Comparison between Simulation technique and Analytical technique:

Simulation Technique	Analytical Technique
Simulation gives specific solutions rather than general solutions. Each execution of simulation tells only whether a particular set of conditions did or did not meet the goal.	Analytical solution gives general solution.
Many simulation runs may be needed to find a maximum.	Mathematical solution is preferable, when the solution being sought is maximizing condition
Various types of complex problems can be solved through simulation.	The range of problems that can be solved mathematically is limited. Mathematical techniques require that the model be expressed in some particular format.

Simulation Technique	Analytical Technique
It needs a little abstraction for no abstraction to apply simulation methods.	Sometimes, the degree of abstraction required to apply analytical method is too severe. It reduces the degree of accuracy.
The ideal way of using simulation is an execution of mathematical solutions that might have been obtained at the cost of too much simplification.	Sometimes, it needs too much simplification to form a model for analytical solution.
Simulation easily removes many limitations on a system, such as physical stop, finite time delays, nonlinear forces, etc.	These limitations make solvable mathematical model in solution.
Simulation will provide a quicker or more convenient way of deriving results.	Many analytical results occur in the form of complex series or integrals that require extension evolution.

Drawbacks of Simulation Method

- It gives specific solution rather than general solution.

For example, in the study of automobiles wheel, an analytical solution gives all the condition that can cause oscillation. But each execution of a simulation only tells whether a particular set of condition did or did not cause oscillation. To try to find all such condition required that the simulation be repeated under many different condition.

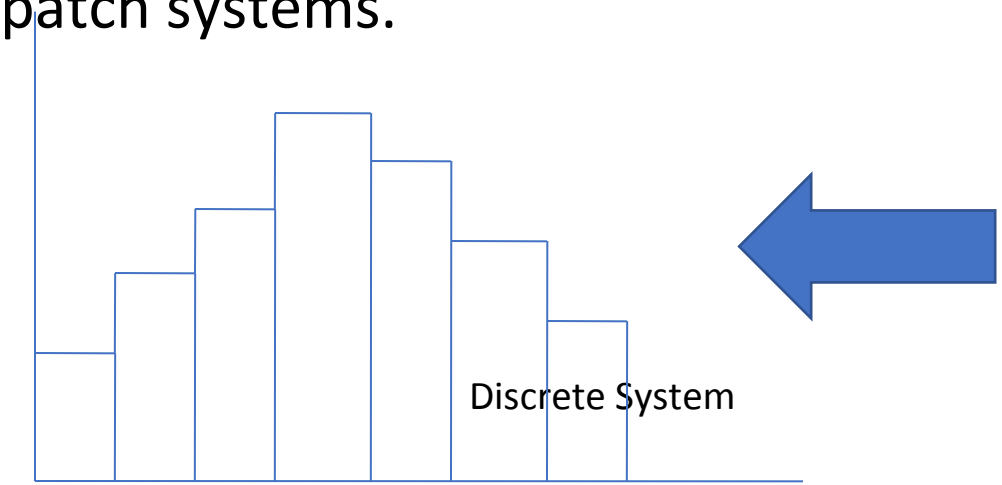
- The step by step nature of the simulation technique means that the amount of computation increase very rapidly as the amount of detail increases. Coupled with the need to make many runs, the simulation model result in extensive amount of computing.
- Many simulation runs may be needed to find a maximum and yet leave undecided the question of whether it is a local or global maximum.

Drawbacks of Analytical Technique

- The range of problem that can be solve mathematically is limited.
- Mathematical technique requires that the model be expressed in some particular format. For example, in the form of linear algebraic equation and continuous linear differential equation.
- There are many simple limitation on a system such as physical stock, finite time delays or non-linear forces which makes a soluble mathematical model insoluble. But simulation removes this limitation.

DISCRETE SYSTEM

- A Discrete System is a System with a countable number of states that may be described in precise mathematical model.
- Discrete-event models deal with events and specific time intervals.
- Examples of discrete events include computer-performance evaluation and inventory dispatch systems.



DISCRETE SIMULATION/DISCRETE-EVENT SIMULATION

- Discrete Simulation technique generally applied on discrete model. Discrete-event simulation concerns the modeling of a system as it evolves over time by representation in which the state variables change instantaneously at separate point in time.
- In more mathematical terms, the system can change at only a countable number of points in time. These points in time are the once at which an event occurs, where an event is defined as an instantaneous occurrence that may change the state of the system.
- Although discrete-event system simulation could conceptually be done by hand calculations, the amount of data that must be stored and manipulated for most real world systems dictates that discrete-event simulation be done on a digital computer.

Discrete Event Simulation – Key Features

Discrete event simulation is generally carried out by a software designed in high level programming languages such as Pascal, C++, or any specialized simulation language. Following are the five key features −

- **Entities** − These are the representation of real elements like the parts of machines.
- **Relationships** − It means to link entities together.
- **Simulation Executive** − It is responsible for controlling the advance time and executing discrete events.
- **Random Number Generator** − It helps to simulate different data coming into the simulation model.
- **Results & Statistics** − It validates the model and provides its performance measures.

DISCRETE SYSTEM TERMINOLOGIES

- **State :** A variable characterizing an attribute in the system such as level of stock in inventory or number of jobs waiting for processing.
- **Event:** An occurrence at a point in time, which may change the state of the system, such as arrival of a customer or start of work on a job.
- **Entity :** An object that passes through the system, such as cars in an intersection or orders in a factory. Often an event (e.g., arrival) is associated with an entity (e.g., customer).
- **Queue :** A queue is not only a physical queue of people, it can also be a task list, a buffer of finished goods waiting for transportation or any place where entities are waiting for something to happen for any reason.

DISCRETE SYSTEM TERMINOLOGIES

- **Creating :**Creating is causing an arrival of a new entity to the system at some point in time.
- **Scheduling** ***:****Scheduling is the act of assigning a* new future event to an existing entity.
- **Random Variable** : A random variable is a quantity that is uncertain, such as inter-arrival time between two incoming flights or number of defective parts in a shipment.
- **Random Variate :**A random variate is an artificially generated random variable.
- **Distribution:** A distribution is the mathematical law, which governs the probabilistic features of a random variable.

CONTINUOUS OR DISCRETE?

1) Experimental data is discrete, but large amount of data is almost continuous.
2) More mathematical tools are available for continuous model, computer can only analyze Discrete Model.
3) Continuous Model can be discretized , and discrete model can be approximated by Continuous One.
4) Continuous model needs calculus, discrete model needs only algebra.

Numerical Computation Technique-Discrete Event Simulation

Numerical Computing : It is an approach for solving complex mathematical problems using only simple arithmetic operations.

To illustrate the general computational technique of simulation with discrete models, consider the following example:

- Consider an example, a clerk begins his days of works with a file of document to be processed. The time taken to process them varies.
- He works through the file beginning each document, as soon as he finishes the previous one except that he takes that 5min break if at that time, he finishes a document is an hour or more, since he begins work or he left a break.

DISCRETE EVENT SIMULATION-EXAMPLE

- We assume that the time to process the documents are given. We will keep a count of no. of documents left for processing.
- This count will be initially set to the no. of document at the beginning of the day and we assume that no. of document arrive during the day.
- The count will be decremented for each completed job and the work will stop when count goes to zero.
- The first column numbers the documents, the second column gives the time the clerk begins to work for a document denoted by (t_b), the third column is the time required to work on the document called as the work time(t_w), the fourth column gives the time at which the document is finished processing in time (t_f), the fifth column contains the cumulative time since work started or since the last break, measured at the time each job is completed. This is denoted by (t_c). There is a sixth column which contains a flag, denoted by F, that takes the value 1 if the clerk should take a break after the ith document and the value 0 if he should not.

DISCRETE EVENT SIMULATION-EXAMPLE

- The clerk works until there are no more documents, or the time he finishes a document goes beyond some time limit.
- The computation proceeds row by row, and from left to right. The first row shows that starts on the first document at time zero. The processing time is 46 minutes, so the job is finished at 45, with a cumulative time (in this case, since the start of work) of 45. This is not long enough for a break, so the flag is set to 0. The count, which was initialized to 57 jobs, is dropped to 56.
- Because the flag is zero and the count is not zero, the second document is begun at 45. It needs 16 minutes for processing, which leads to a cumulative time of 61, so the flag is set to 1 to indicate that a break should be taken. Because of the five minute break, the third document starts at 66. The computational-continues in this manner until either N, the count of documents to be processed, drops to zero, or the finish time, tf, reaches some limit representing the end of the work period.

