Chapter 3
General Principles

Banks, Carson, Nelson & Nicol

Discrete-Event System Simulation

Purpose

- Develops a common framework for the modeling of complex systems.
- Covers the basic blocks for all discrete-event simulation models.
- Introduces and explains the fundamental concepts and methodologies underlying all discrete-event simulation packages.
  - These concepts and methodologies are not tied to any particular simulation package.

Outline

- Deals exclusively with dynamic, stochastic systems.
- Discrete-event models are appropriate for those systems for which changes in system state occur only at discrete points in time.
  - Covers general principles and concepts:
    - Event scheduling/time advance algorithm.
    - The three prevalent world views.
  - Introduces some of the notions of list processing.

Concepts in Discrete-Event Simulation

- System: a collection of entities that interact together over time, e.g., people and machines.
- Model: an abstract representation of a system.
- System state: a collection of variables that contain all the info necessary to describe the system at any time.
- Entity: any object or component in the system, e.g., a server, a customer, a machine.
- Attributes: the properties of a given entity.

Concepts in Discrete-Event Simulation

- Lists: a collection of associated entities, ordered in some logical fashion, a.k.a, sets, queues and chains.
- Event: an instantaneous occurrence that changes the state of a system, e.g., an arrival of a new customer.
- Event list: a list of event notices for future events, ordered by time of occurrence, a.k.a. the future event list (FEL)
- Activity: a duration of time of specified length which is known when it begins , e.g., a service time.
- Clock: a variable representing simulated time.

Note: different simulation packages use different terminology for the same or similar concepts.

Concepts in Discrete-Event Simulation

- An activity typically represents a service time, an interarrival time, or any processing time whose duration has been characterized/defined by the modeler.
  - An activity's duration may be specified:
    - Deterministic
    - Statistical
    - A function depending on system variables and/or entity attributes.
  - Duration is not affected by the occurrence of other events, hence, activity is also called an unconditional wait.
  - Completion of an activity is an event, often called a primary event.
  - For example
    - If the current simulated time is CLOCK = 100 minutes, and an inspection time of exactly 5 minutes is just beginning, then an event notice is created that specified the type of event and the event time (100+5 = 105 min).

Concepts in Discrete-Event Simulation

- A delay's duration is determined by system conditions (not specified by the modeler ahead of time.)
  - Also called a conditional wait.
  - For example, a customer's delay in a waiting line may be dependent on the number and duration of service of other customers ahead in line and, whether a server has a failure during the delay.
- Dynamic: Function of time and constantly changing over time.
  - System state, entity attributes, the number of active entities, the contents of sets, and the activities and delays currently in progress are all function of time.

Concepts in Discrete-Event Simulation

- Example: Able-Baker Call Center System. A discrete-event model has the following components:
  - System state:
    - The number of callers waiting to be served at time *t*
    - Indicator that Able is idle or busy at time *t*
    - Indicator that Baker is idle or busy at time *t*
  - Entities: neither the caller nor the servers need to be explicitly represented, except in terms of the state variables, unless certain caller averages are desired.
  - Events:
    - Arrival
    - Service completion by Able
    - Service completion by Baker
  - Activities:
    - Interarrival time.
    - Service time by Able
    - Service time by Baker
  - Delay: a caller's wait in queue until Able or Baker becomes free.

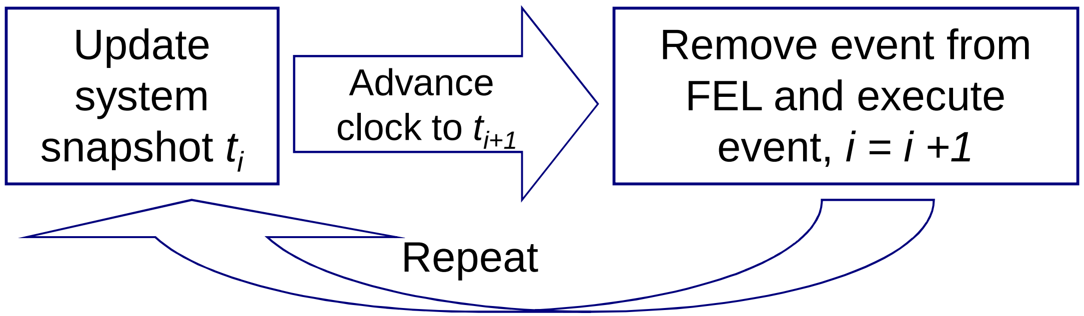
Concepts in Discrete-Event Simulation

- The definition of the model components provides a static description of the model.
- A description of the dynamic relationships and interactions between the components is also needed.
  - e.g., how does each event affect system state? What events mark the beginning or end of each activity? What is the system state at time 0?
- A discrete-event simulation is:
  - The modeling over time of a system all of whose state changes occur at discrete points in time.
  - Proceeds by producing a sequence of system snapshots.

