本讲作业

大作业：编程实现队列模型（M/M/1）的仿真程序，

- 输入参数：平均到达时间，平均服务时间，顾客数目，队列最大长度；
- 输出参数：队列中平均等待客户数；平均等待时间；服务器利用率。

要求提交纸质报告一份，附关键代码和程序截屏。

截止日期：10月16日

报告可以上课时交，也可以交到E206

建模仿真技术

之离散事件仿真

大纲

- 离散事件仿真 (DES)简介
- 术语
- 我们如何推进仿真?
- 案例: 对单服务器排队系统的仿真
- *对DES建模的不同角度 (不同方法)*
- *离散事件仿真系统的组件要素与组织方法*
- *DES仿真与模型构建的步骤*

什么是离散事件仿真(DES)?

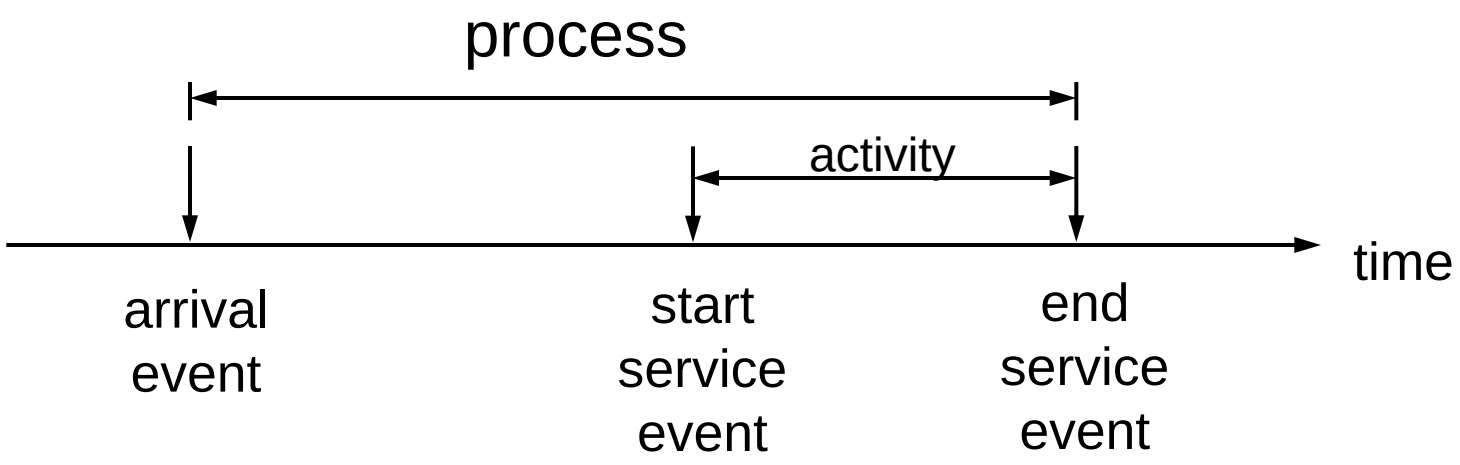
- 离散事件仿真主要研究对象是系统状态只在离散时间点上改变的系统.
  - *一个前提是在连续状态转移之间没有相关事件发生.*
- **系统由实体*(entities)*及其相关属性*(attributes)*构成**
- 系统状态由实体的一系列状态变量属性构成.
- 事件是时间点上瞬时发生的，它可以改变系统的状态.

DES的术语

- 事件：随机发生、能改变系统状态（想象真实世界）.
- 活动：事件触发活动。每个活动意味着实体会在某个时间段上执行某些操作。
- 实体：实体、属性、事件、活动以及他们之间的相互关系构成了离散事件系统的系统模型 。
- 系统环境：任何处于系统边界之外的、能对系统产生影响的对象与活动，所有这些构成了系统环境。
- 仿真时间：某个真实世界的、需要被仿真的时间段.
- 进程：

DES的术语

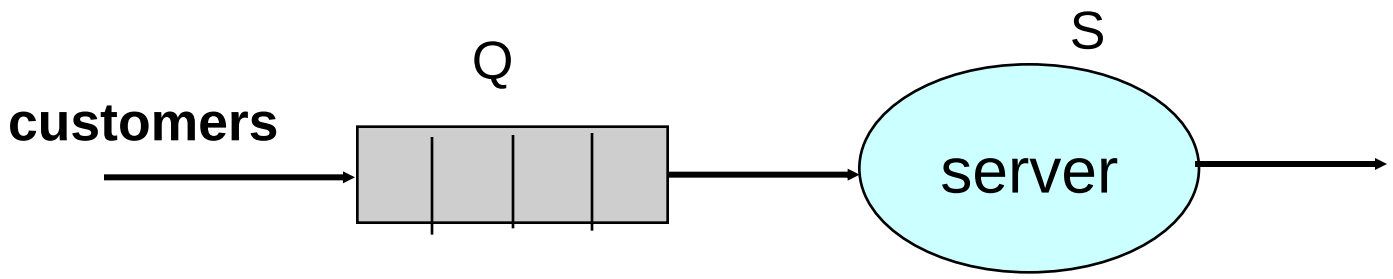
☞一个进程是一个事件序列，它可以包含多个活动(activities).



事件、活动、进程之间的关系

例: 对单服务器排队模型的建模与仿真

- 实体: 顾客, 服务台（服务器）
- **顾客属性:** 所需的服务
- 服务器属性: 能提供的服务、服务率
- **事件**: 顾客到达; 顾客离开
- 活动: 服务顾客，等待新的顾客



如何基于时间来对事件的发生进行建模?

- 我们需要维护一个事件列表来确定下一步发生什么. 事件列表意味着不同类型事件所发生的时间。 (也叫做事件的日历，未来事件列表，事件日志等)

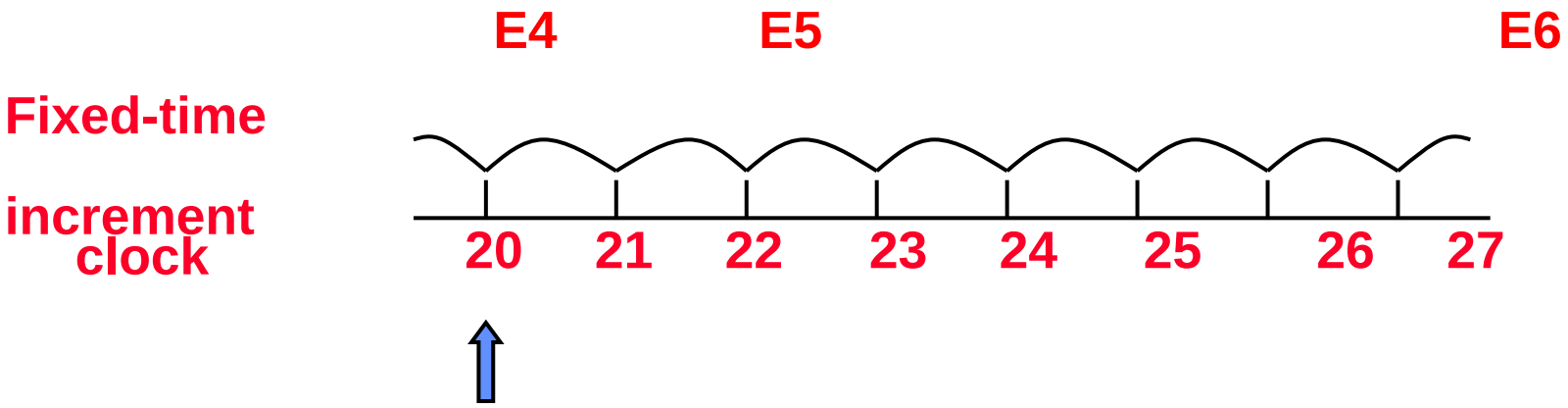
- 同时需要维护一个时钟变量来标记事件发生的时间。

仿真时间的推进

- 每一个动态仿真模型的核心是时间推进机制。因此，每个模型包含一个称为内部时钟的变量，或又称作仿真时钟。
- 我们如何推进仿真时间?
  - *固定时间步长推进方法 (时间的同步模型)*
  - *下一事件推进方法 (时间的异步模型)*

