Problem Description

A certain airport contains a single runway on which arriving aircraft must land. Once an aircraft is cleared to land, it will use the runway, during which time no other aircraft can be cleared to land. Once the aircraft has landed, the runway is available for use by other aircraft. The landed aircraft remains on the ground for a certain period of time before departing.

Define Goals:

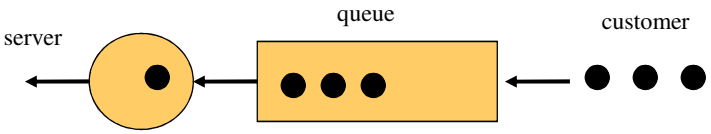
The objective is to determine

- The average time an aircraft must wait when arriving at an airport before they are cleared to land
- The maximum number of aircrafts that will be on the ground at one time

Problem Description

- The output metrics suggest focusing on
  - Waiting process
  - Number of aircraft on the ground
- We could develop a detailed model keeping track of the position of each aircraft every second, but this may be too much.
- Queuing models are a natural abstraction for modeling systems like these that include
  - Customers competing to use limited resources
  - Waiting (queuing) to use the resource
  - Primary metrics of interest have to do with resource utilization, time customer is being served or waiting
  - Details of what customer is doing while waiting are not important

Conceptual Model



- Customer (aircraft)
  - Entities utilizing the system/resources
- Server (runway)
  - Resource that is serially reused; serves one customer at a time
- Queue
  - Buffer holding aircraft waiting to land

Specification Model

Customers

Schedule of aircraft arrivals:

Often, probability distribution defines time between successive customer arrivals (interarrival time)

Assumes interarrival times *independent, and identically distributed*(iid)

Customer attributes? e.g., priorities

Servers

How much *service time* is needed for each customer?

May use probability distribution to specify customer service time (iid)

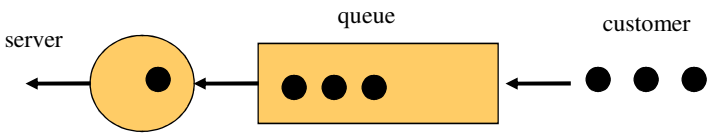
How many servers?

Queue

Service discipline - who gets service next?

- First-in-first-out (FIFO), Last-in-first-out (LIFO), random …
- May depend on a property of the customer (e.g., priority, "smallest" first)

State Variables (back to the ex.)



State:

- InTheAir: number of aircraft either landing or waiting to land
- OnTheGround: number of landed aircraft
- RunwayFree: Boolean, true if runway available

Events

- An event must be associated with any change in the state of the system
- Airport example:
  - Event for an aircraft arrival (`InTheAir`, `RunwayFree`)
  - Event for aircraft landing (`InTheAir`, `OnTheGround`, `RunwayFree`)
  - Event for aircraft departure (`OnTheGround`)

Fixed Increment Time

```
/* ignore aircraft departures */
Float InTheAir: # aircraft landing or waiting to land
Float OnTheGround: # landed aircraft
Boolean RunwayFree: True if runway available
Float NextArrivalTime: Time the next aircraft arrives
Float NextLanding: Time next aircraft lands (if one is landing)

For (Now = 1 to EndTime) { /* time step size is 1.0 */
   if (Now >= NextArrivalTime) /* if aircraft just arrived */
   {  InTheAir := InTheAir + 1;
      NextArrivalTime := NextArrivalTime + RandExp( μ);
      if (RunwayFree)
      { RunwayFree := False;
        NextLanding := Now + RandExp( λ);
      }
   }
   if (Now >= NextLanding) /* if aircraft just landed */
    { InTheAir := InTheAir – 1;
      OnTheGround := OnTheGround + 1;
      if (InTheAir > 0) NextLanding := Now + RandExp(λ)
      else {RunWayFree := True; NextLanding := EndTime+1;}
   }
}
```

Problems With Fixed increment time advance

- State changes may occur between time steps
  - Use small time steps to minimize error
- Multiple state changes within the same time step may be processed in the wrong order
  - Solvable by ordering state changes within time step (more work)
- Inefficient
  - Many time steps no state changes occur, especially if small time steps

The Example: Air traffic…

Single runway for incoming aircraft, ignore departure queueing

- L = mean time runway used for each landing aircraft (exponential distrib.)
- G = mean time on the ground before departing (exponential distribution)
- A = mean interarrival time of incoming aircraft (exponential distribution)

State:

- Now: current simulation time
- InTheAir: number of aircraft landing or waiting to land
- OnTheGround: number of landed aircraft
- RunwayFree: Boolean, true if runway available

Events:

- Arrival: denotes aircraft arriving in air space of airport
- Landed: denotes aircraft landing
- Departure: denotes aircraft leaving

Landing Event

An aircraft has completed its landing.