Event Scheduling/Time Advance Algorithm

- The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order.
- At any given time *t*, the future event list (FEL) contains all previously scheduled future events and their associated event times ($t_1, t_2, \ldots$)
  - FEL is ordered by event time, and the event time satisfy:

$$t \le t_1 \le t_2 \le t_3 \le \ldots \le t_n \qquad \text{where t is the value of CLOCK}$$



Event Scheduling/Time Advance Algorithm

Clock	System state	…	Future event list	…
t	(5,1,6)		$(3,t_1)$ – Type 3 event to occur at time t_1 $(1,t_2)$ – Type 1 event to occur at time t_2 $(1,t_3)$ – Type 1 event to occur at time t_3 . . . $(2,t_n)$ – Type 2 event to occur at time t_n	

- *Step 1*. Remove the event notice from the imminent event.
- *Step 2*. Advance clock to imminent event time.
- *Step 3*. Execute imminent event. Update system state.
- *Step 4*. Generate future event list.
- *Step 5*. Update comulative statistics and counters.

Event Scheduling/Time Advance Algorithm

Clock	System state	…	Future event list	…
T_1	(5,1,5)		$(1,t_2)$ – Type 1 event to occur at time t_2 $(4,t^*)$ – Type 4 event to occur at time t^* $(1,t_3)$ – Type 1 event to occur at time t_3 . . . $(2,t_n)$ – Type 2 event to occur at time t_n	

Advancing simulation time and updating system image.

List Processing

[Event Scheduling]

- The management of a list.
  - The major list processing operations performed on a FEL are:
    - Removal of the imminent event
    - Addition of a new event to the list
    - Occasionally removal of some event (cancellation of an event).
  - Efficiency of search within the list depends on the logical organization of the list and how the search is conducted.
- When an event with event time *t** is generated, its correct position on the FEL can be found via:
  - A top-down search, or
  - A bottom-up search.

Future Events

- An exogenous event is a happening "outside the system" that impinges on the system, e.g., arrival and service completion in a queueing system.
- Arrival event:
  - An exogenous event is a happening "outside the system" that impinges on the system.
  - For example, an arrival to a queueing system, at time 0, the *1st* arrival event is generated and is scheduled on the FEL. When the clock eventually is advanced to the time of this first arrival, a second arrival event is generated.
  - The end of an interarrival interval is an example of a primary event (primary event is managed by placing an event notice on the FEL.)

Future Events

- Service completion event:
  - Conditional event
  - Triggered only on the condition that a customer is present and a server is free.
  - A service time is an example of an activity.
- Alternate generation of runtimes and downtimes for a machine subject to breakdowns.
- Stopping event, E:
  - At time 0, schedule a stop simulation event at a specified future time T_E.
  - Run length T_E is determined by the simulation itself. Generally, T_E is the time of occurrence of some specified event E.

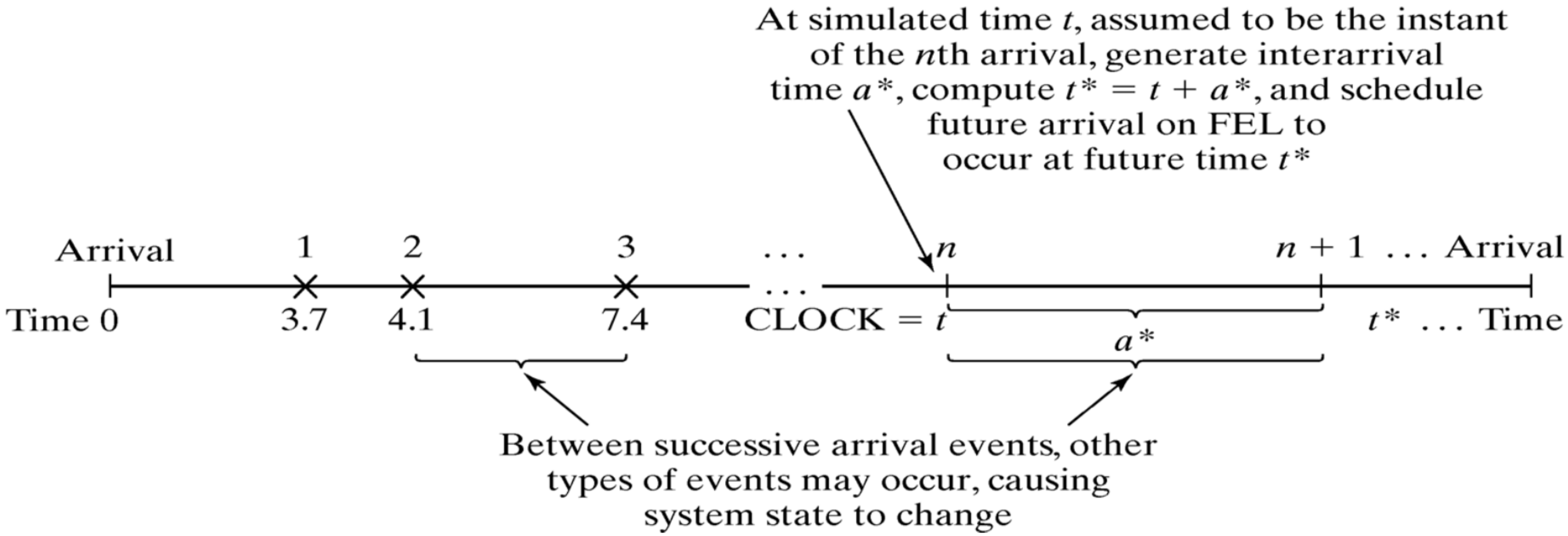
Generation of an external arrival

- The system snapshot at time 0 is defined by the initial conditions and the generation of exogenous events.