DISCRETE EVENT SIMULATION-EXAMPLE

Document Number I	Start Time t_b	Work Time t_w	Finish Time t_f	Cumulative Time t_c	Break Flag F	Number of jobs N
1	0	45	45	45	0	57
2	45	16	61	61	1	56
3	66	5	71	5	0	55
4	71	29	100	34	0	54
5	100	33	133	67	1	53
6	138	25	163	25	0	52
7	163	21	184	46	0	51

Simulation and Queuing Problems

- A major application of simulation has been in the analysis of waiting line, or queuing systems.
- Queuing Theory is the mathematical study of waiting lines which are the most frequently encountered problems in everyday life.
- Since the time spent by people and things waiting in line is a valuable resource, the reduction of waiting time is an important aspect of operations management.
- Waiting time has also become more important because of the increased emphasis on quality. Customers equate quality service with quick service and providing quick service has become an important aspect of quality service.

Queuing Problems

- For queuing systems, it is usually not possible to develop analytical formulas, and simulation is often the only means of analysis.
- Simulation can hence be used to investigate problems that are common in any situation involving customers, items or orders arriving at a given point, and being processed in a specified order.
- For example:Customers arrive in a bank and form a single queue, which feeds a number of service desks. The arrival rate of the customers will determine the number of service desks to have open at any specific point in time.
- Queuing Model is a suitable model used to represent service oriented problems, where customer arrive randomly to receive some service, the service time being also a random variable.
- Queuing models provide the analyst with a powerful tool for designing and evaluating the performance of a queuing system.

Components of a Queuing System

A queue system can be divided into four components

- Arrivals: Concerned with how items (people, cars etc.) arrive in the system.
- Queue or waiting line: Concerned with what happens between the arrival of an item requiring service and the time when service is carried out.
- Service: Concerned with the time taken to serve a customer.
- Outlet or departure: The exit from the system.

A queuing problem involves striking a balance between the cost of making reductions in service time and the benefits gained from such a reduction.

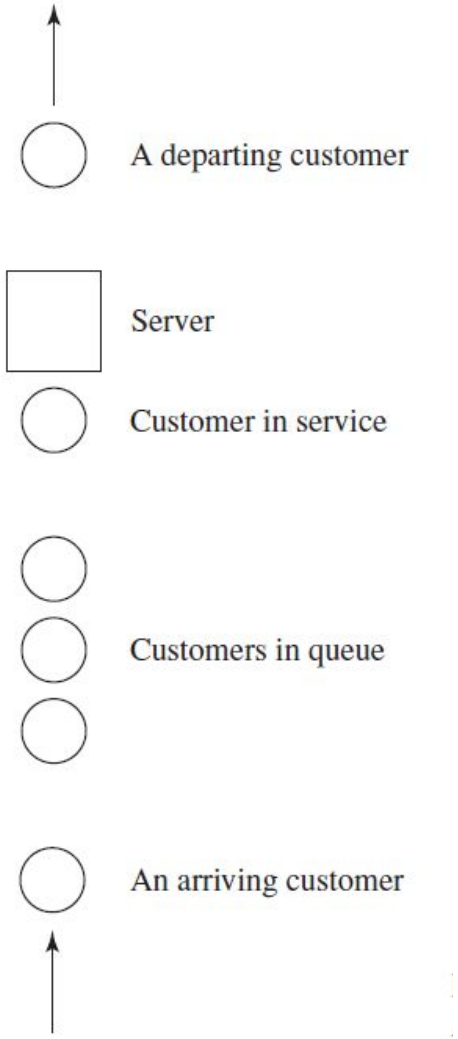
Structures of Queuing System

- There are a number of structures of queuing systems in practice.
- We are considering a *single queue – single service point.*

• Single queue – single service point

• Queue discipline is first come – first served.

SINGLE-SERVER QUEUING SYSTEM



Notations in Single-Server Queueing Systems

- ti =time of arrival of the ith customer
- Ai = interarrival time between (i - 1)st and ith arrivals of customers
- Si = time that server actually spends serving ith customer (exclusive of customer's delay in queue)
- Di = delay in queue of ith customer
- ci =ti + Di + Si = time that ith customer completes service and departs
- ei = time of occurrence of ith event of any type

SINGLE-SERVER QUEUING SYSTEM

- An ***Event*** is defined as an instantaneous occurrence that may change the state of the system. how to simulate a single-server queueing system such as a one-operator barbershop.
- Consider a single-server queueing system for which the interarrival times $A1, A2, \ldots$ are *independent and identically distributed* (IID) random variables.
- service times S1, S2, . . . of the successive customers are IID random variables
- "Identically distributed" means that the interarrival times have the same probability distribution.
- Queue discipline: FIFO
- Stopping Condition: We wish to simulate this system until a fixed number (n) of customers have completed their delays in queue; i.e., the simulation will stop when the nth customer enters service.

SINGLE-SERVER QUEUING SYSTEM

- The simulation will begin in the "empty-and-idle" state; i.e., no customers are present and the server is idle.
- To measure the performance of this system, we will look at estimates of **three** quantities.
- First, we will estimate the **expected average delay in queue of the *n* customers** completing their delays during the simulation; we denote this quantity by $d(n)$.
- Here, the average delay on a given run of the simulation is properly regarded as a random variable itself. What we want to estimate, $d(n)$, is the *expected value* of this random variable.From a single run of the simulation resulting in customer delays $D1, D2, \ldots, Dn$, an obvious estimator of $d(n)$ is

$$\hat{d}(n) = \frac{\sum_{i=1}^{n} D_i}{n}$$

  which is just the average of the *n Di*'s that were observed in the simulation.

SINGLE-SERVER QUEUING SYSTEM

- Second, **the expected average number of customers in the queue** (but not being served)
- While an estimate of $d(n)$ gives information about system performance from the customers' point of view, the management of such a system may want different information; indeed. One such measure for our simple model here is the expected average number of customers in the queue (but not being served), denoted by $\hat{q}(n)$, where the n is necessary in the notation to indicate that this average is taken over the time period needed to observe the n delays defining our stopping rule.

SINGLE-SERVER QUEUING SYSTEM

- If we let *Ti* be the *total* time during the simulation that the queue is of length *i*, then

  $T(n) = T0 + T1 + T2 + \cdots$

$$\hat{q}(n) = \frac{\sum_{i=0}^{\infty} iT_i}{T(n)}$$

- In other words, the summation in the numerator is just the *area under the Q(t) curve between the beginning and the end of the simulation*. Remembering that "area under a curve" is an integral, we can thus write

$$\sum_{i=0}^{\infty} iT_i = \int_0^{T(n)} Q(t)\, dt$$

and the estimator of $q(n)$ can then be expressed as

$$\hat{q}(n) = \frac{\int_0^{T(n)} Q(t)\, dt}{T(n)}$$

Third: **Expected Server Utilization**

SINGLE-SERVER QUEUING SYSTEM

Given a system, we need to illustrates a possible time path, or *realization*, of $Q(t)$ for this system where $n = 6$.

- Arrivals occur at times 0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.
- Departures (service completions) occur at times 2.4, 3.1,3.3, 4.9, and 8.6, and the simulation ends at time $T(6) = 8.6$.
- $Q(t)$ does not count the customer in service (if any).