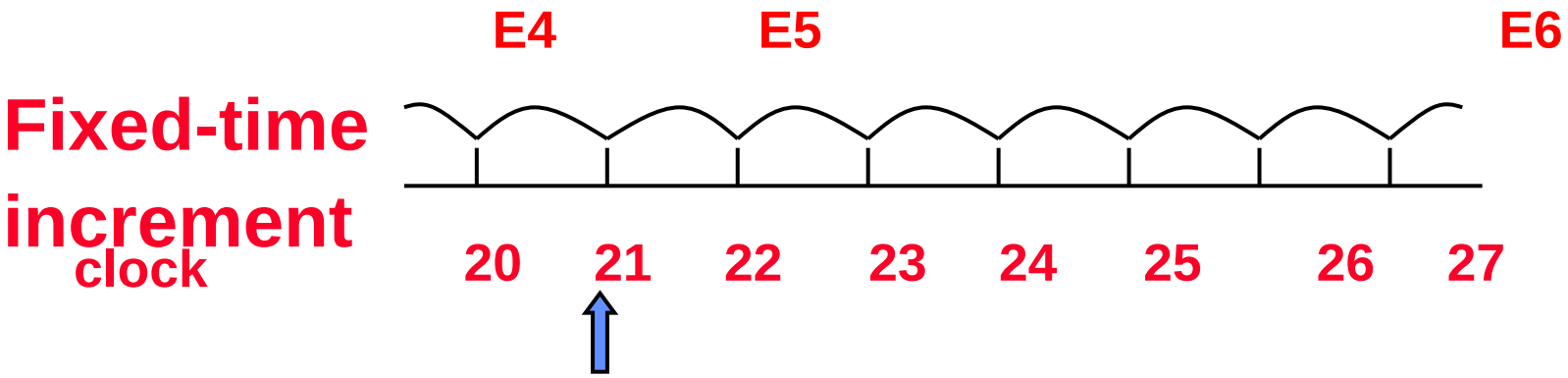
如何推进仿真时间?

- 固定时间步长推进方法 (同步模型)
  - *所有实体的时钟以同样的步长更新, Δt*



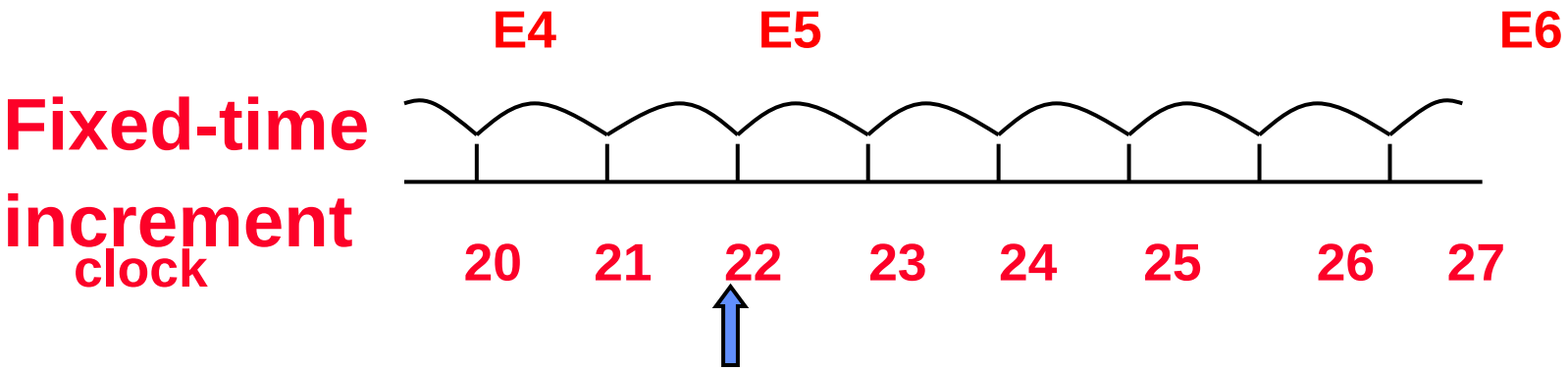
如何推进仿真时间?

- 固定时间步长推进方法 (同步模型，**synchronous model of time)**
  - *所有实体的时钟以同样的步长更新, Δt*



如何推进仿真时间?

- 固定时间步长推进方法 (同步模型)
  - *所有实体的时钟以同样的步长更新, Δt*

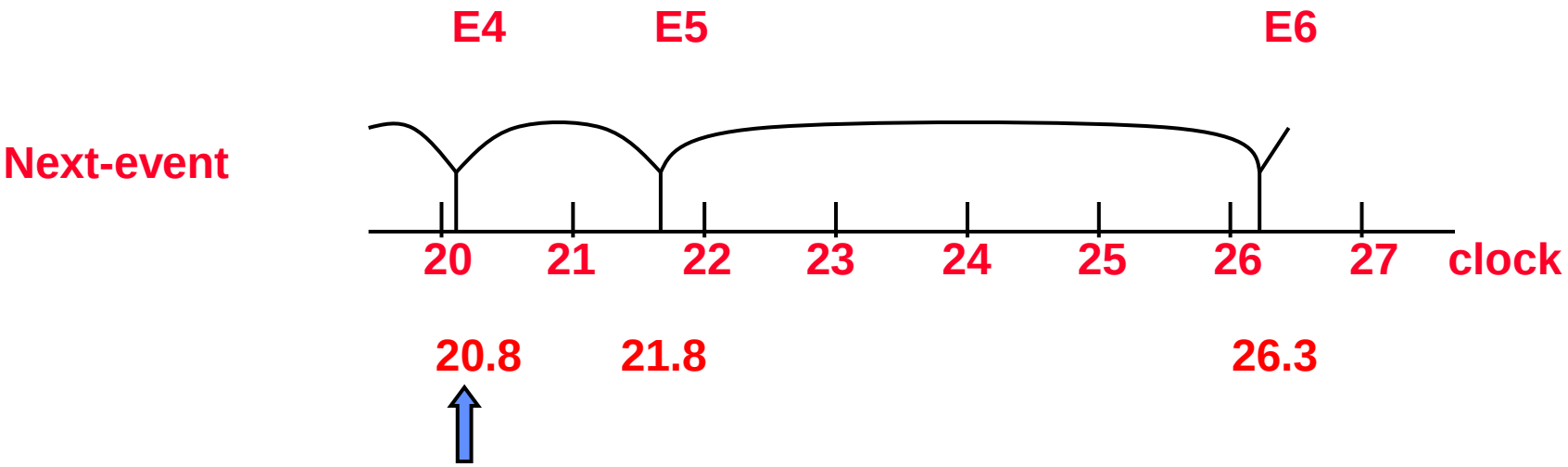


固定时间间隔推进

- 当所有实体按步长更新后，所有应该在步长内发生的事件就都清楚了。我们会假设事件都在步长的最后一刻发生，相应地，系统状态也在此时更新。
- 当系统事件发生在固定时间间隔时非常有用。
- 缺点:
  - *无用的扫描 – 如果 Δt没有事件发生*
  - *准确度问题- Δt 太大会损失精度 (事件只在 Δt的倍数时间上发生)*

如何推进仿真时间?