- L = mean time runway is used for each landing aircraft
- G = mean time required on the ground before departing
- Now: current simulation time
- InTheAir: number of aircraft landing or waiting to land
- OnTheGround: number of landed aircraft
- RunwayFree: Boolean, true if runway available

```
Landed Event:
InTheAir:=InTheAir-1;
OnTheGround:=OnTheGround+1;
Schedule Departure event @ Now + RandExp(G);
If (InTheAir>0)
   Schedule Landed event @ Now + RandExp(L);
Else
   RunwayFree := TRUE;
```

Arrival Event

New aircraft arrives at airport. If the runway is free, it will begin to land. Otherwise, the aircraft must circle, and wait to land.

- `A:` mean interarrival time of incoming aircraft
- `Now:` current simulation time
- `InTheAir`: number of aircraft landing or waiting to land
- `OnTheGround`: number of landed aircraft
- `RunwayFree`: Boolean, true if runway available

```
Arrival Event:
InTheAir := InTheAir+1;
Schedule Arrival event @ Now + RandExp(A);
If (RunwayFree) {
  RunwayFree:=FALSE;
  Schedule Landed event @ Now + RandExp(L);
}
```

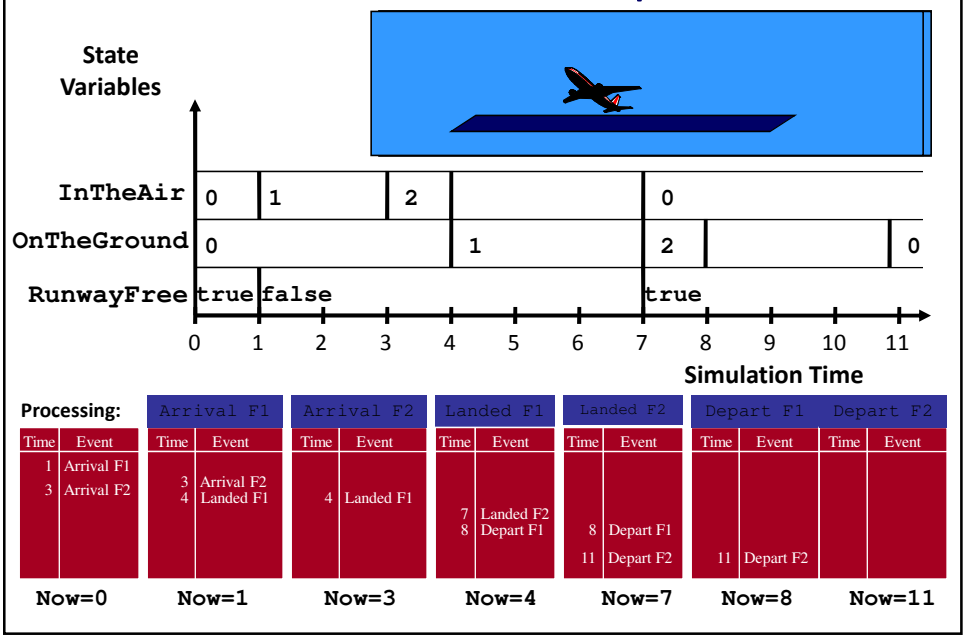
Departure Event

An aircraft now on the ground departs for a new destination.

- `Now:` current simulation time
- `InTheAir`: number of aircraft landing or waiting to land
- `OnTheGround`: number of landed aircraft
- `RunwayFree`: Boolean, true if runway available

```
Departure Event:
OnTheGround := OnTheGround – 1;
```

Execution Example

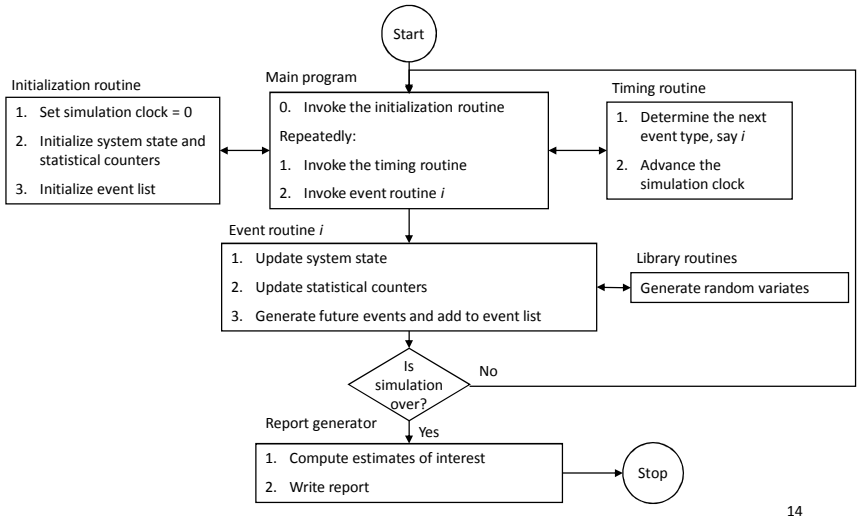


Processing:

Arrival F1		Arrival F2		Landed F1		Landed F2		Depart F1		Depart F2	
Time	Event	Time	Event	Time	Event	Time	Event	Time	Event	Time	Event
1	Arrival F1	3	Arrival F2	4	Landed F1	7	Landed F2	8	Depart F1	11	Depart F2
3	Arrival F2	4	Landed F1			8	Depart F1	11	Depart F2		

Now=0 **Now=1** **Now=3** **Now=4** **Now=7** **Now=8** **Now=11**

Discrete-Event Simulation Logical Flow



14

Discrete-Event Simulation Components

Initialization routine: A subprogram to initialize the simulation model at time 0

Timing routine: A subprogram that determines the next event from the event list and then advances the simulation clock to the time when the next event is to occur

Event routine: A subprogram that updates the system state when a particular type of event occurs (there is one event routing for each type of event)

Library routines: A set of subprograms used to generate **random observations from probability distributions** that were determined as part of the simulation model

15

Discrete-Event Simulation Components

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 routing 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 when the simulation is over.

16

Discrete-Event Simulation Stopping Rules

Number of events of a certain type reached a pre-defined value

Example: stop M/M/1 simulation after the 1000th departure

Simulation time reaches a certain value; this is usually implemented by scheduling an "end-simulation" event at the desired simulation stop time.

17

Output Statistics

Compute

- The maximum number of aircraft that will be on the ground at one time
- Average time an aircraft must wait before they are cleared to land

- Maximum on ground
  - Variable for airport indicating number currently on ground
  - Maximum "on the ground" so far
- Wait time
  - Variables for airport indicating total wait time, number of aircraft arrivals
  - State variable for each aircraft indicating arrival time

Input Analysis

- Stochastic simulation models use random variables to represent inputs such as inter-arrival times, service times, waiting time at an ATM machine…etc
- We need to know the **distribution** and **parameters** of each of these random variables.

Driving Simulation models

Some methods to do this:

- Collect real data, and feed this data to the simulation model. (**trace-driven simulation**.)
- Collect real data, build an **empirical distribution** of the data, and sample from this distribution in the simulation model.
- Collect real data, fit a **theoretical distribution** to the data, and sample from that distribution in the simulation model.
- Focus on the last two of these methods.

Why do we use Theoretical Distribution

- Theoretical distributions "smooth out" the irregularities that may be present in the *empirical* distributions.
- Gives the simulation the ability to generate wider range of values.
  - Test extreme conditions.
- Theoretical distributions are a compact way to represent very large datasets. Easy to change, very practical.

Describing Distributions

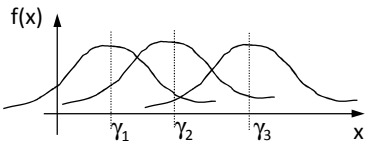
A probability distribution is described by its **family** and its **parameters**. The choice of family is usually made by examining the density function, because this tends to have a unique shape for each distribution family.

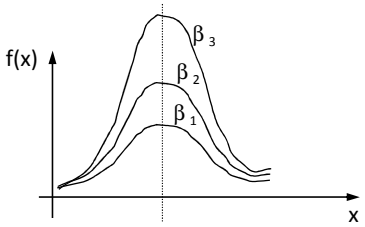
Distribution parameters are of three types:

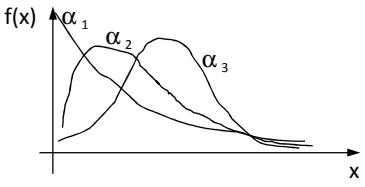
location parameter(s) γ

scale parameter(s) β

shape parameter(s) α







Selecting an Input Distribution - Family

- The 1st step in any input distribution fit procedure is to hypothesize a family (i.e. exponential).
- Prior knowledge about the distribution and its use in the simulation can provide useful clues.
  - Normal shouldn't be used for service times, since negative values can be returned.
- Mostly, we will use heuristics to settle on a distribution family.

Determining the "Goodness" of the Model

- As you might imagine, determine whether our hypothesize model is "good".
- Both heuristic and analytical tests exist to determine "goodness".
- Heuristic tests:
  - Density/Histogram over plots.
  - Frequency comparisons.
  - Distribution function difference plots.
  - Probability-Probability Plots (P-P plots).
- Analytical tests:
  - Chi-Squared tests.
  - Kolmogorov-Smirnov tests