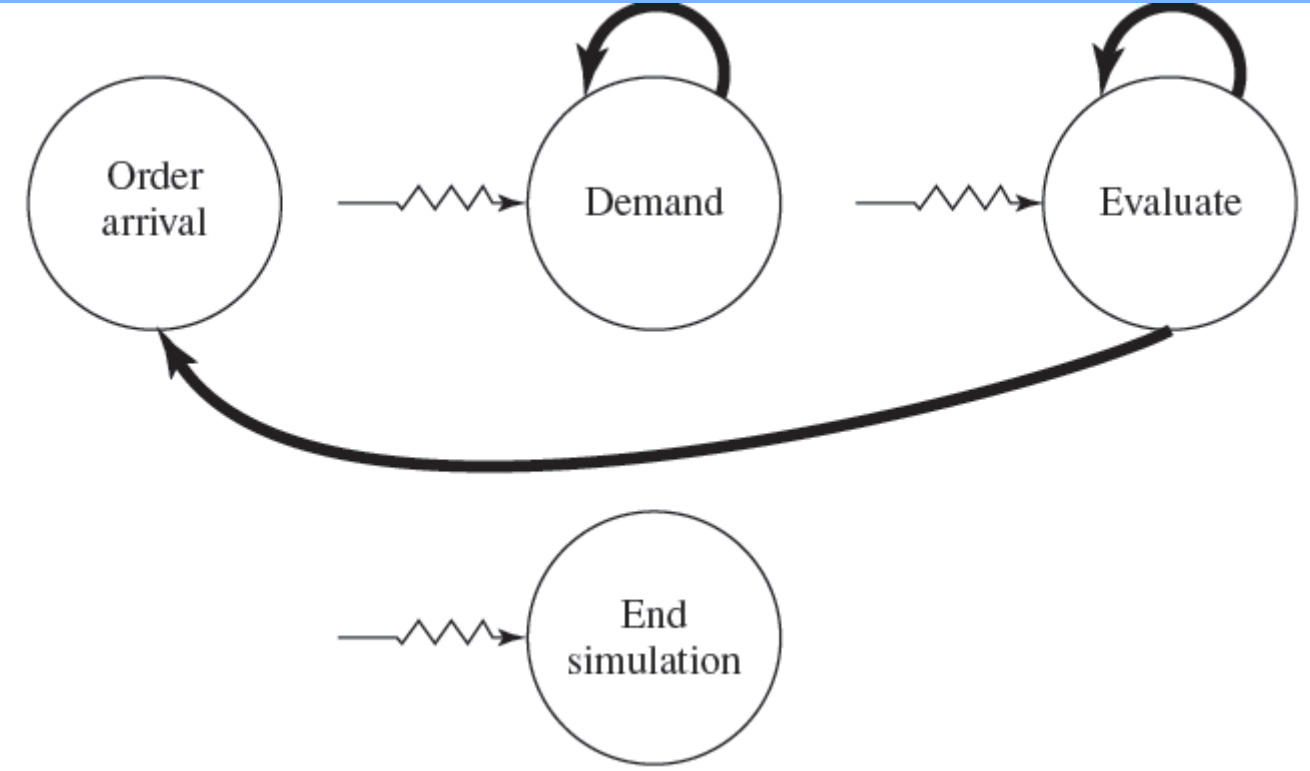
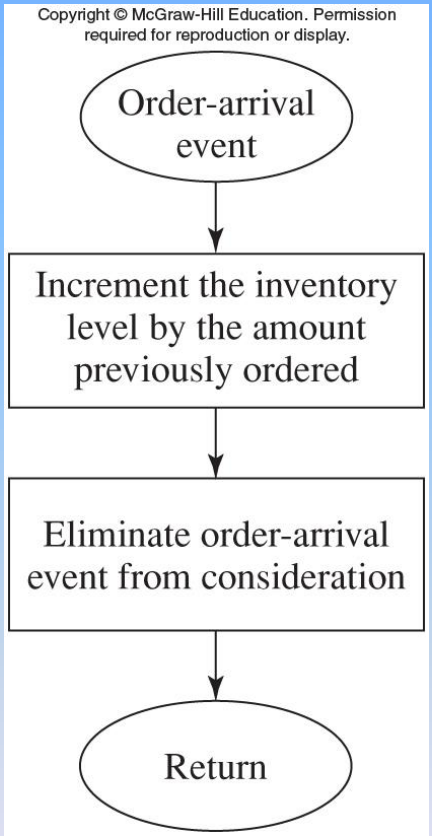