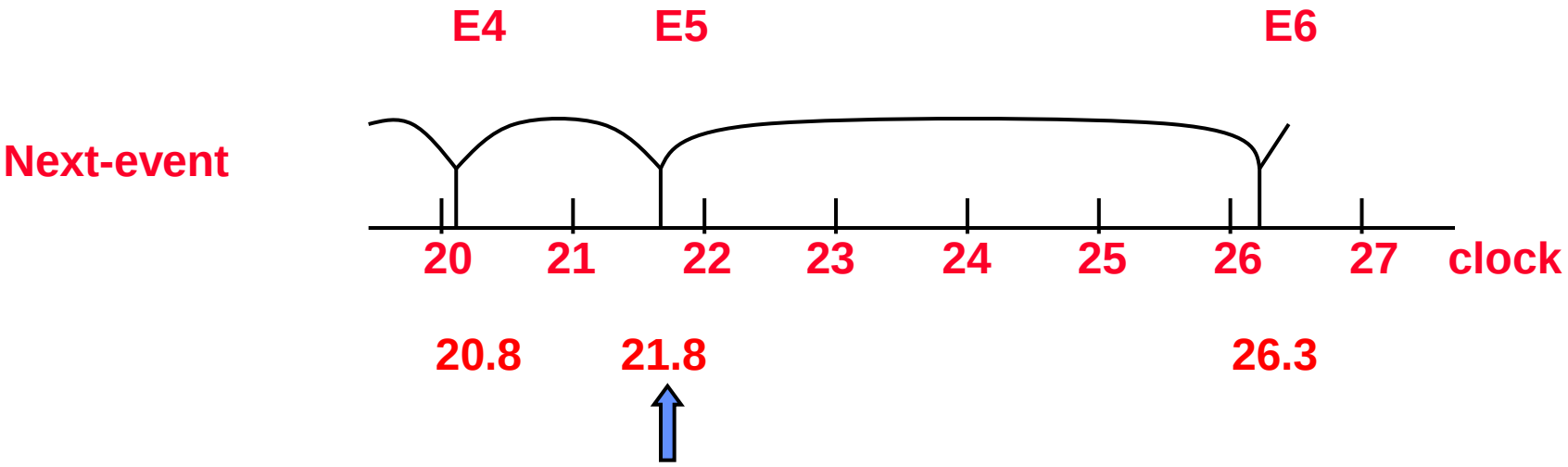
- ☞ 下一事件推进方法(时间的异步模型，asynchronous model of time)
  - *时间推进时从当前事件发生时间到下一事件发生时间，也就是说，仿真会跳过没有事件发生的时间段*



如何推进仿真时间?

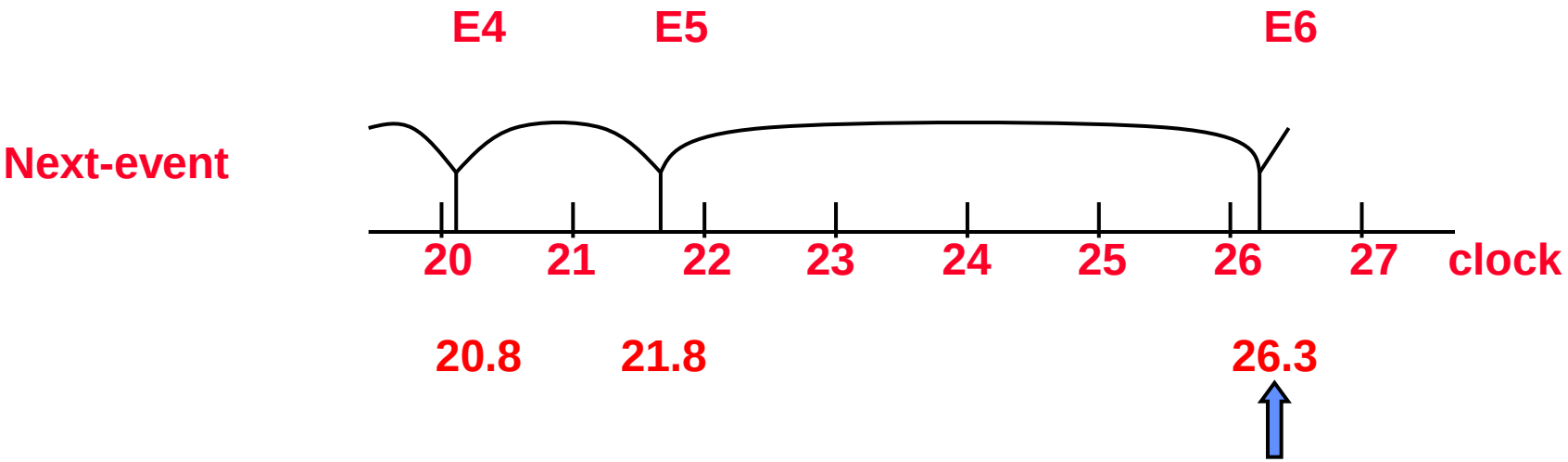
☞下一事件推进方法(时间的异步模型)

- *时间推进时从当前事件发生时间到下一事件发生时间，也就是说，仿真会跳过没有事件发生的时间段*



如何推进仿真时间?

- ☞ 下一事件推进方法(时间的异步模型，asynchronous model of time)
  - *时间推进时从当前事件发生时间到下一事件发生时间，也就是说，仿真会跳过没有事件发生的时间段*

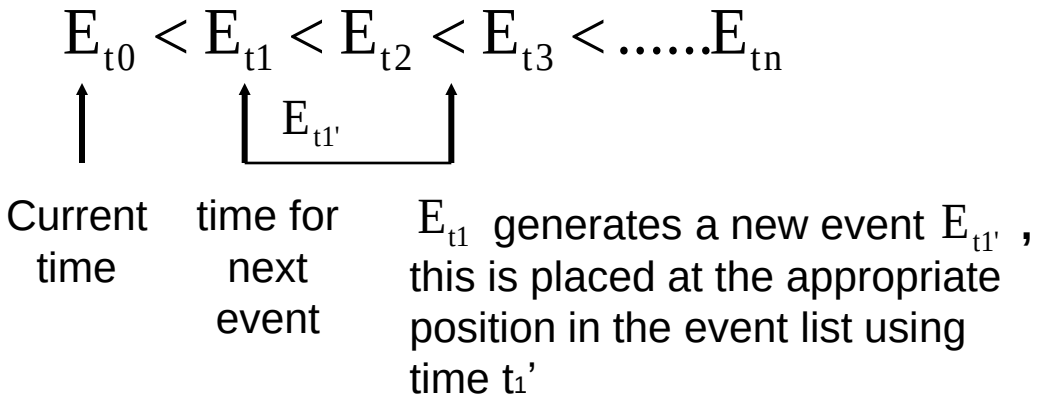


下一事件时间推进

- 推荐使用的方法
- 时间推进从当前时间到下一事件
- **仿真会跳过没有事件的阶段. 节省了计算机的运行时间，其代价是增加的编程难度.**
- 事件推进时，各模型异步执行，与时间步长推进的同步执行相反

未来事件列表(FEL)

- 异步的事件推进与同步的时间推进都假设所有事件是按时序发生的
- 一个事件日历 (也叫未来事件列表FEL,事件日记等)，包含所有将要调用的事件，并且以时序排列．在仿真器中，FEL一般用链表或树实现。



执行过程(Executive Routine)

- 时间/事件管理的基本方法- 执行过程 (也称作执行监控，调度器)
- 取出下一事件，推进仿真时间并且将控制权交给正确的执行过程 (事件，活动，进程)
- 执行过程的思路依赖于相应地建模视图(后面介绍).
- 不同建模视图的区别主要是下一事件选择的策略，以及不同类型的操作方法，比如事件、活动与进程。