SINGLE-SERVER QUEUING SYSTEM

- So between times 0.4 and 1.6 there is one customer in the system being served, even though the queue is empty [$Q(t) = 0$], the same is true between times 3.1 and 3.3, between times 3.8 and 4.0, and between times 4.9 and 5.6.
- Between times 3.3 and 3.8, however, the system is empty of customers and the server is idle, as is obviously the case between times 0 and 0.4. To compute $q\hat{}(n)$, we must first compute the Ti's, which can be read as the (sometimes separated) intervals over which $Q(t)$ is equal to 0, 1, 2, and so on:

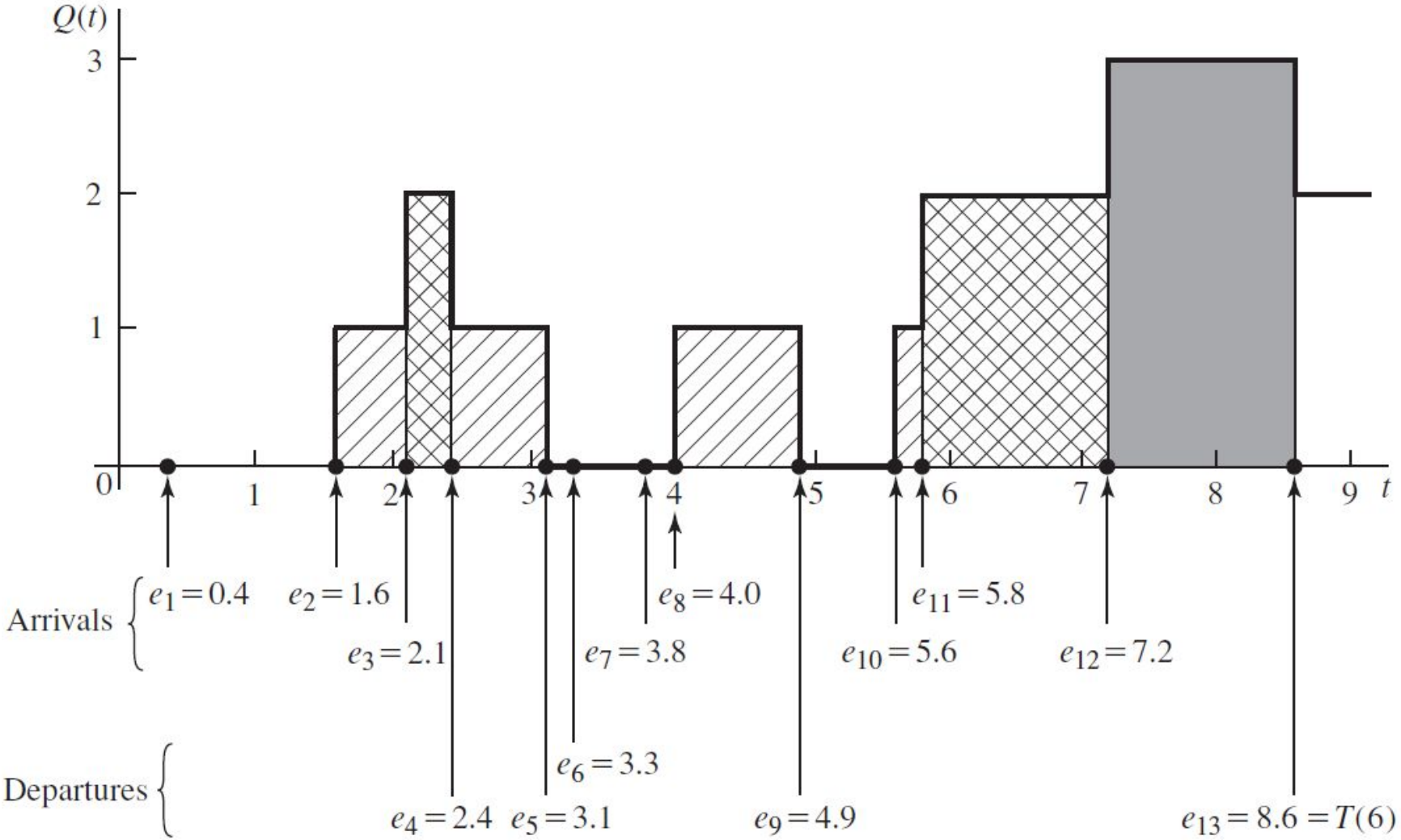
$$T_0 = (1.6 - 0.0) + (4.0 - 3.1) + (5.6 - 4.9) = 3.2$$
$$T_1 = (2.1 - 1.6) + (3.1 - 2.4) + (4.9 - 4.0) + (5.8 - 5.6) = 2.3$$
$$T_2 = (2.4 - 2.1) + (7.2 - 5.8) = 1.7$$
$$T_3 = (8.6 - 7.2) = 1.4$$

SINGLE-SERVER QUEUING SYSTEM

$$\sum_{i=0}^{\infty} iT_i = (0 \times 3.2) + (1 \times 2.3) + (2 \times 1.7) + (3 \times 1.4) = 9.9$$

So our estimate of the time-average number in queue from this particular simulation run is $q\hat{}(6) = 9.9/8.6 = 1.15$.

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.
Departures : 2.4, 3.1,3.3, 4.9, and 8.6



SINGLE-SERVER QUEUING SYSTEM

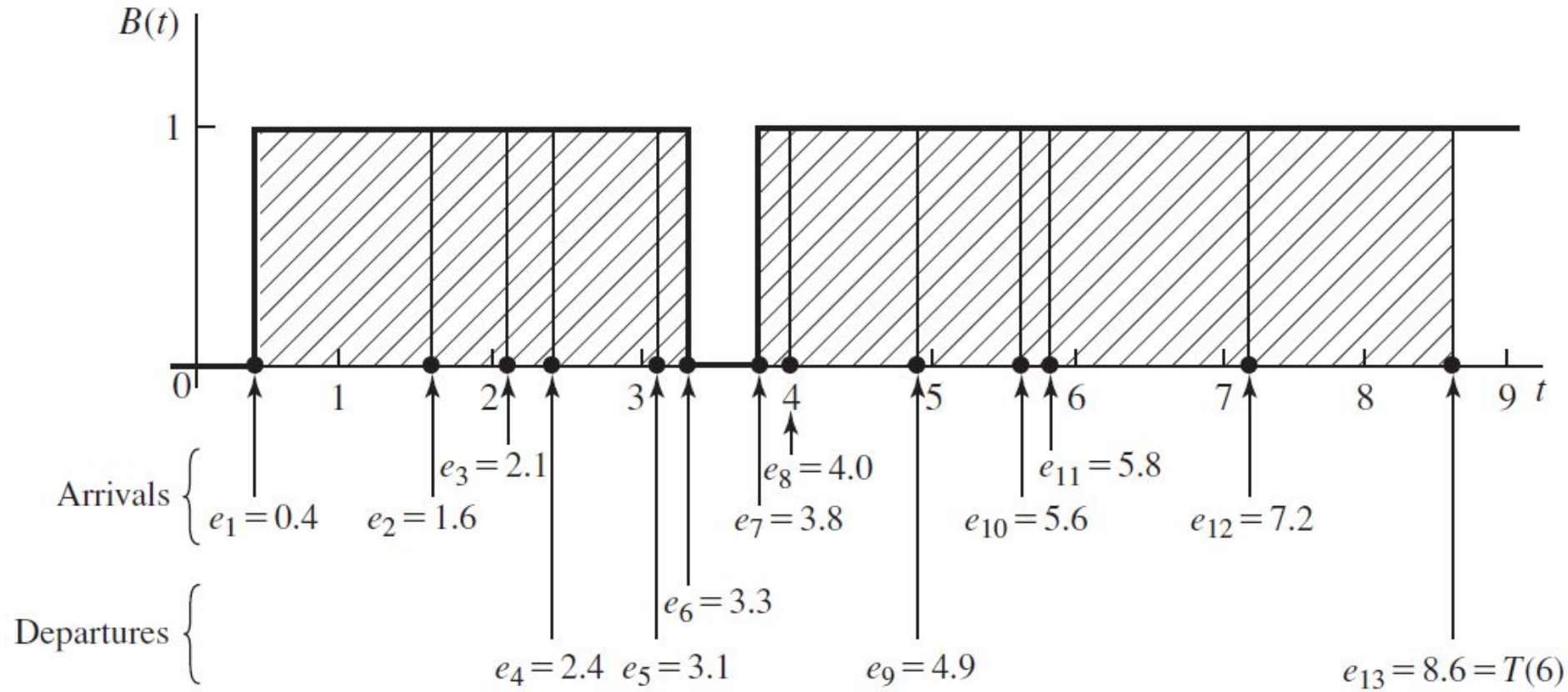
- The third and final output measure of performance for this system is a measure of how busy the server is. The **expected *utilization* of the server** is the expected proportion of time during the simulation [from time 0 to time $T(n)$] that the server is busy (i.e., not idle), and is thus a number between 0 and 1; denote it by $u(n)$.
- From a single simulation, then, our estimate of $u(n)$ is $\hat{u}(n)$ = the *observed* proportion of time during the simulation that the server is busy.
- However, it is easier to look at this quantity as a continuous-time average, similar to the average queue length, by defining the "busy function"

$$B(t) = \begin{cases} 1 & \text{if the server is busy at time } t \\ 0 & \text{if the server is idle at time } t \end{cases}$$

and so $\hat{u}(n)$ could be expressed as the proportion of time that $B(t)$ is equal to 1.