- *Bootstrapping :* Generating of an external arrival stream.

Generation of an external arrival

- Generation of an external arrival by bootstrp



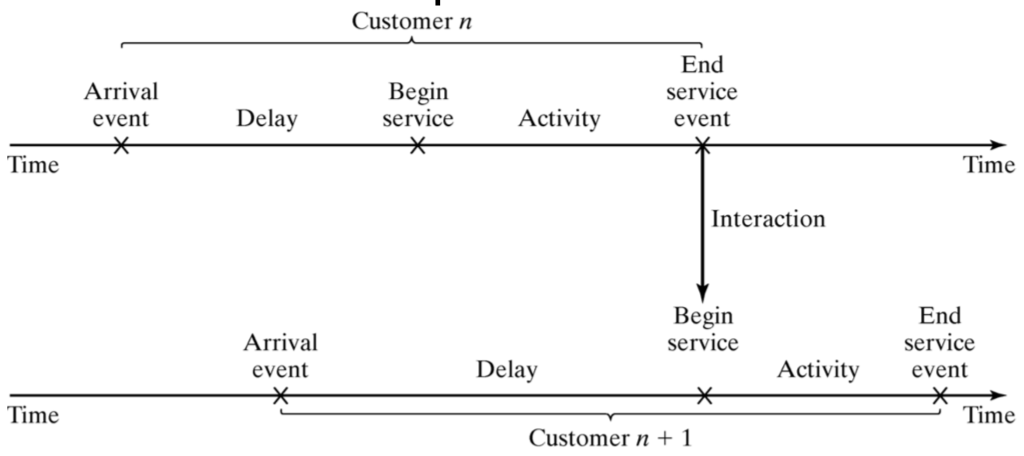
World Views

- The most prevalent world views are:
  - Event-scheduling world view (variable time advance.)
  - Process-interaction world view (variable time advance.)
  - Activity-scanning world view (fixed time increment.)

- Event-scheduling approach
  - Concentrates on events and their effect on system state
  - Summarized in the previous slides; manual simulation will be discussed next.

World Views

- Process-interaction approach
  - Concentrates on the processes: a process is a time-sequenced list of events, activities and delays that define the life cycle of one entity as it moves through a system.
  - Usually, many processes are active simultaneously in a model, and the interaction among processes could be quite complex.
  - A popular approach because it has intuitive appeal and ease of simulation package implementation.
  - An example of a “customer process”:



World Views

- Activity-scanning approach:
  - Concentrates on activities of a model and those conditions that allow an activity to begin.
  - At each clock advance, the conditions for each activity are checked, and if the conditions are true, then the corresponding activities begins.
  - Simple in concept but slow runtime on computers.
  - The modified approach is called the three-phase approach.
    - Events are considered to be activities of duration zero time units.
    - Activities are divided into 2 categories:
      - B-type: activities that are bound to occur, e.g. primary events.
      - C-type: activities or events that are conditional upon certain conditions being true
    - Simulation proceeds with repeated execution of the 3 phases: from removal of events, then execute B-type events, and then execute C-type events.

3-phase approach

- The simulation proceeds with repeated execution of the 3-phase until it is completed:
  - **Phase A.** Remove the imminent event from the FEL and advance the clock to its event time.
  - **Phase B.** Execute all B-type events, that were removed from FEL.
  - **Phase C.** Scan the conditions that trigger each C-type activity.

World Views

- Commercial Use:
  - Process-interaction approach has been adopted by simulation packages most popular in the U.S.
  - Activity-scanning packages are popular in the UK and Europe.