下一事件时间推进算法(Next-event time advance algorithm)

- 数据结构: 未来事件列表 (FEL), 状态向量(SV), 时钟(CL)
- 伪代码:
  - ***Repeat***
    - ***remove smallest event (with time t) from FEL***
    - ***set CL=t***
    - ***process the event (and modify the SV)***
  - ***End_of_simulation***

对排队系统的仿真

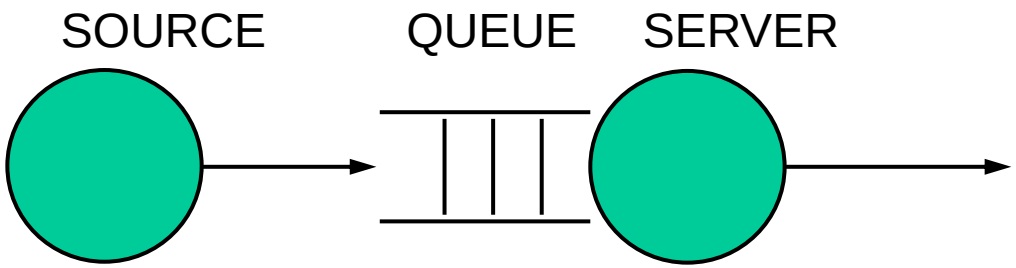
排队模型为什么特别重要?

系统	顾客	服务器
医院	**Patients**	**Nurses**
计算机	**Jobs**	**CPU, disk drive**
电话	**Calls**	**Exchange**
地铁	**Passengers**	**trains**
机场	**Airplanes**	**Runway**
酒店前台	**People**	**receptionist**
维修处	**Machines**	**repairperson**

例：单服务器排队模型(*M/M/1*)

☞ *衡量指标*:

- 平均等待时间(average delay in the queue):
  - 总延迟(total_of_delays)/ 被延迟的顾客数(num_custs_delayed)
- 队列中的平均顾客数(average number of customers in queue, q(n)):
  - 队列面积(area_num_in_q)/当前时间(current_time);
  - 队列面积(area_num_in_q) += 排队人数(num_in_q) * 排队时间(time_since_last_event);
- 服务器利用率(utilisation of server, u(n)):
  - 服务器状态面积(area_server_status)/当前时间(current_time);
    area_server_status += server_status * time_since_last_event;
- 所考察的仿真时间范围(Time simulation ended):
  - 当前时间(current_time)



A Single-Queue Single-Server System

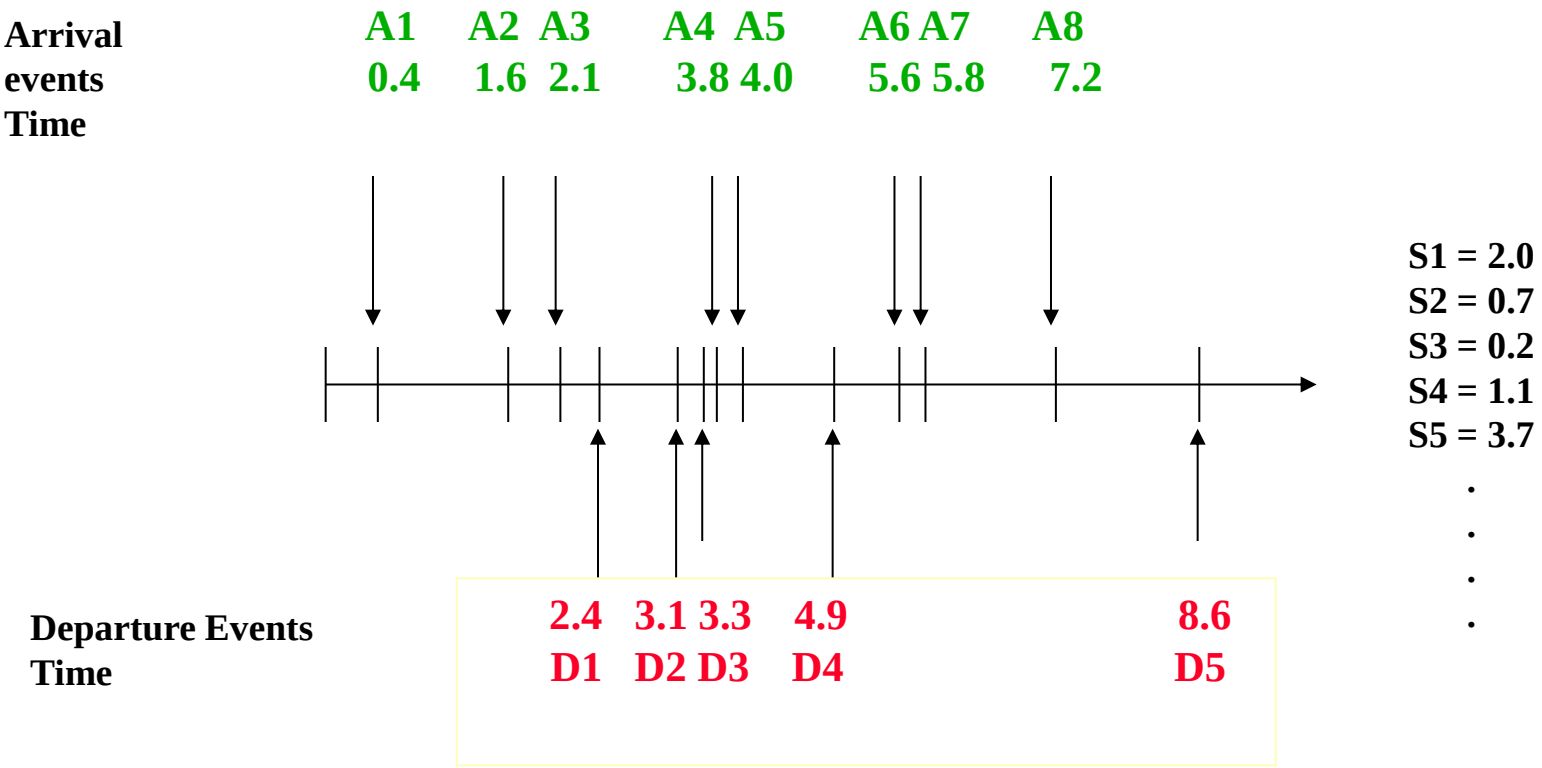
单服务器排队系统的建模与仿真

- 建模方面
  - *顾客的到达*
    - ***Inter-arrival times***
  - *服务策略*
    - ***First-in-first-out (FIFO)***
  - *中止规则*
    - ***when n customers receive service***
- 仿真方面
  - *时间推进机制*
    - ***Next-event time advance***
  - *事件类型:*
    - ***A: Arrival of a customer to the system***
    - ***D: Departure of a customer***
  - *事件执行方法:*
    - ***one per event type***

Example

Arrival times: 0.4, 1.6, 2.1, 3.8, 4.0, 5.6, 5.8, 7.2 …

Service times: 2, 0.7, 0.2, 1.1, 3.7...



三种对世界建模的视图

- event scheduling - 关注事件以及它们如何影响系统状态
- process interaction - 关注实体以及它所经历的事件/活动序列.
- activity scanning - 既使用调度时间也使用条件来选择事件列表中的下一事件

IMPORTANT

All three approaches use the concept of event to advance the simulation clock.