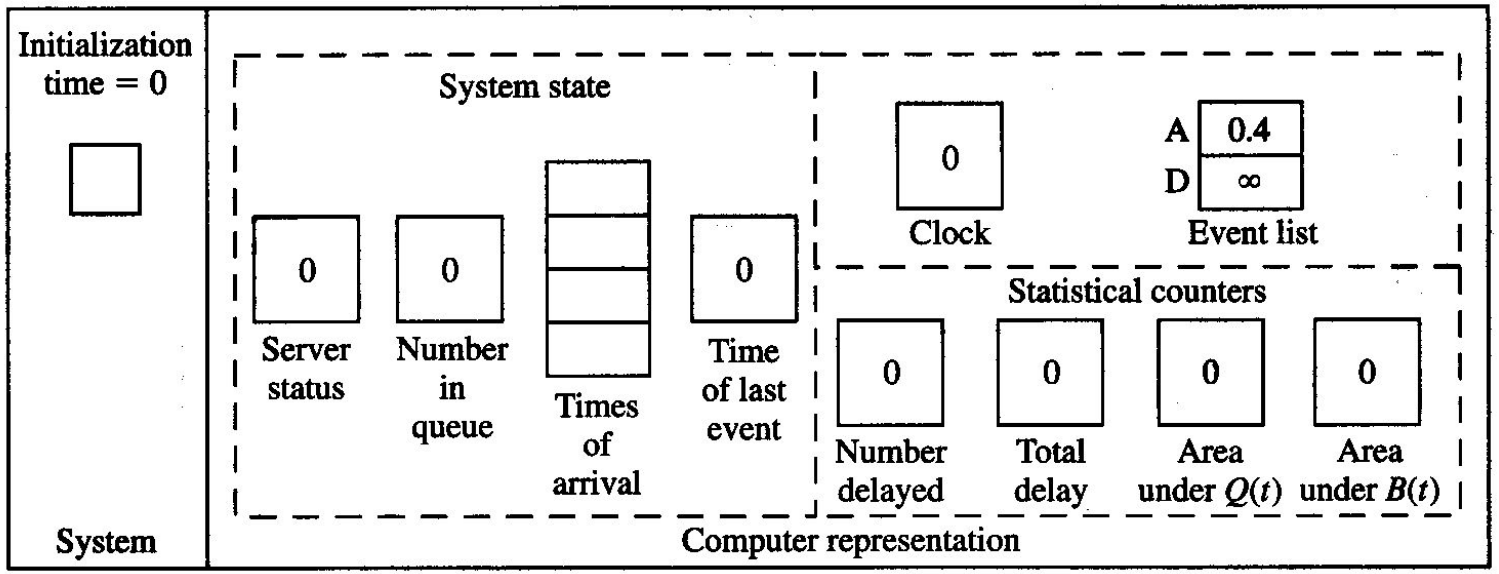
- $B(t)$ for the same simulation realization

$$\hat{u}(n) = \frac{(3.3 - 0.4) + (8.6 - 3.8)}{8.6} = \frac{7.7}{8.6} = 0.90$$


B(t)
1
0
1
2
3
4
5
6
7
8
9
t
Arrivals
Departures
$e_1 = 0.4$
$e_2 = 1.6$
$e_3 = 2.1$
$e_4 = 2.4$
$e_5 = 3.1$
$e_6 = 3.3$
$e_7 = 3.8$
$e_8 = 4.0$
$e_9 = 4.9$
$e_{10} = 5.6$
$e_{11} = 5.8$
$e_{12} = 7.2$
$e_{13} = 8.6 = T(6)$

Intuitive Explanation

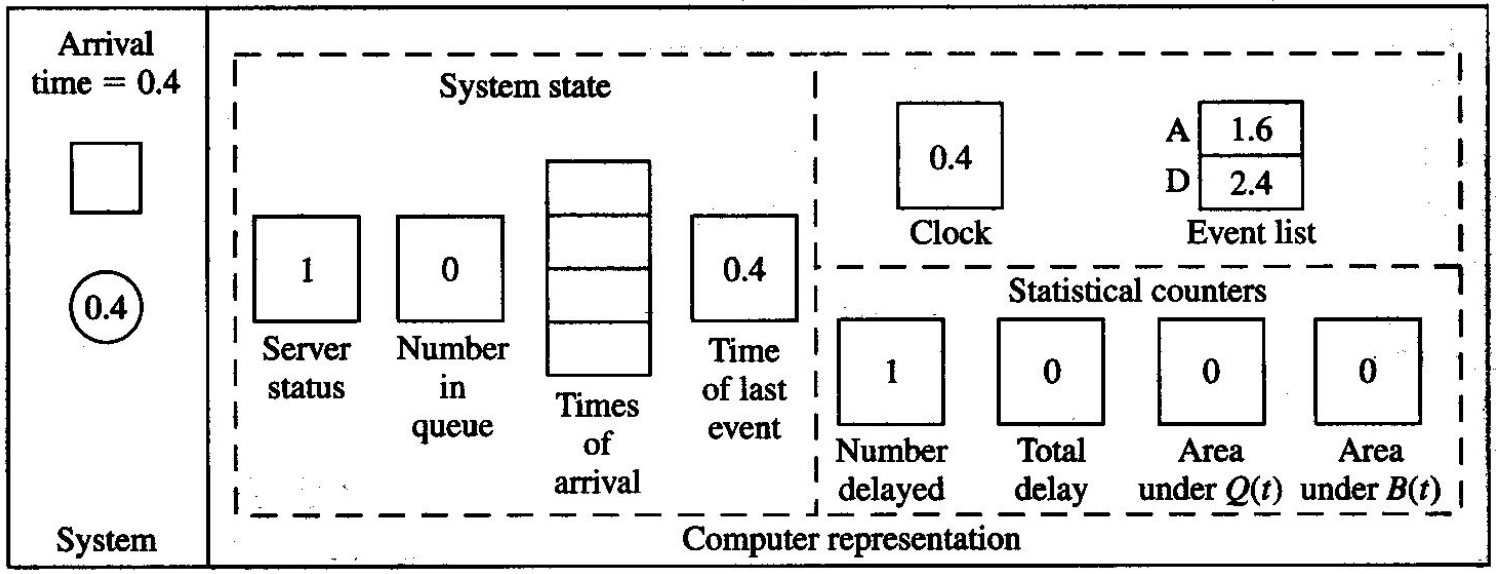
Status shown is *after* all changes have been made in each case …



Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Intuitive Explanation (cont'd)



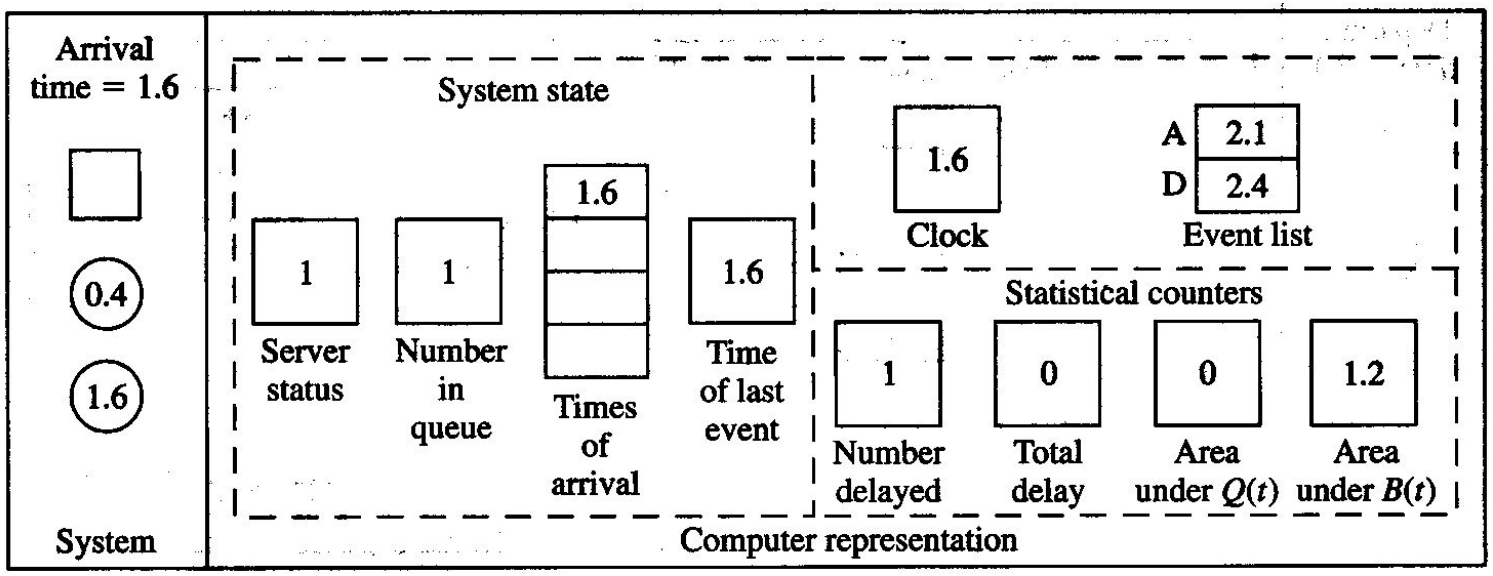
Interarrival times: 0.4, ~~1.2, 0.5~~, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, 0.2, 1.1, 3.7, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



