# Discrete - time Event Simulation

1. Events
2. State
3. Clock

```
                              7:23
                               ↑
  |----0-----1----2--3-|-2---→ time
  7:00    ↑      ↑     ↑
       7:10   7:18   7:22
```
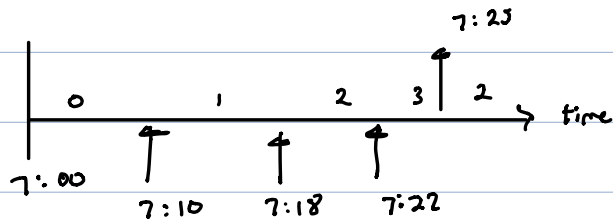
★ **Event queue**

Clock will go there immediatly
- State remains same
- Every second = discrete time
- Clock only updates when event happens

| 7:10 | Arr |  |

| 7:18 | Arr |  | → | 7:22 | Arr |  |

| 7:25 | Depart |  | → Back to original

Clock : 7:00
State : # of people in class = ∅.

Clock : 7:10
State : 1

Clock : 7:18
State : 2

Clock: 7:25
State: 3

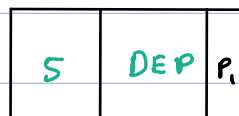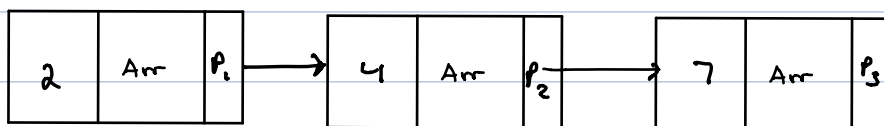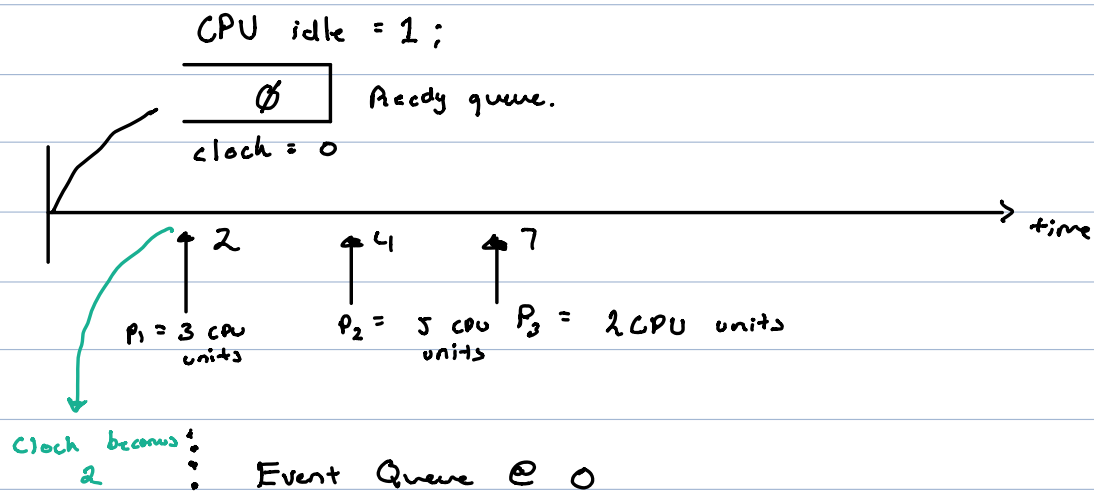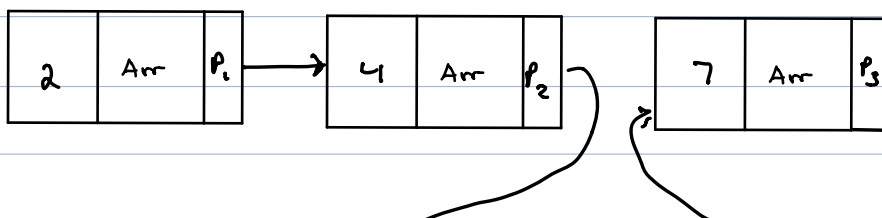Event Queue only holds events

1. Events → Change the state
2. State → Datastructures (eg. ready queue)
3. Clock → A variable (double)

__FCFS__   First come first serve

· 3 Arrivals

CPU idle = 1;

| ∅ |   Ready queue.