进程交互方法

- 一个进程要处理实体在系统中流动时发生的所有事件（包括确定事件和条件事件）。例如：

customer process

joins QUEUE, waits in QUEUE, receives service, leaves system

server process

waits for customer, services customer

- 每个实体可能在进程中的不同时间点上被阻塞或延迟。

过程交互方法

☞进程交互法的调度执行包括两类时序的事件列表:

1. *未来事件列表，future events list (FEL)* ***- contains processes scheduled at a future clock time***

2. *当前事件列表，current events list (CEL)* ***- contains processes eligible to be executed at the present time but are blocked at some stage in their process simulation and are waiting in the queue.***

过程交互方法

☞调度函数是这样执行的:

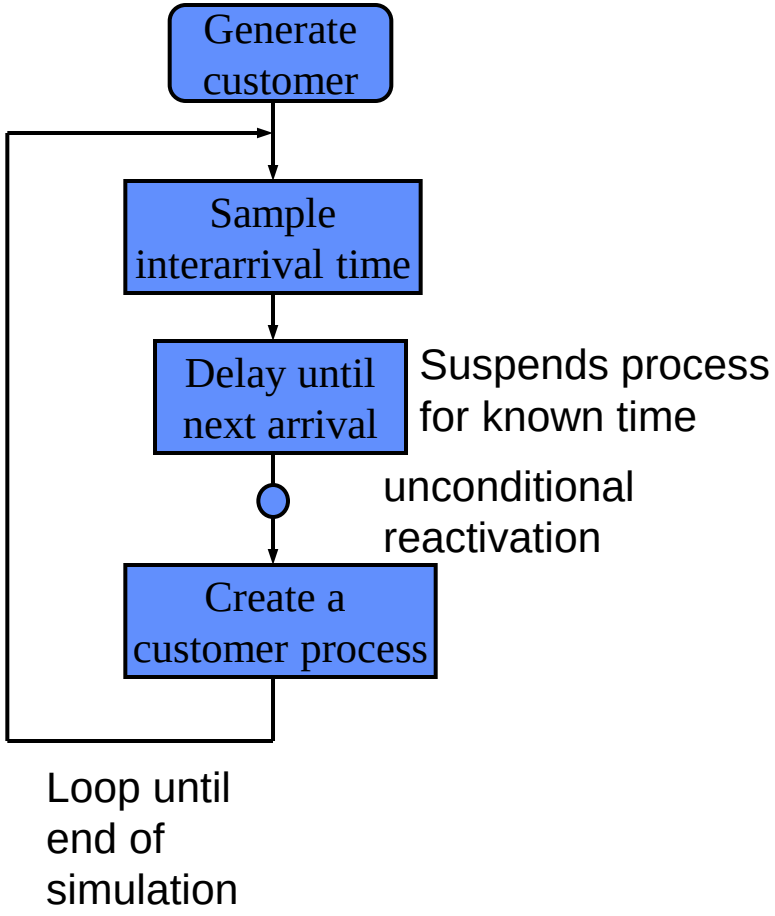
1. ***advance clock to the occurrence time of the next event (FEL)***

2. ***all events scheduled for the current time are moved from the FEL to the CEL***

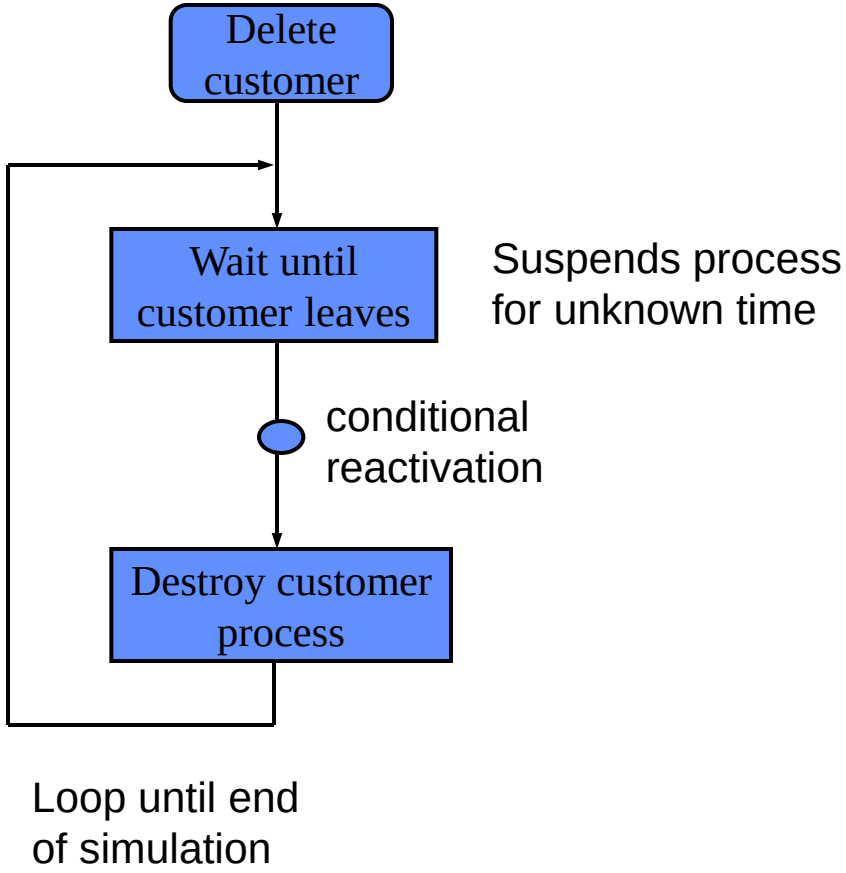
3. ***scan CEL to see if any entities can execute the next step in their process***