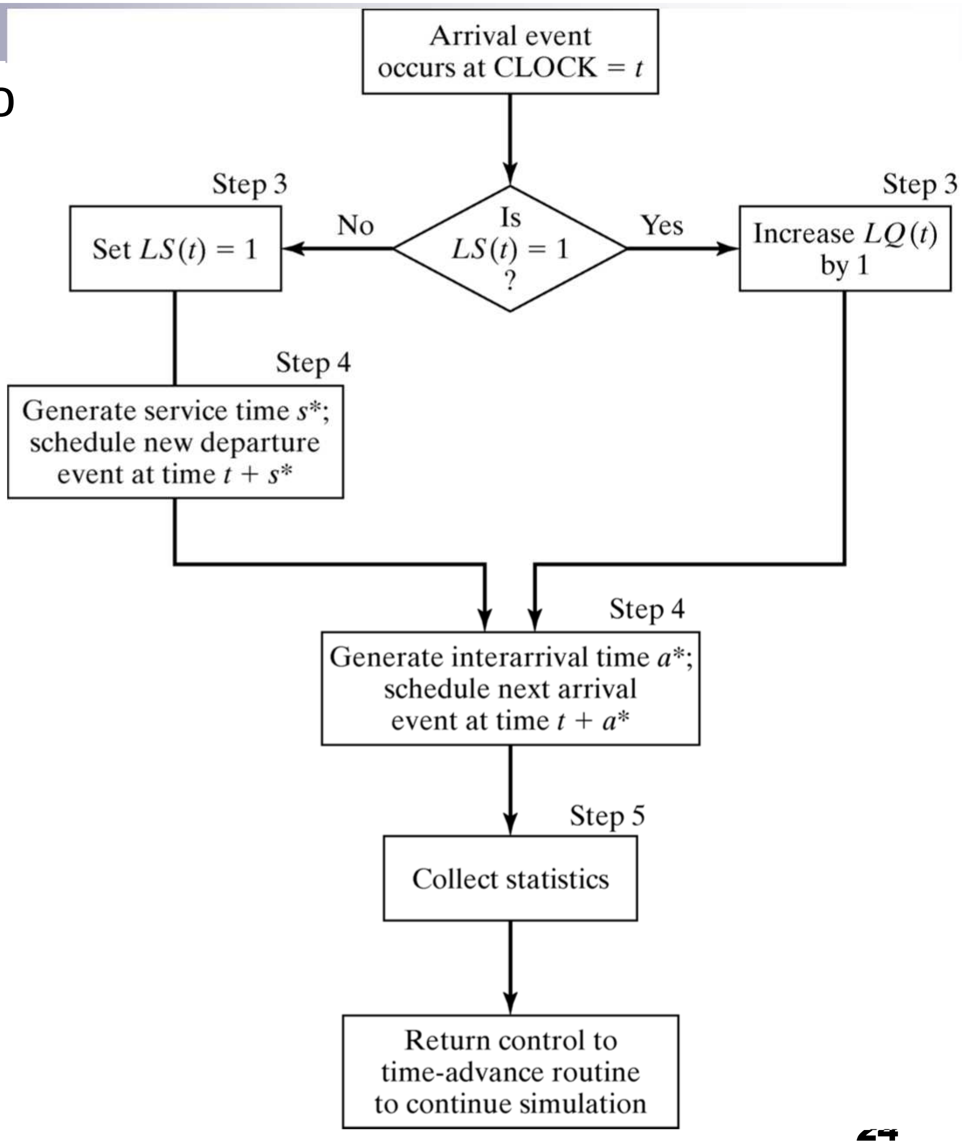
Manual Simulation Using Event Scheduling

- Grocery Store Example: Single-channel queue. Reconsider the single checkout counter problem.
  - The system consists of those customers in the waiting plus the one (if any) checking out.
  - For this example, a stopping time of 60 minutes is set.
  - Model components:
    - System state: *LQ(t)* – # of customers in line at time *t*, *LS(t)* - # being served at time *t*.
    - Entities: the server and customers are not explicitly modeled, except in terms of the state variables.
    - Events: arrival (A), departure (D), stopping event (E).
    - Event notices (event type, event time):
      - (A, *t*), representing an arrival event to occur at future time *t*;
      - (D, *t*), representing a customer departure at future time *t*;
      - (E, 60), representing the simulation stop event at future time *60*.
    - Activities: interarrival time and service time.
    - Delay: customer time spent in waiting line.

Grocery Store Example

[Manual Simulation]

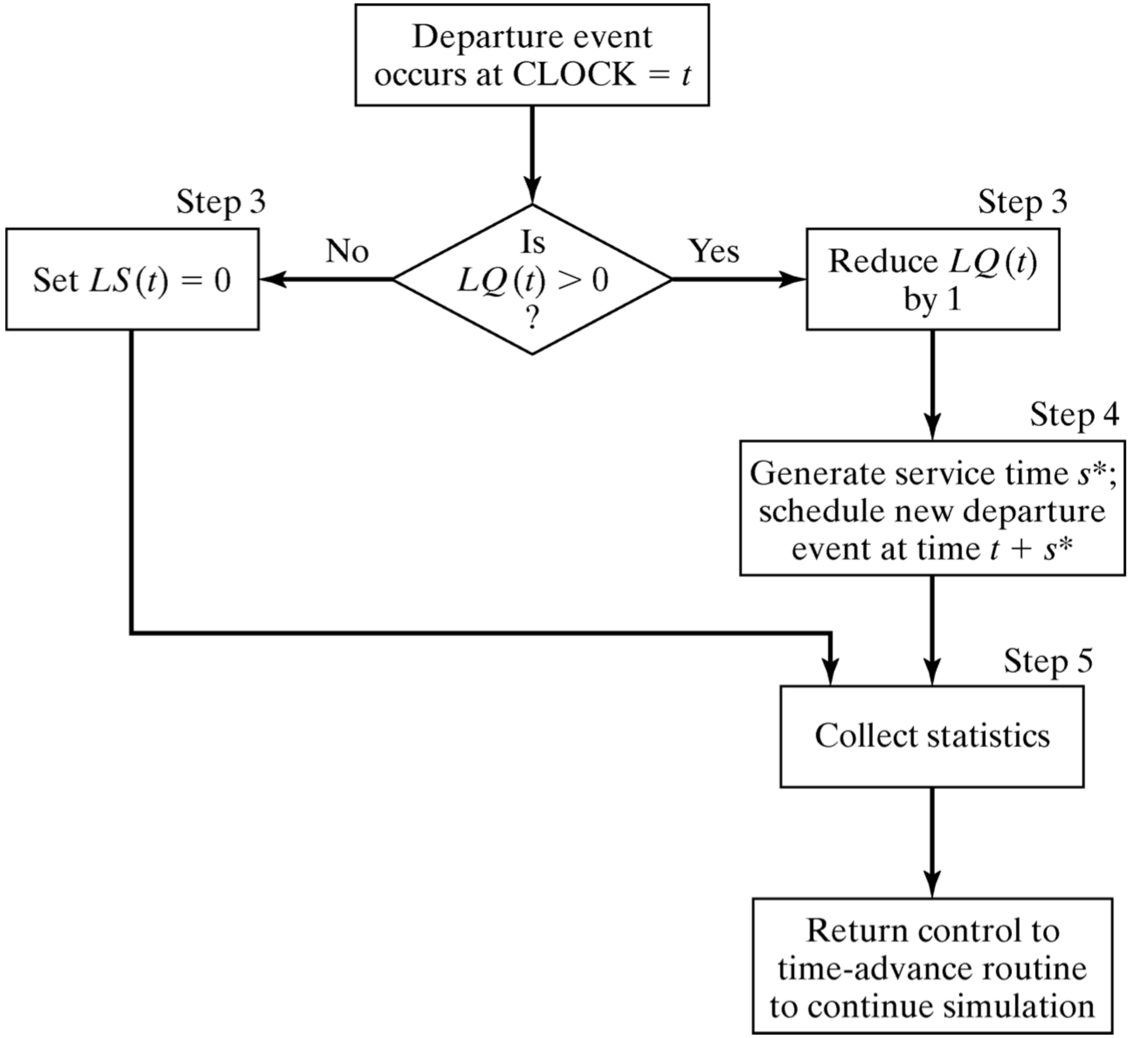
- FEL will always contain two or three event notices.
- Event logic – execution of arrival event.