clock = 0



2     4     7 → time

$P_1 = 3$ CPU units    $P_2 = 5$ CPU units    $P_3 = 2$ CPU units

Clock becomes 2

Event Queue @ 0

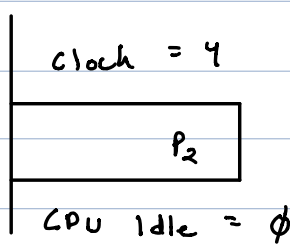| 2 | Arr | $P_1$ | → | 4 | Arr | $P_2$ | → | 7 | Arr | $P_3$ |

| 5 | DEP | $P_1$ |

⋮

Event Queue @ 2

| 2 | Arr | $P_1$ | → | 4 | Arr | $P_2$ | | 7 | Arr | $P_3$ |

Then:

Ready queue


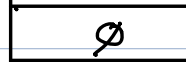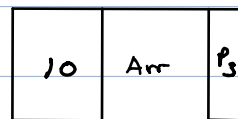
CPU Idle: $\phi$

Clock = 4

$P_2$

CPU Idle = $\phi$

Clock = 5;

CPU idle = 0;

$\phi$
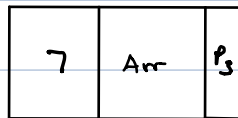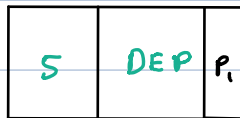


Block 5 now at

time 10
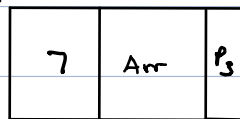
General Layout shown Above.



$T_s = 0.06$

16.66

$\rho = \dfrac{\lambda}{\mu} = \lambda * T_s$

At that point

beyond this point = overload

Arrival rate > Utilization
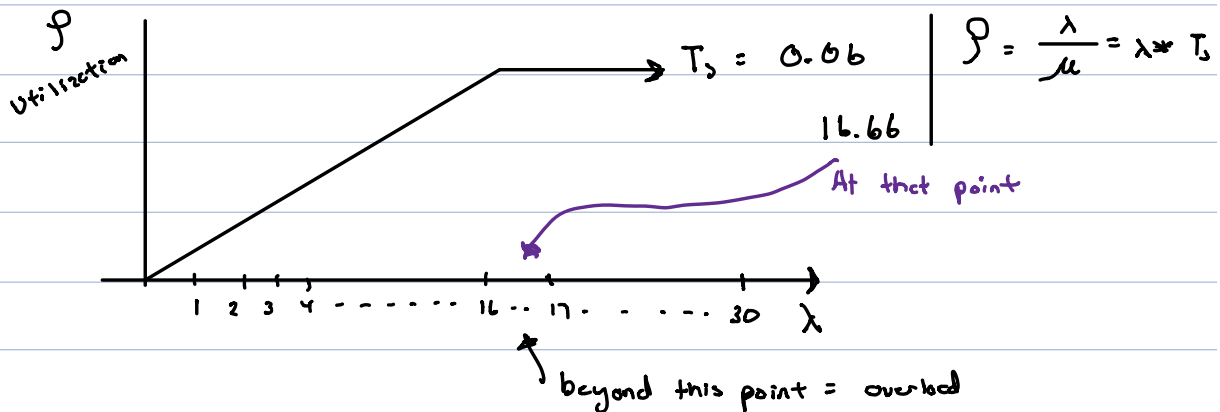
• Generate Arrival time

• Generate Service time
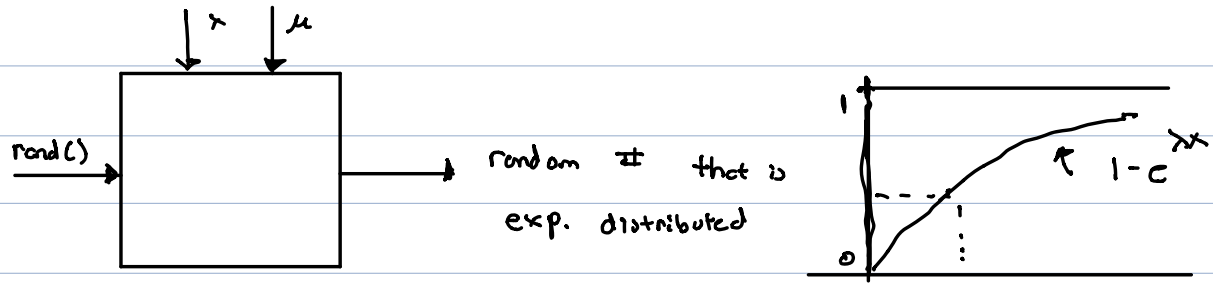
• Different lines for FCFS in terms of the data.

rand() → [ box with λ, μ inputs on top ] → random # that is exp. distributed

Graph: axis from 0 to 1, curve approaching $1 - e^{-\lambda x}$

## Function Call          Code

```
Process {
    int id;
    double service_time;
    double arrival_time;
    double remaing_time;
    double completion_time;
}
```

First Come First Serve