顾客生成/删除过程

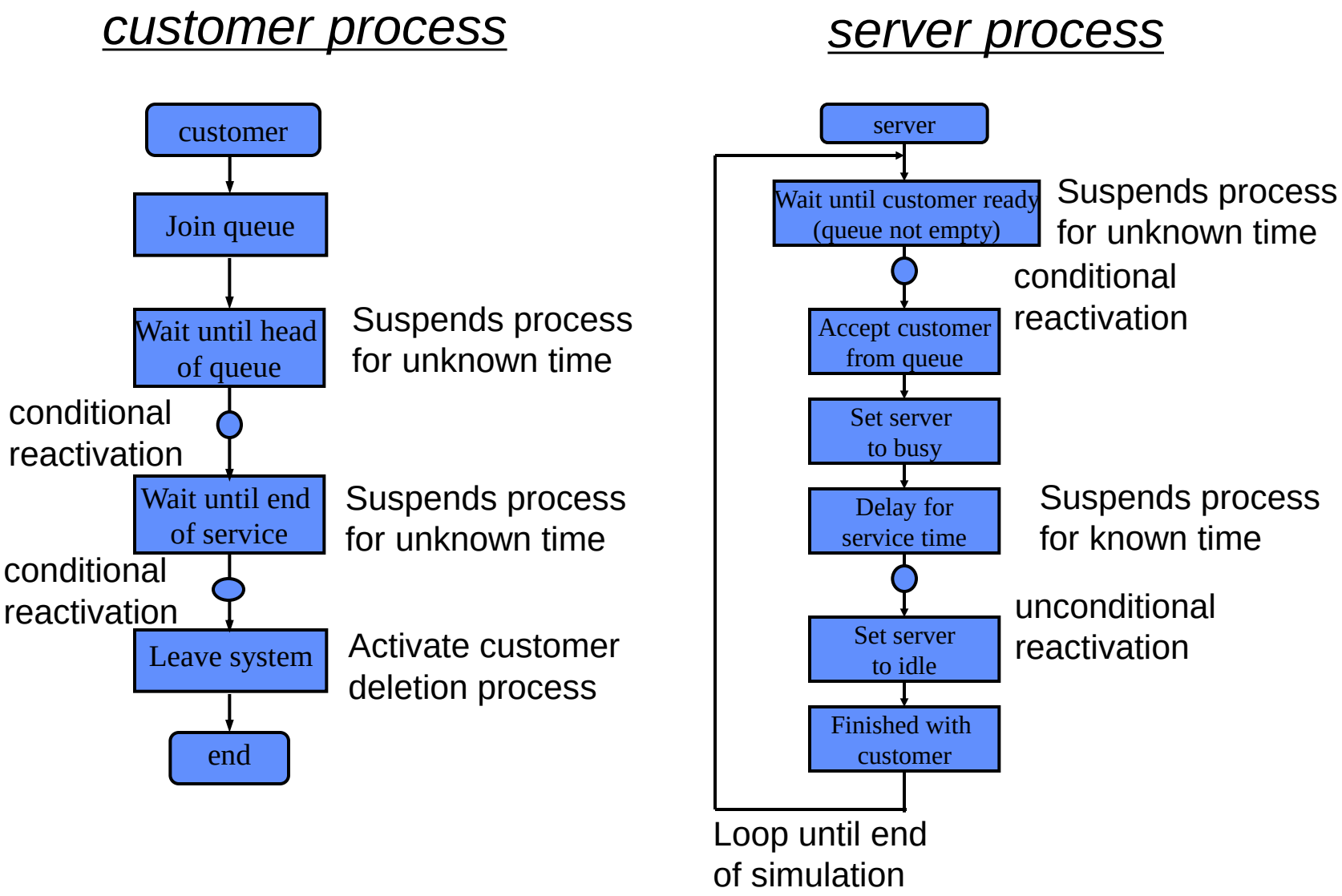
Customer generation process



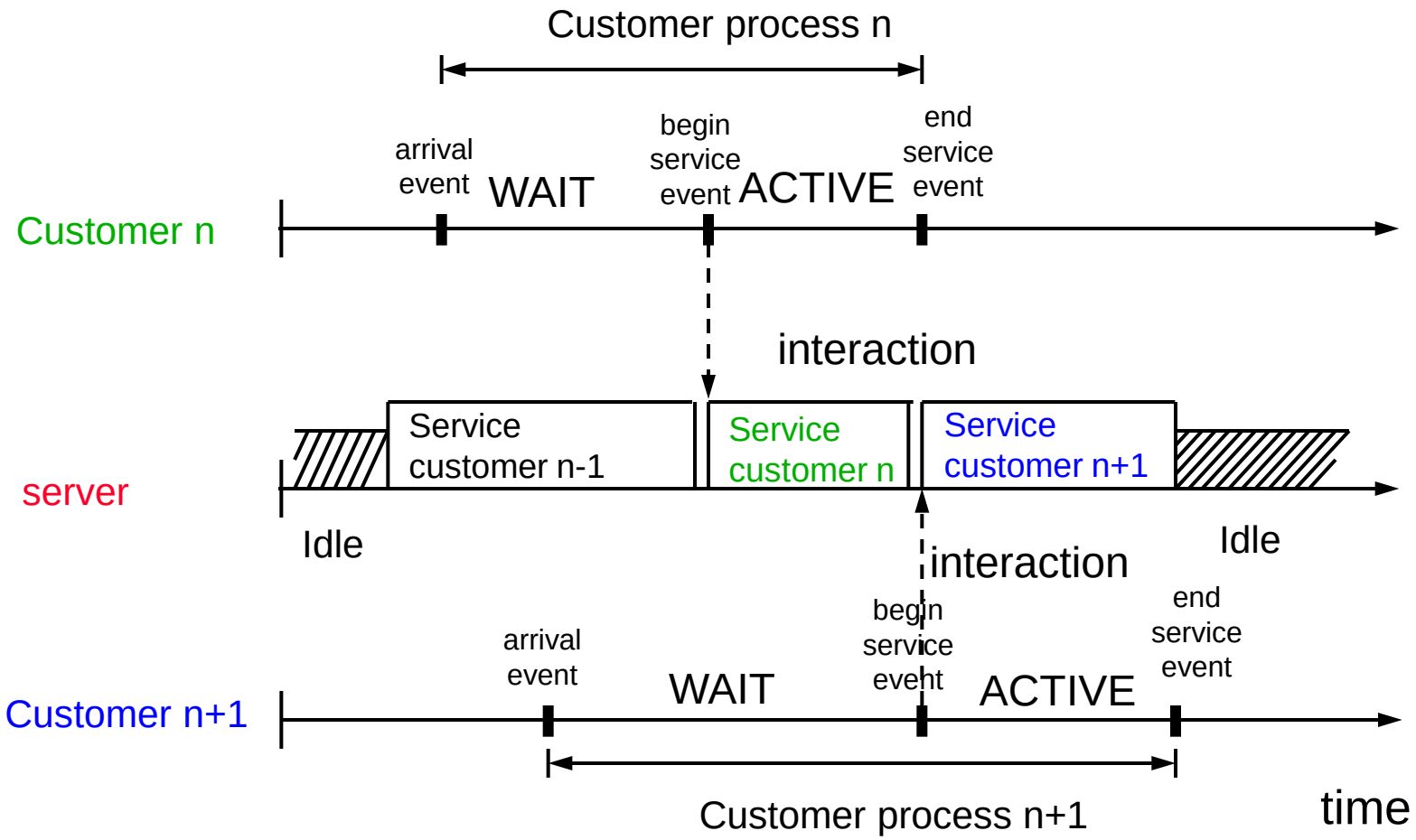
Customer deletion process



顾客/服务器过程


customer process
server process
customer
Join queue
Wait until head of queue
Suspends process for unknown time
conditional reactivation
Wait until end of service
Suspends process for unknown time
conditional reactivation
Leave system
Activate customer deletion process
end
server
Wait until customer ready (queue not empty)
Suspends process for unknown time
conditional reactivation
Accept customer from queue
Set server to busy
Delay for service time
Suspends process for known time
unconditional reactivation
Set server to idle
Finished with customer
Loop until end of simulation

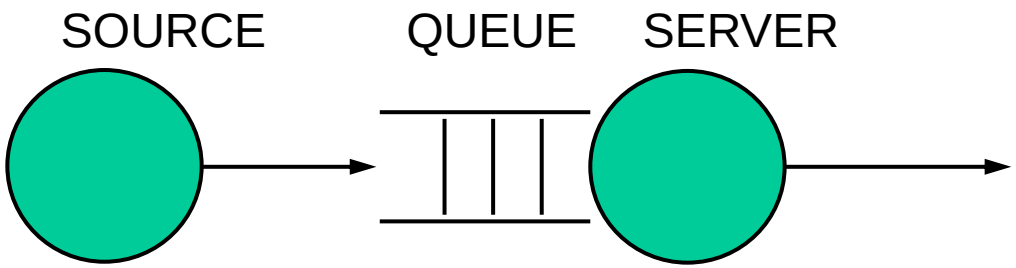
单服务器排队系统的交互过程



例：单服务器排队模型(*M/M/1*)

☞ *衡量指标*:

- 平均等待时间(average delay in the queue):
  - 总延迟(total_of_delays)/ 被延迟的顾客数(num_custs_delayed)
- 队列中的平均顾客数(average number of customers in queue, q(n)):
  - 队列面积(area_num_in_q)/当前时间(current_time);队列面积(area_num_in_q) += 排队人数(num_in_q) * 排队时间(time_since_last_event);
- 服务器利用率(utilisation of server, u(n)):
  - 服务器状态面积(area_server_status)/当前时间(current_time); area_server_status += server_status * time_since_last_event;
- 所考察的仿真时间范围(Time simulation ended):
  - 当前时间(current_time)