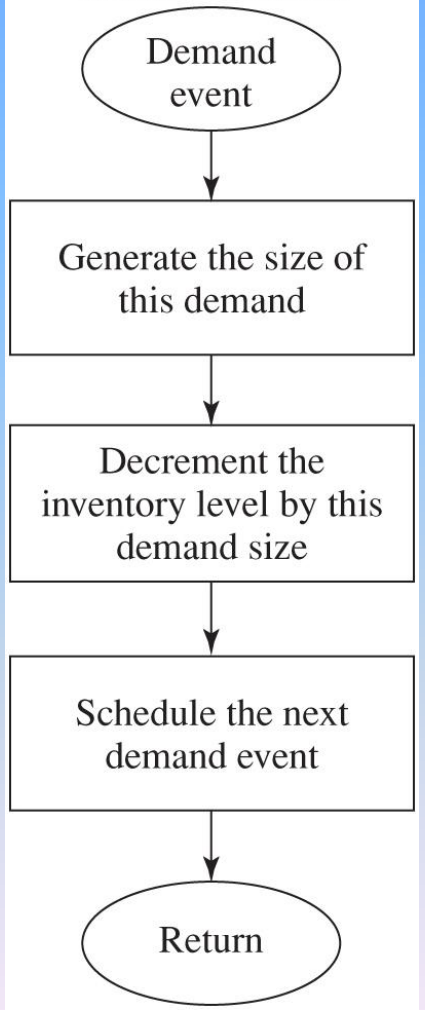
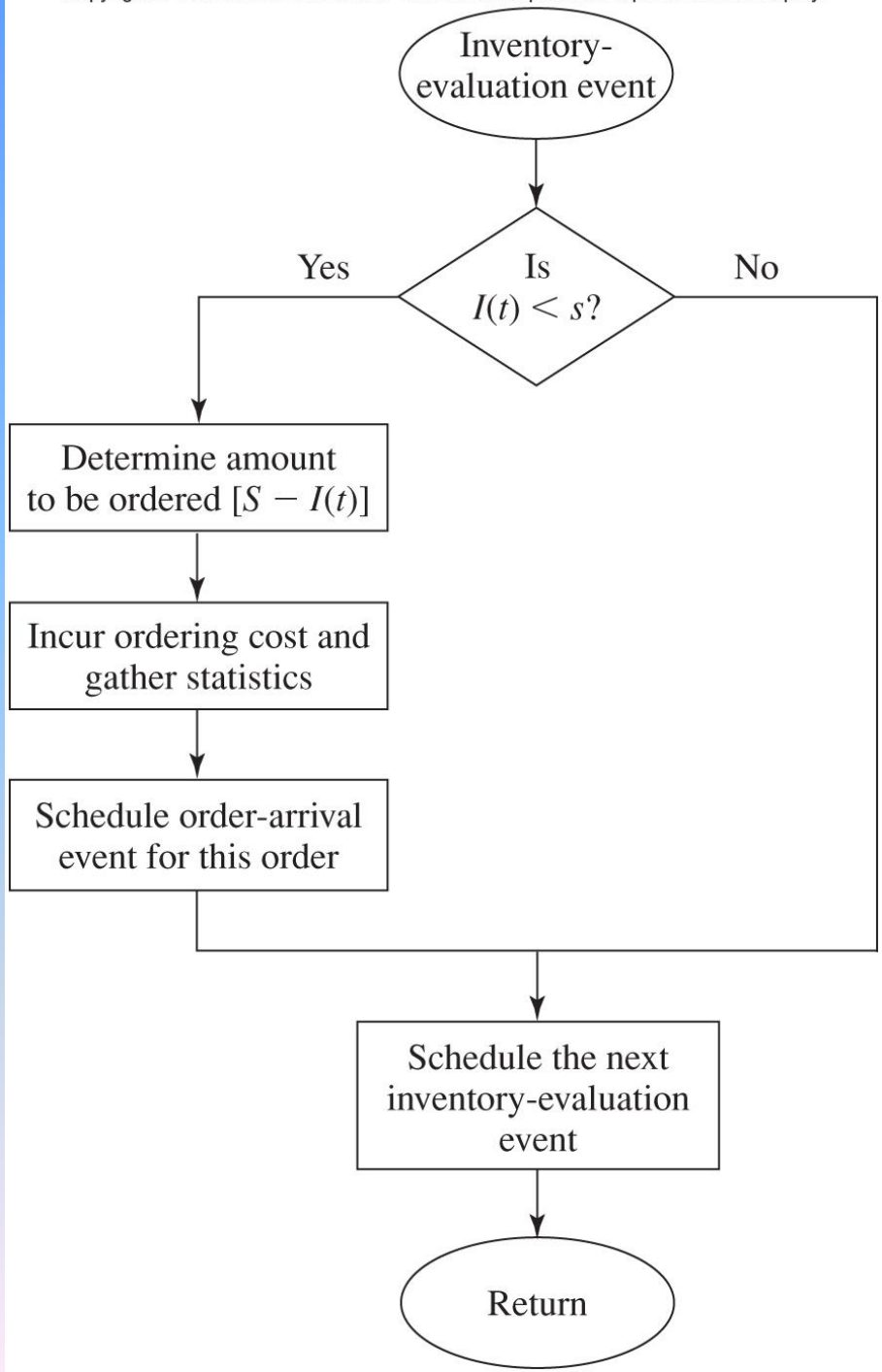