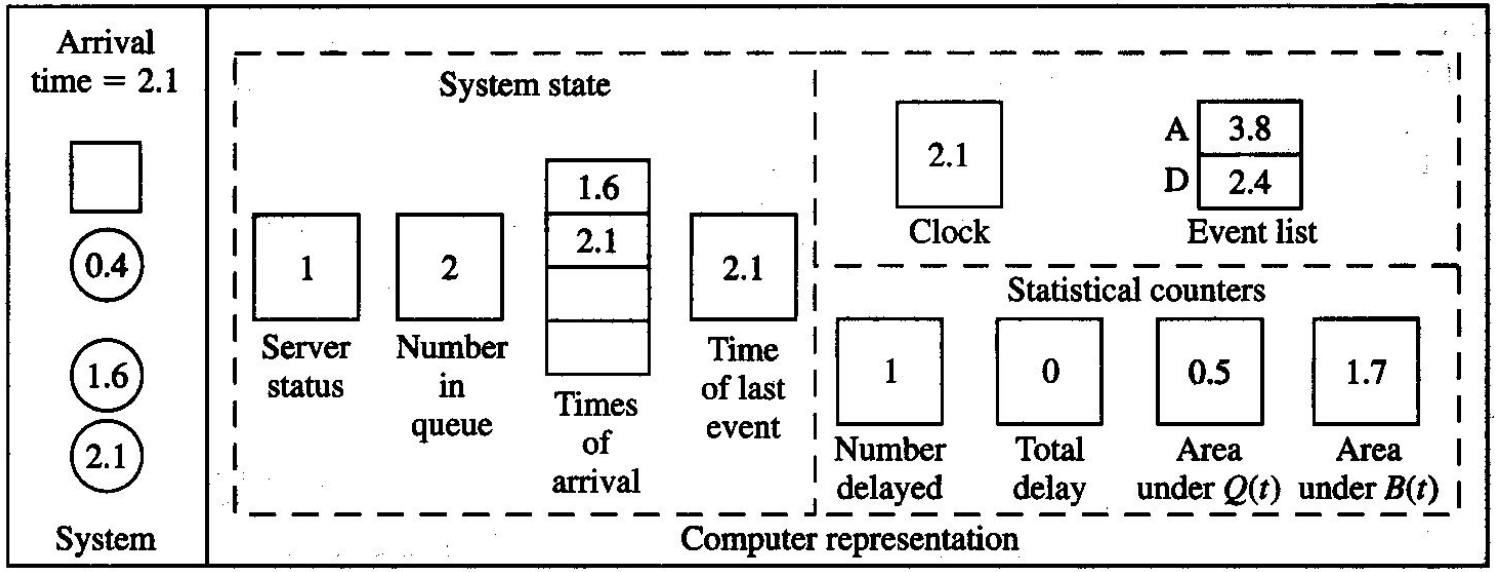
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, 0.2, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, 0.2, 1.1, 3.7, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



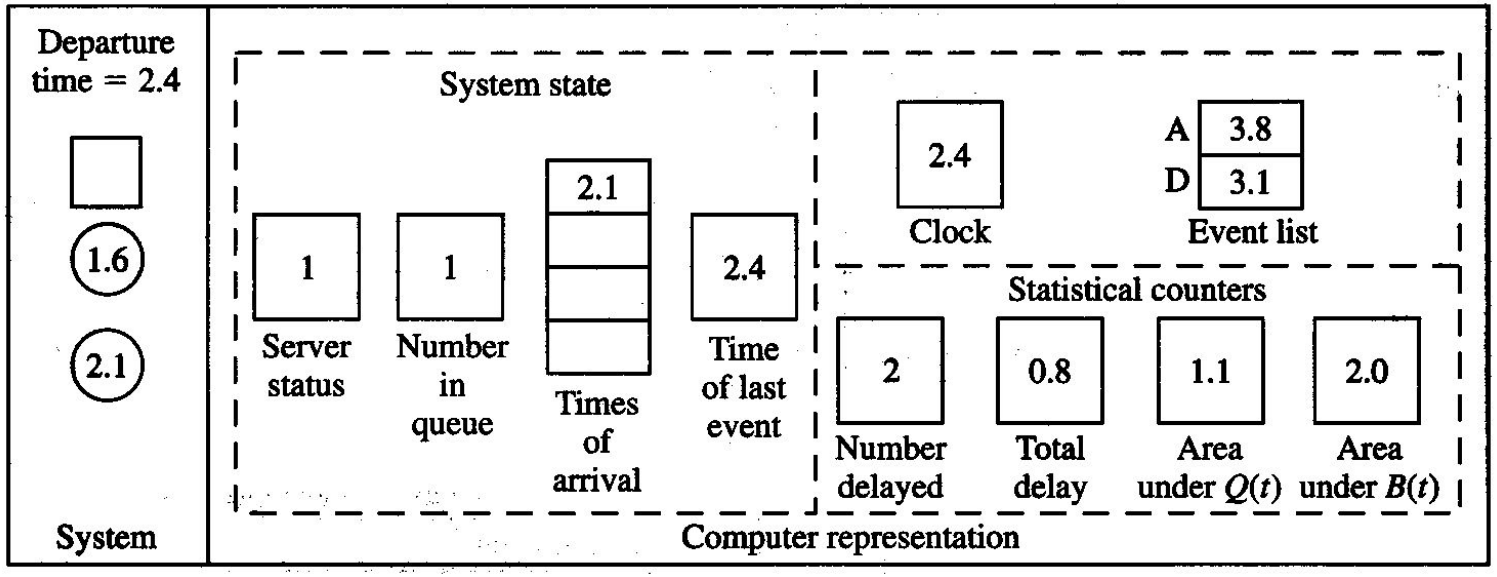
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, 0.2, 1.1, 3.7, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



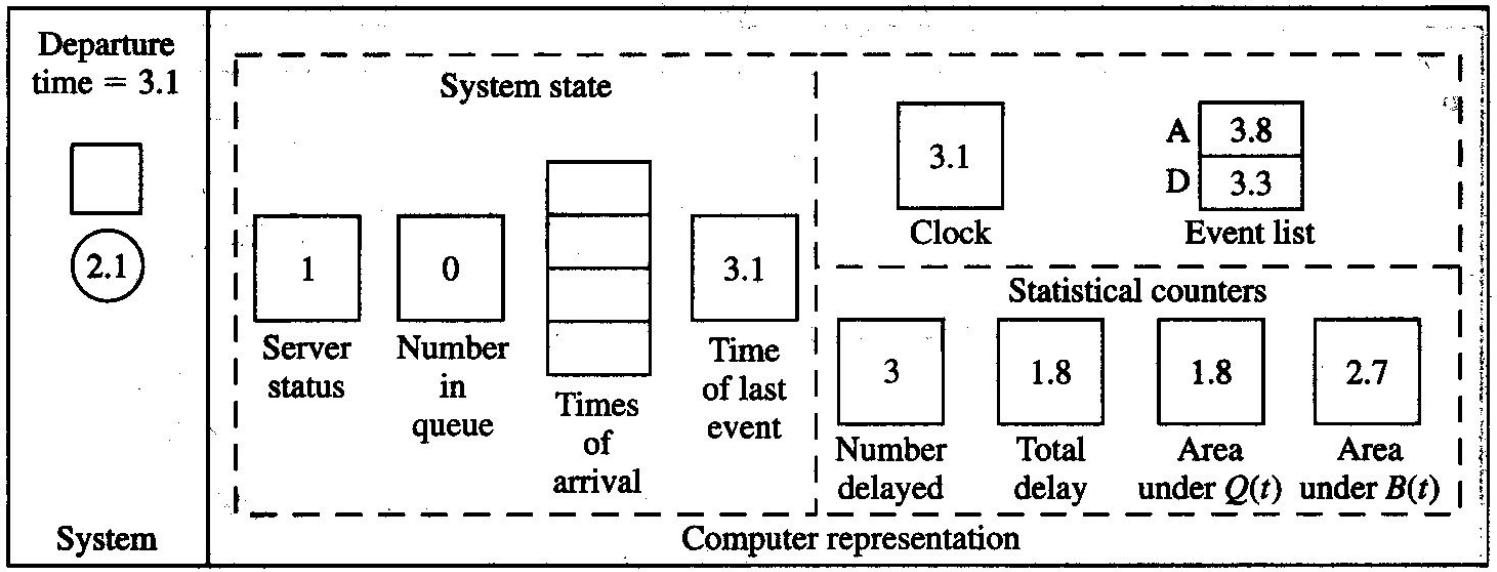
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, 1.1, 3.7, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