A Single-Queue Single-Server System

进程交互法 实例 - 牙科诊所[1]

- **1. 以主函数诊所(Clinic)的角度, 生成队列(Queue)中所有顾客(patient)到达的时间,对应所有顾客到达的事件;**

```
Patient::Patient()
{
   showta = 6 + rand()%4;//每位顾客在上一位顾客到达6~9分钟后到达;
   treatt = 5 + rand()%7;//每位顾客需要5~11分钟治疗时间
   showt = 0;
}
```

```
void Queue::Next_P_Arrive(Patient *patient0, int i)
{
    showtime += patient0[i].showta; //从0叠加每一位顾客在上一位顾客之后所需的到达时间
    patient0[i].showt = showtime;//记录每一位顾客自己在事件轴上的到达时间

}
```

进程交互法 实例 - 牙科诊所[2]

- 2. 以诊所(Clinic)的角度, 以超过出诊时间(2小时)作为仿真结束标志,记录最后一个顾客的序号j;
- 3. 开始for循环(i<j)依次处理每个顾客(单服务台排队系统中只需这样就可以描述所有的事件);
- 4. 如果服务台不忙,

```
void main()
{
   srand(time(NULL));//保证每次都是新的随机数

  Patient patient[20];
  Queue queue;
  for (i = 0; i < 20; i++)
  {
    queue.Next_P_Arrive(patient, i);
    if (queue.showtime > 120)
      {  j = i;
      break;
      }
   }
}
```

从0叠加每一位顾客在上一位顾客之后所需的到达时间

记录每一位顾客自己在事件轴上的到达时间

```
void Queue::Next_P_Arrive(Patient *patient0, int i)
{
  showtime += patient0[i].showta;
patient0[i].showt = showtime;
}
```

- **4. 如果服务台不忙，**
- 当前顾客开始被治疗的时间。因为诊所不忙碌，所以他一到达就接受治疗
- 判断的意义：这位顾客治疗完的时间若大于下位顾客到达时间，那么诊所呈忙碌状态

```
//开始义诊
for (i = 0; i < j; i++){
queue.Next_P_Treated(patient, i);
}
```

```
//判断是否需要等待及被治疗的过程
void Queue::Next_P_Treated(Patient *patient0, int i)
{
  if ( busy == 0 )//不忙碌
  {
     realtime = patient0[i].showt;//时间轴：
     if (i > 0)
       cout << "第" << i << "位顾客没有蛀牙了！用时：" 
<< patient0[i - 1].treatt <<"分钟。"<< endl;

    cout << "第" << i+1 << "位顾客正在接受治疗" << endl;
    cout << "他是在第 " << patient0[i].showt<<"分钟到达的。
"<<endl;
   cout << "他在第" << realtime << "分钟开始接受治疗" << endl;

  if ((realtime + patient0[i].treatt) > patient0[i+1].showt)
  {
       busy = 1;
        realtime += patient0[i].treatt;//下位顾客接受治疗
的时间
  }
       cout << endl << endl << endl;
}
```

进程交互法 实例 - 牙科诊所[4]

- 4. 如果服务台忙，
- 判断的意义：这位顾客开始接受治疗的时间是上一位顾客接受完治疗的时间；这位顾客治疗完的时间若大于下位顾客到达时间，那么诊所呈忙碌状态。

```
else
{
    cout << "第" << i+1 << "位顾客正在候诊室等待，兄弟们加把劲！"<<endl;
   cout << "第" << i << "位顾客没有蛀牙了！用时："<<patient0[i-1].treatt<<"分钟。"<<endl;
   cout << "第" << i + 1 << "位顾客正在接受治疗" << endl;
   cout << "他是在第 " << patient0[i].showt << "分钟到达的。" << endl;
   cout << "他在第" << realtime << "分钟开始接受治疗。" << endl<<endl<<endl;

  if ((realtime + patient0[i].treatt) > patient0[i + 1].showt)//
{
      busy = 1;
      realtime += patient0[i].treatt;
  }

   else busy = 0;//治疗完下位顾客还没来，所以呈空闲状态
}
```

进程交互法 实例 - 牙科诊所

- 牙科诊所例程的*问题*
- 1. 生成所有顾客到达的时间，随机数是均匀分布的，不是指数分布；
- 2. 如果是指数分布，应该输入λ；这是变量，可以每次仿真不一样；同样，平均服务时间也应该类似输入；
- 3. 结果缺少相关的统计计算，比如：平均等待时间、队列中的平均顾客数、服务器利用率；

活动扫描方法

- 使用固定时间间隔和基于规则的方法来决定是否开始一个活动**(activity)**
- 建模者关注模型的活动及触发条件
- 在每一个时间步长，检查条件，如果条件为真，触发相应活动（实际上就是时间步长推进）
- 完全的活动扫描是费时费力的，程序效率低
- 通常与事件调度法相结合

活动扫描方法

- 三段扫描法, **where we divide activities**
  - ***B** 类活动描述确定事件*
  - ***C** 类活动描述条件事件*
- **B** 类活动在 **FEL**中在某一时刻必定被执行
- 在每一次时间推进，检查是否有**B**类活动需要执行
- 然后检查**C**类活动并执行
- 重复此过程直到没有事件
  - ***Phase 1: remove event from FEL (advance time to event time), and all events at same event time***
  - ***Phase 2: execute these events***
  - ***Phase 3: scan conditions and activate those C activities that can be started***

事件调度方法

在单服务队列中，只有两类事件:

1. ***arrival***
2. ***departure or service completion***

状态: 队列长度, 服务器的状态

The event scheduling approach consists of the following two phases:

1. *把时钟推进到下一事件发生的时间,*
2. *认为下一事件发生并执行它.*

事件调度算法

main (executive routine):

1. ***set CLOCK = 0***
2. ***set cumulative statistics to 0***
3. ***define initial system state (queue empty, server idle)***
4. ***generate the occurrence time of the first arrival and place in future events list***
5. ***select the next event on the future events list (arrival or departure event)***
6. ***advance simulation CLOCK to time of next event***
7. ***execute the next event (arrival or departure event)***
8. ***if not terminating condition for simulation run goto step 5***

事件调度算法

arrival event

1. **generate the occurrence time of the next arrival and place arrival event in future events list**

2. **if SERVER is idle then**

   set server to busy

   update server idle time statistics

   generate the occurrence time of the departure event and place

   departure event in future events list

   else (* server busy *)

   place the customer in QUEUE

   update QUEUE length statistics

事件调度算法

departure event

1. destroy the current event

2. if QUEUE is empty then
set server to idle
update SERVER idle time statistics
else
remove the customer from QUEUE
update QUEUE waiting time statistics
generate the occurrence time of the departure event and place departure event in future events list

小 结

- 离散事件仿真
- 对于动态仿真,
  - 需要时钟来对模型的推进进行仿真
  - 两种时间推进的方式
  - 三种建模方法（视图）
  - 离散事件仿真的代码示例

小结

- 随机变量及其分布
- 均匀分布
- 泊松分布
- 指数分布
- 随机数生成

本讲作业

大作业：编程实现队列模型（M/M/1）的仿真程序，

- 输入参数：平均到达时间，平均服务时间，顾客数目，队列最大长度；
- 输出参数：队列中平均等待客户数；平均等待时间；服务器利用率。

要求提交程序和报告。

songxiao@buaa.edu.cn

F306

COMPONENTS AND ORGANISATION OF DISCRETE-EVENT SIMULATION

Components:

- system state
- simulation clock
- event list
- statistical counters
- initialization routine
- timing routine
- event routines
- library routines
- report generator
- main program

