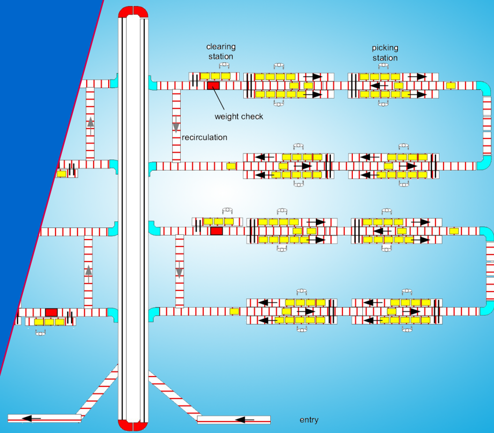
Stochastic Models of Manufacturing Systems

Ivo Adan



Technische Universiteit Eindhoven
University of Technology

Tuesday May 19

Systems

- *Continuous systems*
  State changes continuously in time (e.g., in chemical applications)
- *Discrete systems*
  State is observed at fixed regular time points (e.g., periodic review inventory system)
- *Discrete-event systems*
  The system is completely determined by random event times $t_1, t_2, \ldots$ and by the changes in state taking place at these moments (e.g., production line, queueing system)

Time advance

- Look at regular time points $0, \Delta, 2\Delta, \ldots$ (*synchronous* simulation); in continuous systems it may be necessary to take Δ very small
- Jump from one event to the next and describe the changes in state at these moments (*asynchronous* simulation)

We will concentrate on asynchronous simulation of discrete-event systems

Some common terms

- **System**
  Collection of objects interacting through time (e.g. production system)
- **Model**
  Mathematical representation of a system (e.g., queueing or fluid model)
- **Entity**
  An object in a system (e.g., jobs, machines)
- **Attribute**
  Property of an entity (e.g., arrival time of a job)
- **Linked list**
  Collection of *records* chained together

Some common terms

- **Event**
  Change in state of a system
- **Event notice**
  Record describing when event takes place
- **Process**
  Collection of events ordered in time
- **Future-event set**
  Linked list of event notices ordered by time (FES)
- **Timing routine**
  Procedure maintaining FES and advancing simulated time

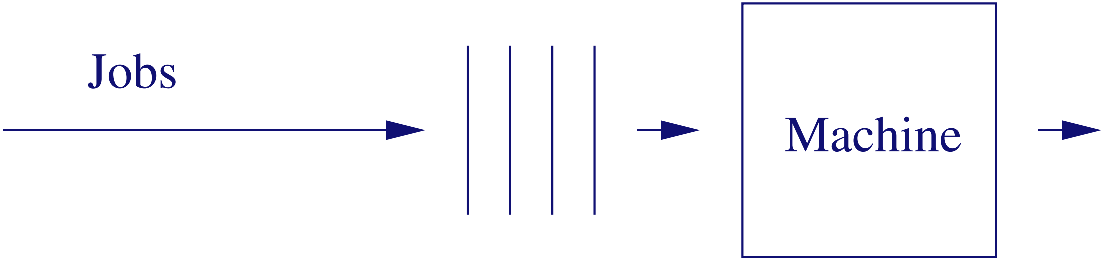
Basic approaches

- *Event-scheduling approach*
  Focuses on events, i.e., the moments in time when state changes occur
- *Process-interaction approach*
  Focuses on processes, i.e., the flow of each entity through the system

In general-purpose languages one mostly uses the event-scheduling approach; simulation languages (e.g., χ) use the process-interaction approach

Event-scheduling approach

Example: Single machine model



A single machine processes jobs in order of arrival. The interarrival times and processing times are exponential with parameters λ and μ (with $\lambda < \mu$).

- What is the mean waiting time?
- What is the mean queue length?
- How does the performance change if we speed up the machine?

Discrete simulation

A_n the interarrival time between job n and $n+1$

B_n the processing time of job n

W_n the waiting time of job n

Then (Lindley's equation):

$$W_{n+1} = \max(W_n + B_n - A_n, 0)$$

Program

```
model lindley():
    int n, N = 100;
    real w, sumw, a, b;

    while n < N:
        a = sample exponential(1.25);
        b = sample exponential(1.0);
        w = max(w + b - a, 0.0);
        sumw = sumw + w;
        n = n + 1
    end;

    writeln("E(W) = %g", sumw / N);
end
```

Discrete-event simulation

Entity	**Attribute**
Job	Arrival time
Machine	Status (idle or busy)