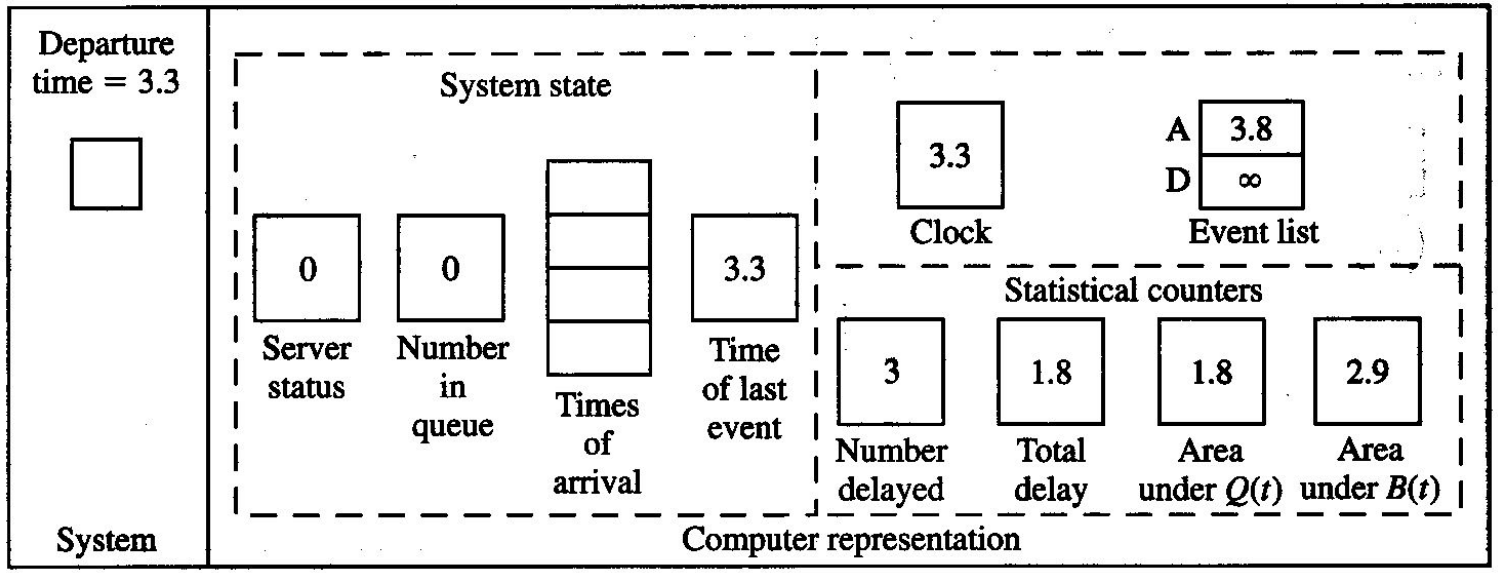
Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



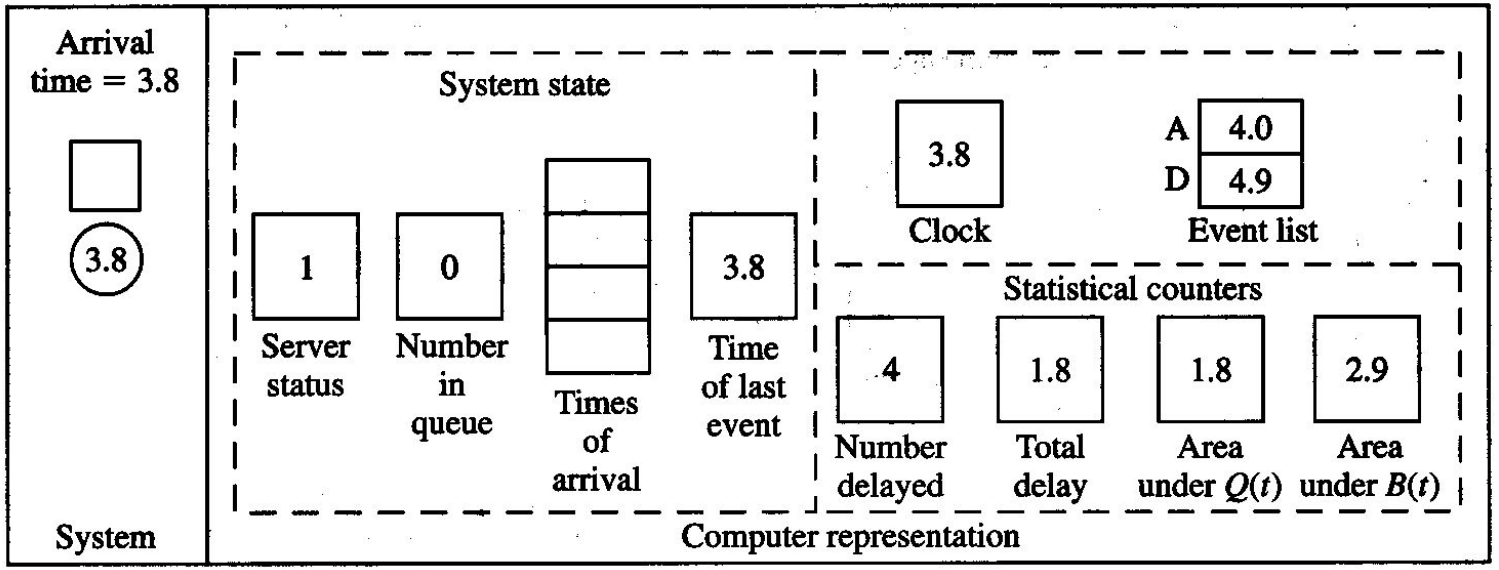
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, 1.6, 0.2, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, ~~1.1~~, 3.7, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



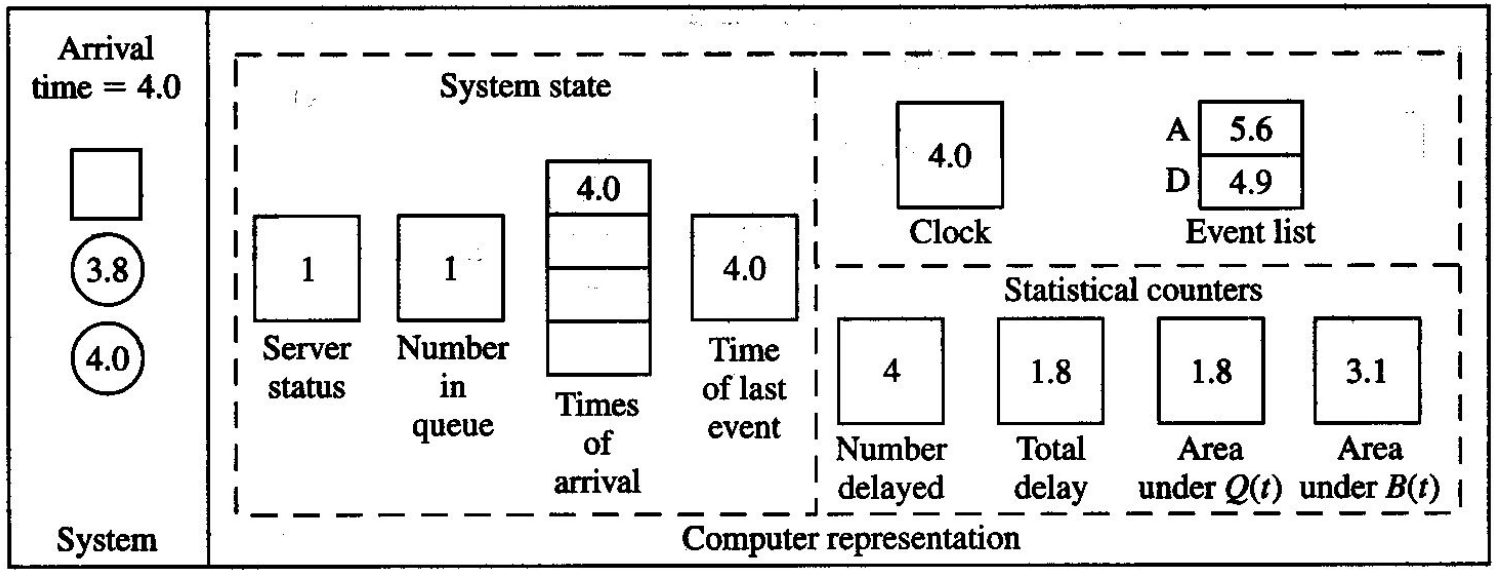
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, ~~1.6~~, 0.2, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, ~~1.1~~, ~~3.7~~, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