Grocery Store Example

[Manual Simulation]

- Event logic – execution of departure event.



Grocery Store Example [Manual Simulation]

- Initial conditions are the *1st* customer arrives at time 0 and begin service.
- Only two statistics: server utilization (B) & maximum queue lengths (MQ).
- Simulation table:

Arrival at time 8

Departure at time 4

Terminating event at time 60

Clock	System State		Future Event List	Comment	Cumulative Statistics	
	LQ(t)	LS(t)			B	MQ
0	0	1	(D, 4), (A, 8), (E, 60)	First A occurs: (a* = 8), schedule next A; (s*=4) Schedule first D	0	0
4	0	0	(A, 8), (E, 60)	First D occurs: (D, 4)	4	0
8	0	1	(D, 9), (A, 14), (E, 60)	Second A occurs: (A, 8); (a* = 6) Schedule next A; (s* = 1) Schedule next D	4	0
9	0	0	(A, 14) (E, 60)	Second D occurs: (D, 9)	5	0
14	0	1	(A, 15) (D, 18) (E, 60)	Third A occurs: (A, 14); (s* = 4) Schdeule next D	5	0
15	1	1	(D, 18), (A, 23), (E, 60)	Fourth A occurs: (A, 15) (Customer delayed)	6	1
18	0	1	(D, 21) (A, 23) (E, 60)	Third D occurs: (D, 18); (s* = 3) schedule next D	9	1

Manual Simulation

- When an event-scheduling algorithm is computerized, only one snapshot (the current one or partially updated one) is kept in computer memory.
  - A new snapshot can be derived only from the previous snapshot, newly generated random variables, and the event logic.
  - The current snapshot must contain all information necessary to continue the simulation.

Grocery Store Example [Manual Simulation]

- Suppose the simulation analyst desires to estimate mean response time and mean proportion of customers who spend *5* or more minutes in the system.
  - It is necessary to expand the previous model to represent the individual customers explicitly.
    - Customer entity with arrival time as an attribute will be added to the list of model components,
    - Customer entities will be stored in a list to be called "CHECKOUTLINE" as: *C1, C2, C3*, …,
  - Three new cumulative statistics will be collected.
    - *S*, the sum of customer response times for all customers who have departed by the current time.
    - *F*, the total number of customers who spend 4 or more minutes at the checkout counter.
    - N_D the total number of departures up to the current simulation time.

Grocery Store Example [Manual Simulation]

Simulation Table:

(C_i, t) –customer C_i who arrived at time t

(D, t, C_i) –the departure of customer C_i at future time t

(A, t, C_i) –the arrival of customer C_i at future time t

Clock	System State		List "CHECKOUT LINE"	Future Event List	Cumulative Statistics		
	LQ(t)	LS(t)			S	N_D	F
0	0	1	(C1, 0)	(D, 4, C1), (A, 8, C2), (E, 60)	0	0	0
4	0	0		(A, 8, C2), (E, 60)	4	1	1
8	0	1	(C2, 8)	(D, 9, C2), (A, 14, C3), (E, 60)	4	1	1
9	0	0		(A, 14, C3) (E, 60)	5	2	1
14	0	1	(C3, 14)	(A, 15, C4) (D, 18, C3) (E, 60)	5	2	1
15	1	1	(C3, 14) (C4, 15)	(D, 18, C3), (A, 23, C5), (E, 60)	5	2	1
18	0	1	(C4, 15)	(D, 21, C4) (A, 23, C5) (E, 60)	9	3	2

At time 18, when the departure event (D, 18, C3) is being executed, the response time for customer C3 is computed by:

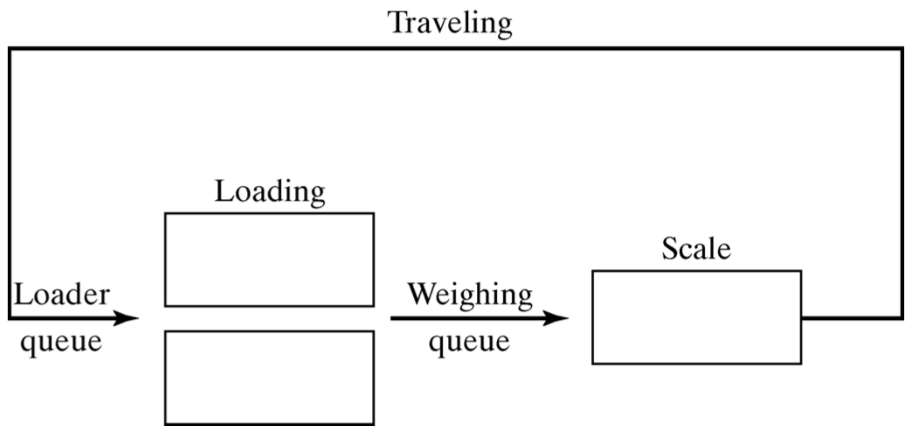
Response time = CLOCK TIME – attribute “time of arrival”
= 18 – 14
= 4 minutes

The S is incremented by 4 minutes, and F and N_D by one customer.

Dump-Truck Example

[Manual Simulation]

- Six dump trucks are used to haul coal from the entrance of a small mine to the railroad.
  - Each truck is loaded by one of two loaders.
  - After loading, the truck immediately moves to the scale to be weighed.
  - The loaders and the scale have a FCFS waiting line (or queue) for trucks.
  - After being weighed, a truck begins a travel time (during which the truck unloads) and returns to the loader queue.



Dump-Truck Example

[Manual Simulation]

- The distributions of loading time, weighing time and travel time are:

Loading Time	Probability	Cumulative Probability	Random Digit Assignment
5	0.3	0.3	1 - 3
10	0.5	0.8	4 - 8
15	0.2	1	9 - 0

Weighing Time	Probability	Cumulative Probability	Random Digit Assignment
12	0.7	0.7	1 - 7
16	0.3	1	8 - 0

Travel Time	Probability	Cumulative Probability	Random Digit Assignment
40	0.4	0.4	1 - 4
60	0.3	0.7	5 - 7
80	0.2	0.9	8 - 9
100	0.1	1	0

Dump-Truck Example

[Manual Simulation]

- Purpose: to estimate the loader and scale utilizations (% of time busy.)
- The model has the following components:
  - System State [LQ(t), L(t), WQ(t), Q(t)], where
    - LQ(t) = number of trucks in loader queue
    - L(t) = number of trucks (0, 1, or 2) being loaded
    - WQ(t) = number of trucks in weigh queue
    - W(t) = number of trucks (0 or 1) being weighed
  - Event notices:
    - (ALQ, t, DTi), dump truck i arrives at loader queue (ALQ) at time t
    - (EL, t, DTi), dump truck i ends loading (EL) at time t
    - (EW, t, DTi), dump truck i ends weighing (EQ) at time t
  - Entities: The six dump trucks (DT1, …, DT6)
  - Lists:
    - Loader queue, all trucks waiting to begin loading, ordered in FCFS basis.
    - Weigh queue, all trucks waiting to be weighed, ordered on a FCFS basis.
  - Activities: Loading time, weighing time, and travel time.
  - Delay: Delay at loader queue, and delay at scale

Dump-Truck Example

[Manual Simulation]

- When an end-loading (EL) event occurs, say for truck *j* at time *t*, other events are triggered:
  - If the scale is idle [$W(t) = 0$], truck *j* begins weighing and an end-weighing event (EW) is scheduled on the FEL. Otherwise, truck *j* joins the weigh queue.
  - If there is another truck waiting for a loader, it will be removed from the loader queue and begin loading by the scheduling of an end-loading event (EL) on the FEL.
  - Both this logic for the occurrence of the end-loading event and the appropriate logic for the other two events should be incorporated into an event diagram.

Dump-Truck Example

[Manual Simulation]

- Simulation Table:

Truck *3* joins the weigh queue (because the scale is occupied.)

Truck *4* begins to load, schedule an EL event for future time *10*

At time 0, *5* trucks at the loaders and *1* is at the scale

The imminent event is an EL event with time *5*, hence clock is advanced to time *t=5*

Clock t	System State				Loader Queue	Weigh Queue	Future Event List	Cumulative Statistics	
	LQ(t)	L(t)	WQ(t)	W(t)				B_L	B_S
0	3	2	0	1	DT4 DT5 DT6		(EL, 5, DT3) (EL, 10, DT2) (EW, 12, DT1)	0	0
5	2	2	1	1	DT5 DT6	DT3	(EL, 10, DT2) (EL, 5+5, DT4) (EW, 12, DT1)	10	5
10	1	2	2	1	DT6	DT3 DT2	(EL, 10, DT4) (EW, 12, DT1) (EL, 10+10, DT5)	20	10
10	0	2	3	1		DT3 DT2 DT4	(EW, 12, DT1) (EL, 20, DT5) (EL, 10+15, DT6)	20	10
12	0	2	2	1		DT2 DT4	(EL, 20, DT5) (EW, 12+12, DT3) (EL, 25, DT6) (ALQ, 12+60, DT1)	24	12
20	0	1	3	1		DT2 DT4 DT5	(EW, 24, DT5) (EL, 25, DT6) (ALQ, 72, DT1)	40	20
24	0	1	2	1		DT4 DT5	(EL, 25, DT6) (EW, 24+12, DT2) (ALQ, 72, DT1) (ALQ, 24+100, DT3)	44	24
25	0	0	3	1		DT4 DT5 DT6	(EW, 36, DT2) (ALQ, 72, DT1) (ALQ, 124, DT3)	45	25
36	0	0	2	1		DT5 DT6	(EW, 36+16, DT4) (ALQ, 72, DT1) (ALQ, 36+40, DT2) (ALQ, 124, DT3)	45	36