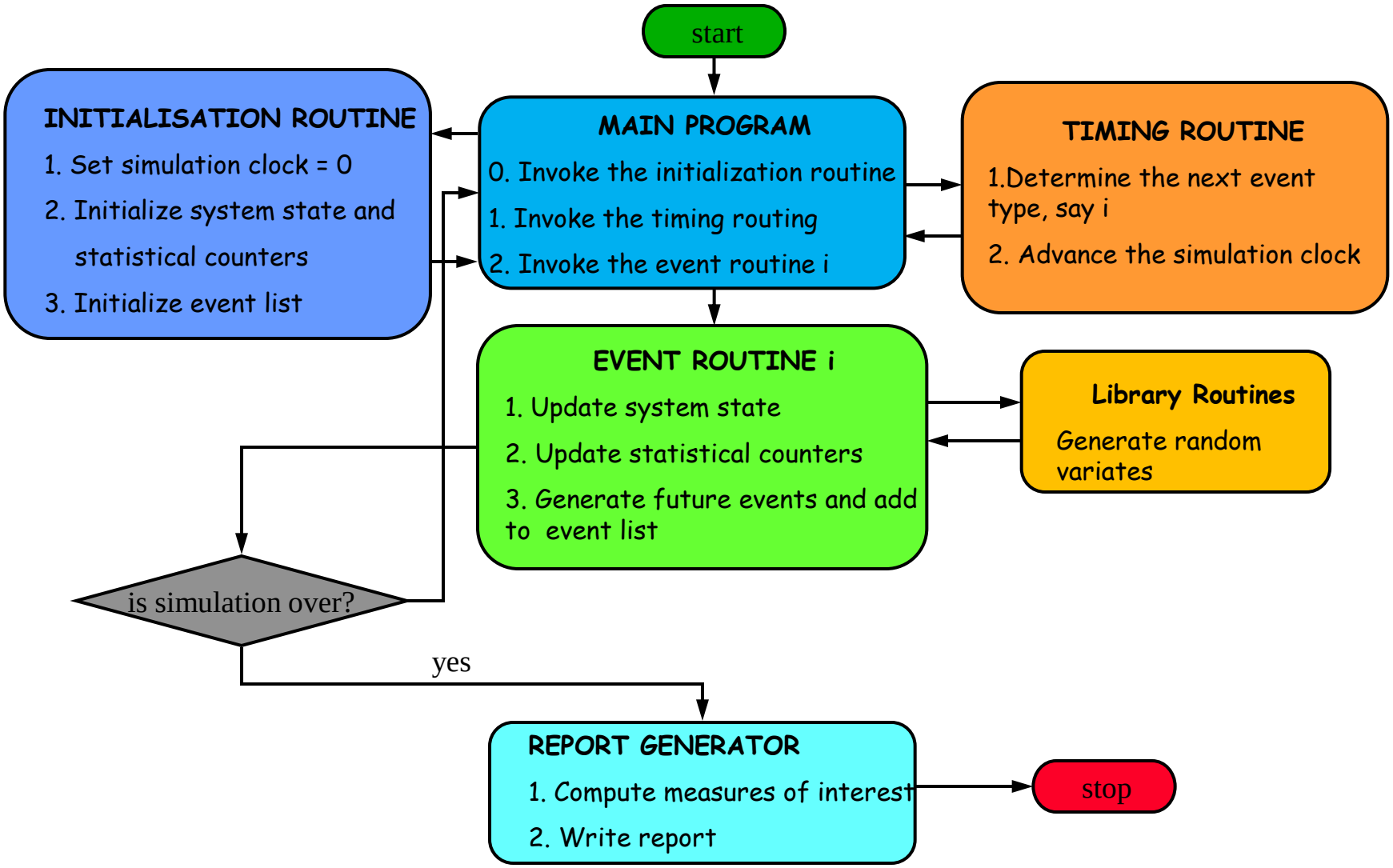
COMPONENTS & ORGANISATION of a DES

- SYSTEM STATE:
  - *the collection of state variables necessary to describe the system at a particular time.*
- SIMULATION CLOCK:
  - *a variable giving the current value of simulated time .*
- EVENT LIST:
  - *a list containing the next time when each type of event will occur*
- Statistical Counters:
  - *Variables used for storing statistical information about system performance*
- Initialisation Routine:
  - *A subprogram to initialise the simulation model at time zero.*
- Timing Routine:
  - *A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur.*

COMPONENTS & ORGANISATION of a DES

- Event Routine:
  - *a subprogram that updates the system state when a particular type of event occurs (one event routine for each event type)*
- Library Routines:
  - *a set of subprograms used to generate random observations from probability distributions that were determined as part of the simulation model*
- Report Generator:
  - *a subprogram that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends*
- Main Program:
  - *a subprogram that invokes the timing routine to determine the next event and then transfers control to the corresponding event routine to update the system state appropriately. The main program may also check for termination and invoke the report generator.*

Organisation of a Discrete-Event Simulator



Important Steps in Simulation and Model Building

1. define an achievable goal

   Goals change with increasing insight.

2. put together a complete mix of skills on the team

   We need:

   - ***knowledge of the system under investigation***
   - ***system analyst skills (model formulation)***
   - ***model building skills (model programming)***
   - ***data collection skills***
   - ***statistical skills (input data representation)***
   - ***more statistical skills (output data analysis)***
   - ***even more statistical skills (design of experiment)***
   - ***management skills (to get everyone pulling in the same direction)***

Important Steps in Simulation and Model Building

3. involve the end-user
   - ***Does anyone believe the results?***
   - ***Will anyone put the results into action?***
   - ***The end-user (your customer) can (and must) do all of the above***
   - ***BUT first, he must be convinced!***
   - ***He must believe it is HIS model.***

4. choose the appropriate simulation tools

   3 alternatives exist:

   - ***build model in a general purpose language***
     - ***Little cost, start from scratch, little reusable code, e.g. java, c***
   - ***build model in a general simulation language***
     - ***Readable code, moderate cost, e.g. GPSS, SIMSCRIPT, SIMAN V***
   - ***use a special purpose simulation package***
     - ***Minimal programming, can be costly, e.g. Arena, AutoMod, Simul8***

Important Steps in Simulation and Model Building

5. model the appropriate level(s) of detail
   - ***define the boundaries of the system***
   - ***some characteristics of "the environment" (outside the boundaries) may need to be included***
   - ***control the tendency to model in great detail well understood elements in the system, while skimming over other, less well understood sections***
6. start early to collect the necessary input data
   - ***TOO MUCH DATA:*** ***we need techniques for reducing it to a form usable in our model***
   - ***TOO LITTLE DATA:*** ***we need information which can be represented by statistical distributions.***

Important Steps in Simulation and Model Building

7. provide adequate and on-going documentation
   - ***In general, programmers hate to document. (They love to program!)***
   - ***What we can do?***
     - ***use self-documenting language***
     - ***insist on built-in user instructions (help screens)***
     - ***set (or insist) standards for coding style***
8. develop a plan for adequate model verification
   - ***Did we get the ”right answers”? (NO SUCH THING!)***
   - ***Simulation provides something that no other technique does:***
     - ***Step by step tracing of the model execution.***
     - ***This provides a very natural way of checking the internal consistency of the model.***

Important Steps in Simulation and Model Building

9. develop a plan for model validation
   - ***"Doing the right thing".***
   - ***How do we know our model represents the system under investigation?***

10. develop a plan for statistical output analysis
   - ***How much is enough? Long runs versus replications***
   - ***Techniques for analysis - point estimate, interval estimate ...***

Summary

- **Discrete-Event Simulation**

- **For dynamic simulation,**
  - ***need a clock to model passage of time***
  - ***Two ways of advancing time***
  - ***Three modelling worldviews***
  - ***Ten important steps in Building a simulator***