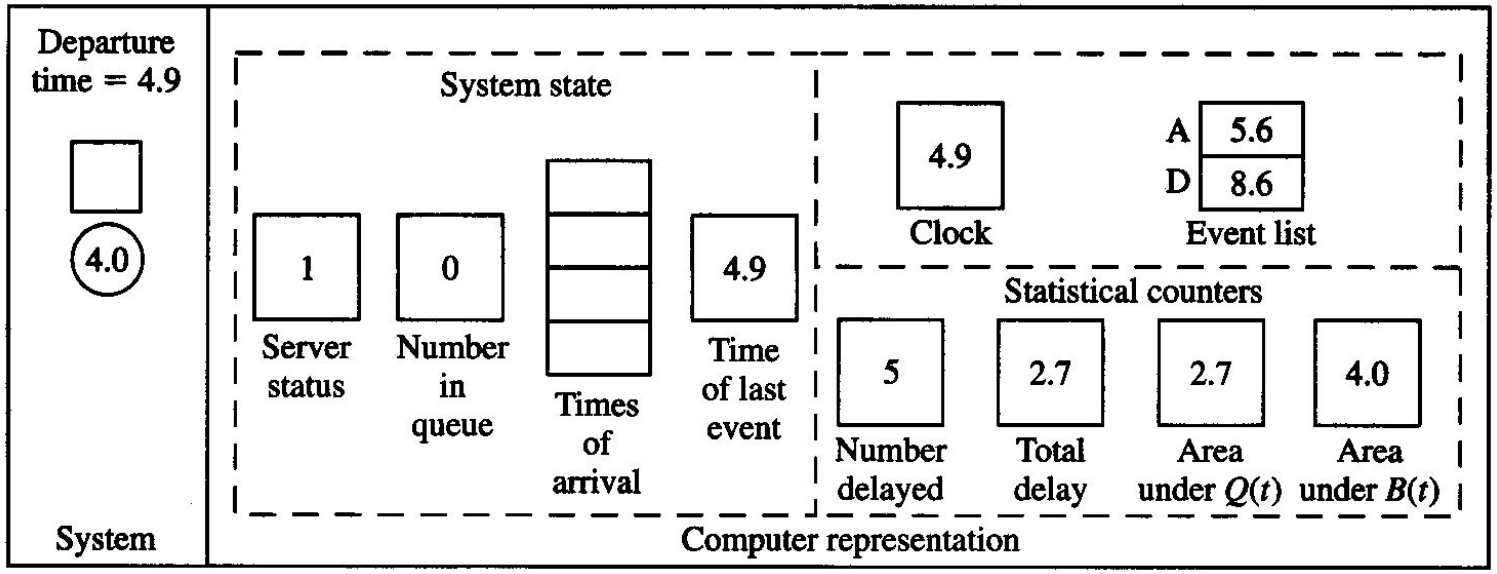
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, ~~1.6~~, ~~0.2~~, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, ~~1.1~~, ~~3.7~~, 0.6, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

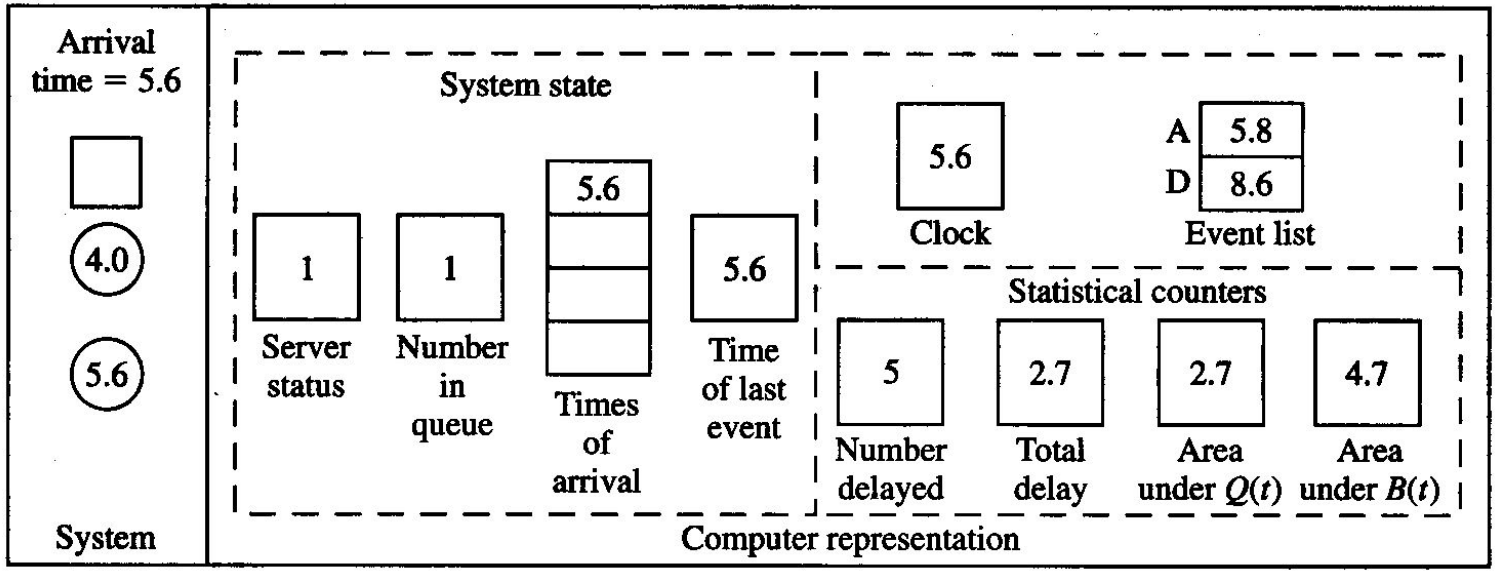
Intuitive Explanation (cont'd)



Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, ~~1.6~~, ~~0.2~~, 1.4, 1.9, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, ~~1.1~~, ~~3.7~~, ~~0.6~~, …

Intuitive Explanation (cont'd)