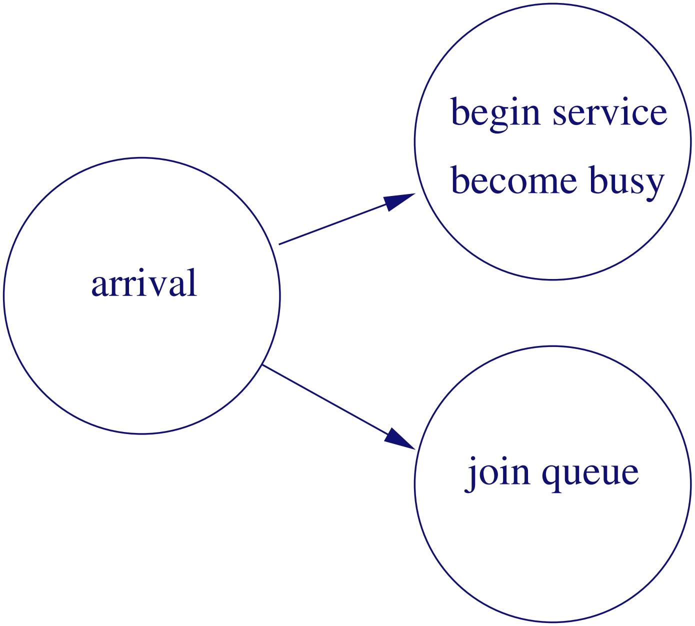
Job is a *temporary* entity
Machine is a *permanent* entity

Elementary events

Job:	**Machine:**
arrival	remove from queue
departure	become busy
begin service	become idle
end service	
join queue	