Figure 1.29 Event graph, inventory model

Event Processing Flow Charts







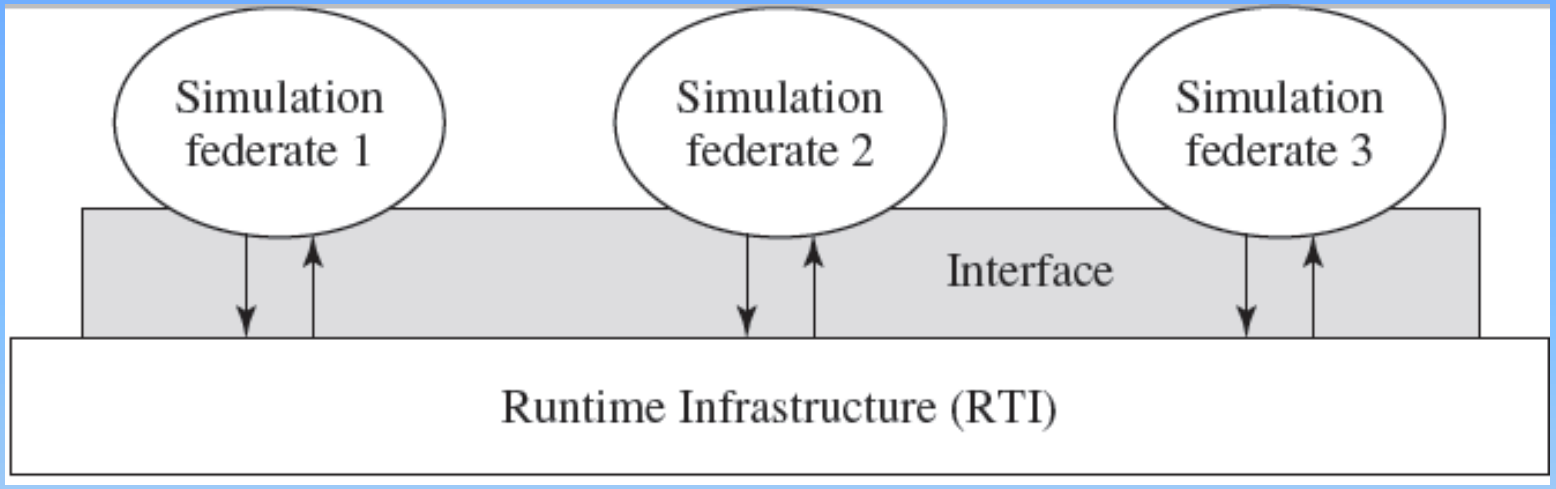
1.6 Parallel/Distributed Simulation and the High Level Architecture

- Parallel-discrete event simulation
  - Execution of the simulation using multiple processors
  - Reduces execution time
  - Done by dividing model into several logical processes (LPs)
  - Critical issue: determining LPs happen in proper sequence

Parallel/Distributed Simulation and the High Level Architecture

- Types of synchronization in parallel simulation
  - Conservative
    - Avoid any violations of local causality constraint
  - Optimistic
    - Time-warp mechanism: best known optimistic approach
    - At the cost of more memory and wasted computation
    - Still has limits
- Distributed simulation
  - High level architecture (HLA) federation

Functional View of An HLA Federation



- HLA
  - The interface specification between RTI and federates
  - The object model template
    - Descriptions of essential shared elements of the federation
  - The rules
    - Key underlying principles

1.7 Steps in a Sound Simulation Study

- Formulate the problem and plan the study
- Collect data and define a model
- Ensure the assumptions are valid
- Construct a computer program and verify
- Make the pilot runs
- Is the programmed model valid?
- Design the experiments

Steps in a Sound Simulation Study

- Make the production runs
- Analyze the output data
- Document, present, and use the results

1.8 Advantages, Disadvantages, and Pitfalls of Simulation

- Most complex, real-world systems cannot be accurately described by an analytical mathematical model
  - Numerical simulation is the only investigation possible
- Simulation allow for the:
  - Evaluation of alternative designs
  - Study of a system with a long time frame

Advantages, Disadvantages, and Pitfalls of Simulation

- Can only estimate
  - Less preferable than analytical study (if feasible)
- Simulation models can be expensive and time-consuming to develop
- Large amounts of data can lead to “overconfidence” in the result
- What are some causes of failure?
  - Lack of well-defined objectives
  - Inappropriate level of detail in the model
- Crucial to involve the right people in the simulation study