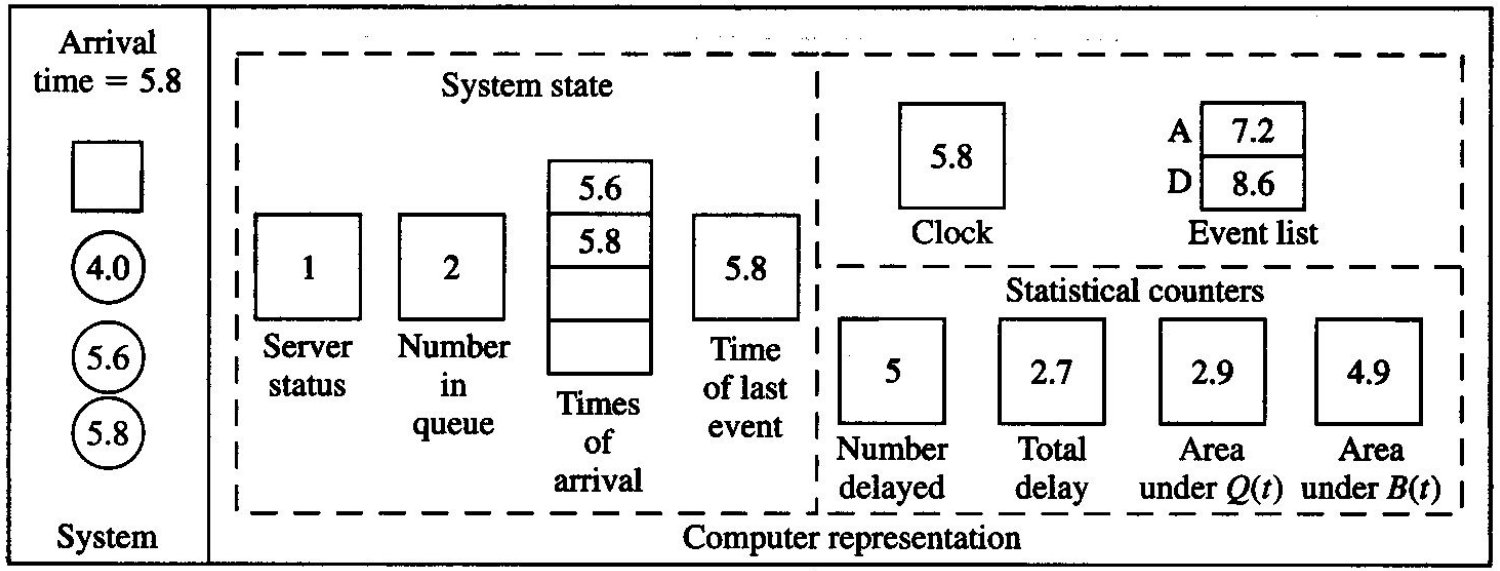
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, ~~1.6~~, ~~0.2~~, ~~1.4~~, 1.9, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, ~~1.1~~, ~~3.7~~, ~~0.6~~, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



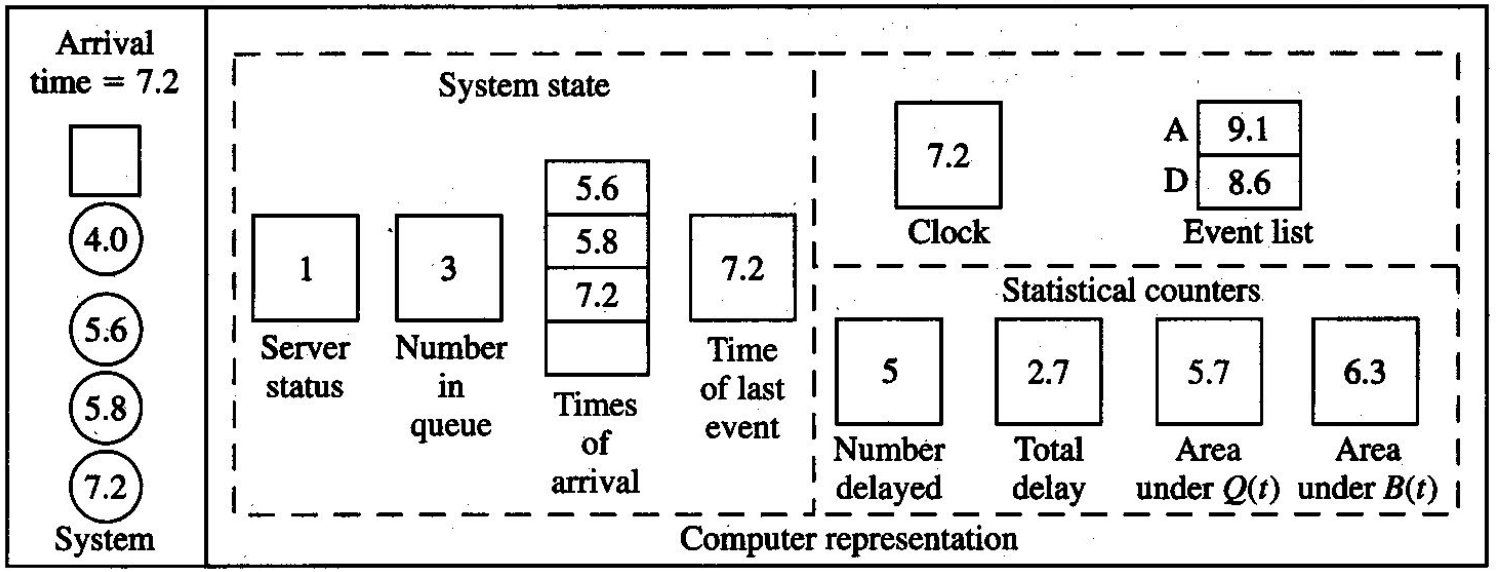
Interarrival times: 0.4, ~~1.2~~, ~~0.5~~, ~~1.7~~, ~~0.2~~, ~~1.6~~, ~~0.2~~, ~~1.4~~, ~~1.9~~, …

Service times: 2.0, ~~0.7~~, ~~0.2~~, ~~1.1~~, ~~3.7~~, ~~0.6~~, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)



Interarrival times: 0.4, ~~1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...~~

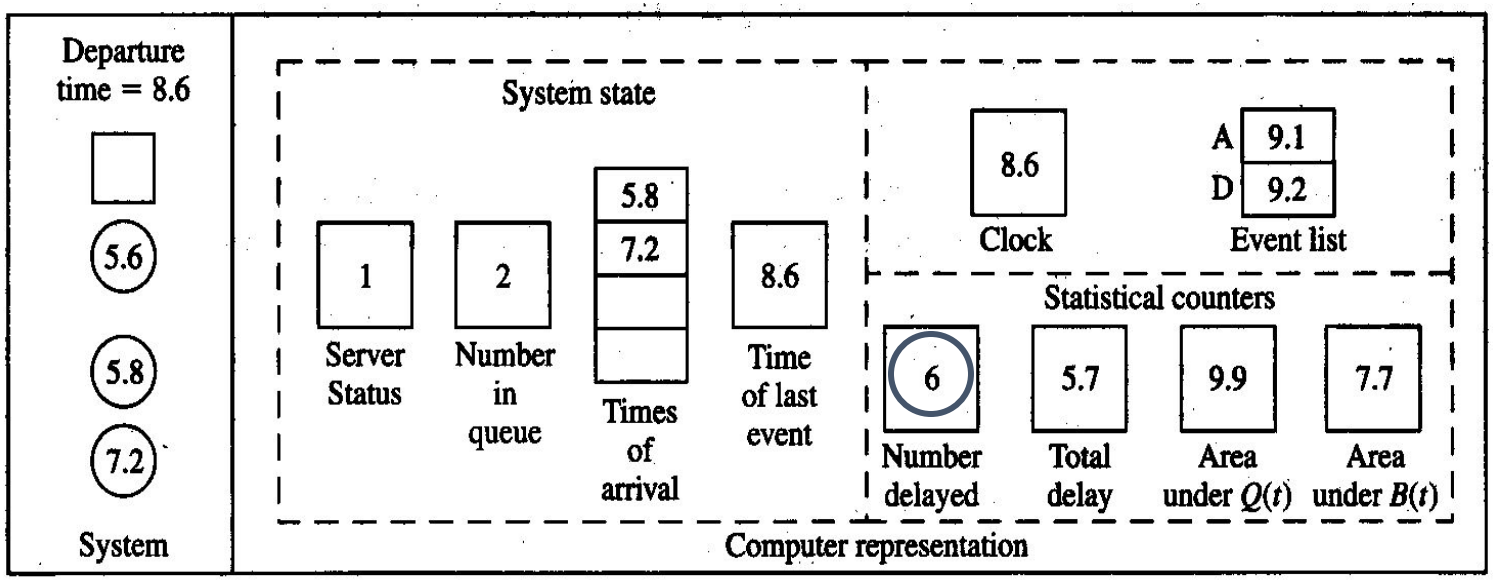
Service times: 2.0, ~~0.7, 0.2, 1.1, 3.7, 0.6~~, …

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.

Departures : 2.4, 3.1,3.3, 4.9, and 8.6

Intuitive Explanation (cont'd)

Arrivals :0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, and 7.2.
Departures : 2.4, 3.1,3.3, 4.9, and 8.6



Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ∞.
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ∞.

Final output performance measures:

Average delay in queue = 5.7/6 = 0.95 min./cust.
Time-average number in queue = 9.9/8.6 = 1.15 custs.
Server utilization = 7.7/8.6 = 0.90 (dimensionless)