Dump-Truck Example [Manual Simulation]

- Two cumulative statistics are maintained:
  - B_L = total busy time of both loaders from time 0 to time t
  - B_S = total busy time of the scale from time 0 to time t
  - From the simulation table, we know that:
    - Both loaders are busy from time 0 to time 20, so B_L = 40 at time = 20.
    - From time 20 to time 24, only one loader is busy, thus B_L increases by only 4 minutes over the time interval [20, 24].
    - From time 25 to time 36, both loaders are idle ($L(25) = 0$), so B_L does not change.

Dump-Truck Example

[Manual Simulation]

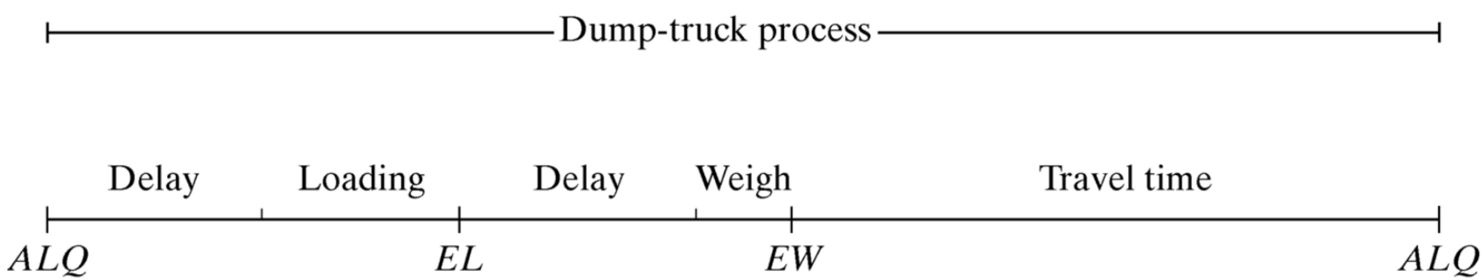
- Under the activity-scanning approach, the conditions for beginning each activity are:

Activity	Condition
Load time	Truck is at front of loader queue, and at least one loader is idle.
Weighing time	Truck is at front of weigh queue, and weigh scale is idle.
Travel time	Truck has just completed a weighing.

Dump-Truck Example

[Manual Simulation]

- Using process-interaction approach, we view the model from the viewpoint of one dumper truck and its “life cycle.”
  - Considering a life cycle as beginning at the loader queue, we can picture a dump-truck process as:



Lists: Basic Properties and Operations

- Lists are a set of ordered or ranked records.
  - In simulation, each record represents one entity or one event notice.
  - They have top or head (*1st* item) and bottom or tail.
  - Ways to traverse the list (to find the *2nd*, *3rd*, etc. items on the list) are necessary.
- An entity identifier and its attributes are *fields* in the entity record.
- Each record on a list has a field that holds a “next pointer” that points to the next record on the list.

Lists: Basic Properties and Operations

- The main operations on a list are:
  - Removing a record from the top of the list
  - Removing a record from any location on the list
  - Adding an entity record to the top or bottom of the list
  - Adding a record at an arbitrary position in the list, specified by the ranking rule.
- In the event-scheduling approach, when time is advanced and the imminent event is due to be executed:
  - First, the removal operation takes place.
  - If an arbitrary event is being canceled, or an entity is removed from a list based on some of its attributes to being an activity, then the second removal operation is performed
  - If a queue has the ranking rule *earliest* due date first, then, upon arrival at the queue, an entity must be added to the list determined by the due-date ranking rule.

Lists: Basic Properties and Operations

- For simulation on a computer:
  - All records are stored in arrays: arrays hold successive records in contiguous locations in computer memory, referenced by array index.
  - All entities and event notices are represented by structure (as in C) or classes (as in Java) allocated from RAM memory as needed, and tracked by pointers to a record or structure.

Using Arrays

- The array method of list storage is typical of FORTRAN.
- Most modern simulation packages do not use arrays for list storage, but rather use dynamically allocated records.
- Advantages of arrays:
  - Any specified record (say the i^{th} record) can be retrieved quickly without searching, merely by referencing $R(i)$
- Disadvantages of arrays:
  - The list is rearranged when items are added to the middle of a list.
  - Have a fixed size which is determined at compile time.

Using Arrays

- Two basic methods for keeping track of record ranking in a list:
  - Method 1: Store the *1st* record in *R(1)*, *2nd* in *R(2)*, …, and list in *R(tailptr)*.
    - Extremely inefficient.
    - For example, adding a record in position *41* in a list of *100* items, it requires that the last *60* records be physically moved down one array position to make space for the new record.
  - Method 2: Use head pointer.
    - A variable that points to the record at the top of the list, denoted as *headptr*.
    - For example, if the record in position *R(11)* were the record at the top of the list, then *headptr* would have the value *11*.

Dump-Truck Example [List Processing]

- Recall the dump-truck problem: At clock time *10*, there is waiting line of *3* dump trucks occurred at the weigh queue, specifically, DT3, DT2 and DT4 (in this order.)
- Suppose the model is tracking one attribute of each dump truck: its arrival time at the weigh queue, updated each time it arrives.
- Suppose that the entities are stored in records in an array dimensioned from *1* to *6*, one record for each dump truck.
  - Each entity is represented by a record with *3* fields:
    - The first is an entity identifier,
    - The second is the arrival time at the weigh queue,
    - The last is pointer field to "point to" the next record.

[DT*i*, arrival time at weigh queue, next index]

Dump-Truck Example [List Processing]

- At time 0, the records would be initialized as follows:

  R(1) = [DT1, 0.0, 0], R(2) = [DT2, 0.0, 0], R(3) = [DT3, 0.0, 0]

  R(4) = [DT4, 0.0, 0], R(5) = [DT5, 0.0, 0], R(6) = [DT6, 0.0, 0]

- At clock time 10, the list of entities in the weigh queue would be defined by:

  headptr = 3

  R(1) = [DT1, 0.0, 0], R(2) = [DT2, 10.0, 4], R(3) = [DT3, 5.0, 2]

  R(4) = [DT4, 10.0, 0], R(5) = [DT5, 0.0, 0], R(6) = [DT6, 0.0, 0]

- To traverse the list, start with the head pointer, go to that record, retrieve that record's next point, and proceed. To create the list in its logical order, for example:

  headptr = 3,

  R(3) = [DT3, 5.0, 2], R(2) = [DT2, 10.0, 4], R(4) = [DT4, 10.0, 0]

  - The zero entry for next point in R(4), as well as tailptr = 4, indicates that DT4 is at the end of the list.

Dump-Truck Example

[List Processing]

- At clock time 12, dump truck DT3 begins weighing and thus leaves the weigh queue.
  - To remove the DT3 entity record form the top of the list, update the head pointer by setting it equal to the next pointer value of the record at the top of the list:

    headptr = R(headptr, next)

  - In this example, we get: headptr = R(3, next) = 2. Hence, dump truck DT2 in R(2) is now at the top of the list.

Using Dynamic Allocation and Linked Lists

- Used in procedural languages, such as C++ and Java, and in most simulation languages.
- Entity records are dynamically created when an entity is created.
- Event notice records are dynamically created whenever an event is scheduled on the future event list.
- Basic mechanics:
  - The languages maintain a linked list of free chunks of computer memory and allocate a chunk of desired size upon request to running programs.
  - When an entity exits from the simulated system, and also after an event occurs and the event notice is no longer needed, the corresponding records are free.

Using Dynamic Allocation and Linked Lists

- A record is referenced by a pointer instead of by an array index.
  - A pointer to a record can be thought of as the physical or logical address in computer memory of the record.
  - If the 3rd item on the list is needed, we need to traverse the list, counting items until we reach the record.
- Notation for records:
  - Entities: [ID, attributes, next pointer]
  - Event notices: [event type, event time, other data, next pointer]
- List types:
  - Singly-linked lists: one-way linkage from the head of the list to its tail.
  - Doubly-linked lists: records have two pointer fields, one for the next record and one for the previous record.

Dump-Truck Example

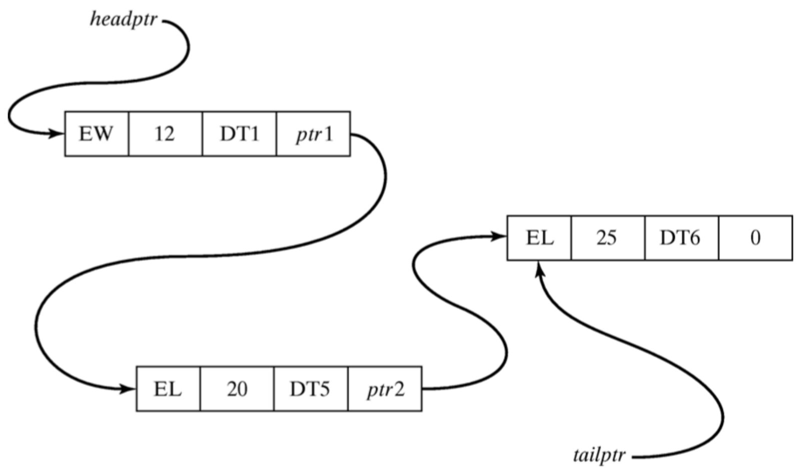
[List Processing]

- Event notices in the dump truck problem are expanded to include a pointer to the next event notice on the future event list, and can be represented by:

  [event type, event time, DT*i*, nextptr]

  - Keep in mind that the records may be stored anywhere in computer memory.

- For example, [EL,10, DT3, nextptr]
  - At future event list at clock time *10*, the future event list as a linked list:



Summary

- Introduced the major concepts and building blocks in simulation:
  - Entities and attributes.
  - Events and activities.
- Three major world views:
  - Event-scheduling.
  - Process interaction.
  - Activity scanning.
- Basic notions of list processing are introduced to gain understanding of this important underlying methodologies.