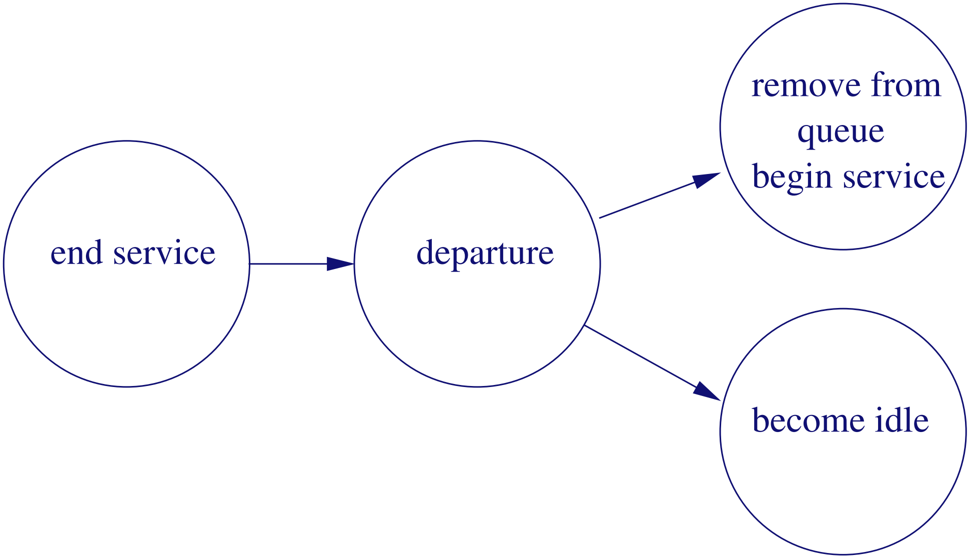
Compound events

Arrival



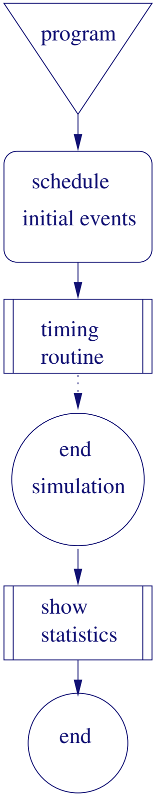
Compound events

Departure



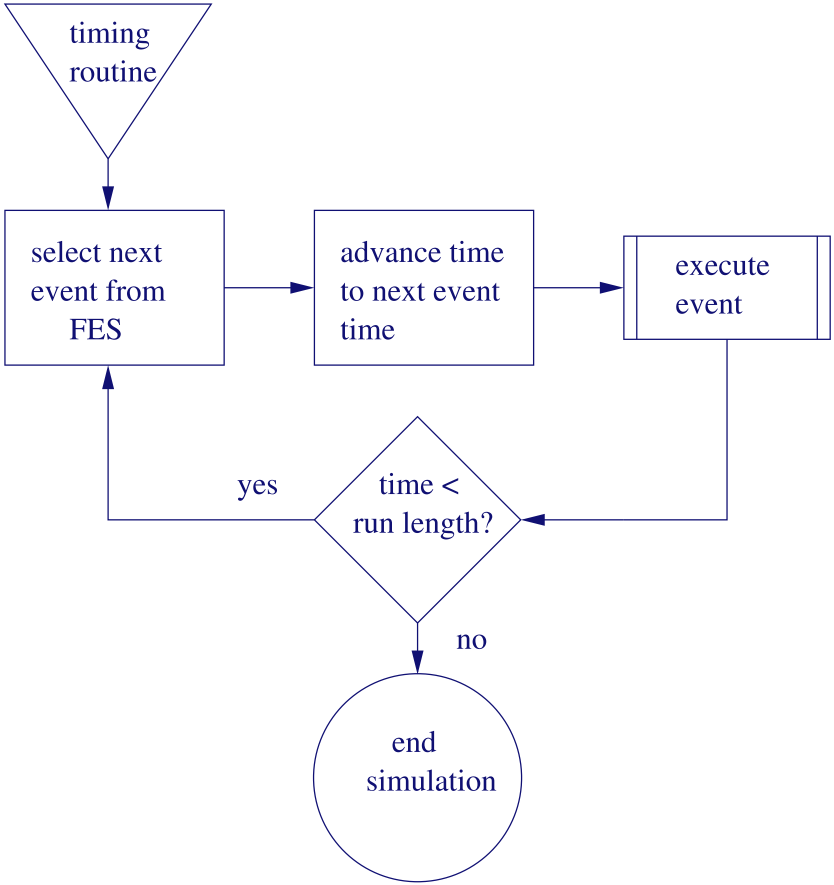
Technische Universiteit
Eindhoven
University of Technology

Prototypical approach

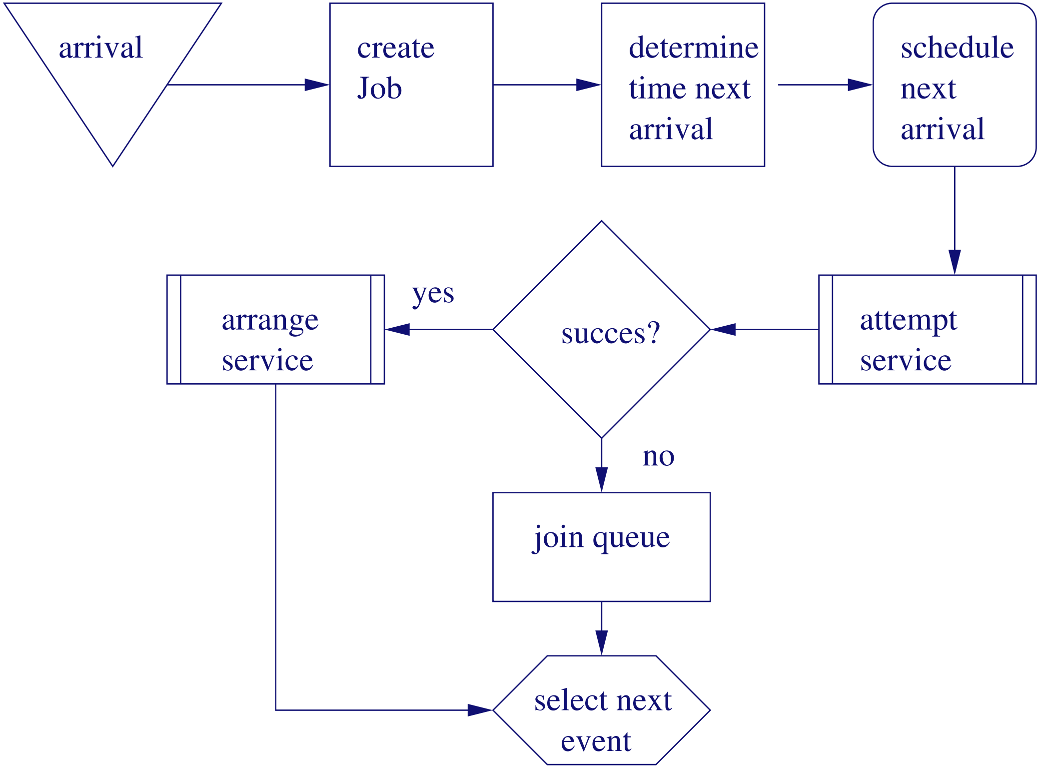


Technische Universiteit
Eindhoven
University of Technology

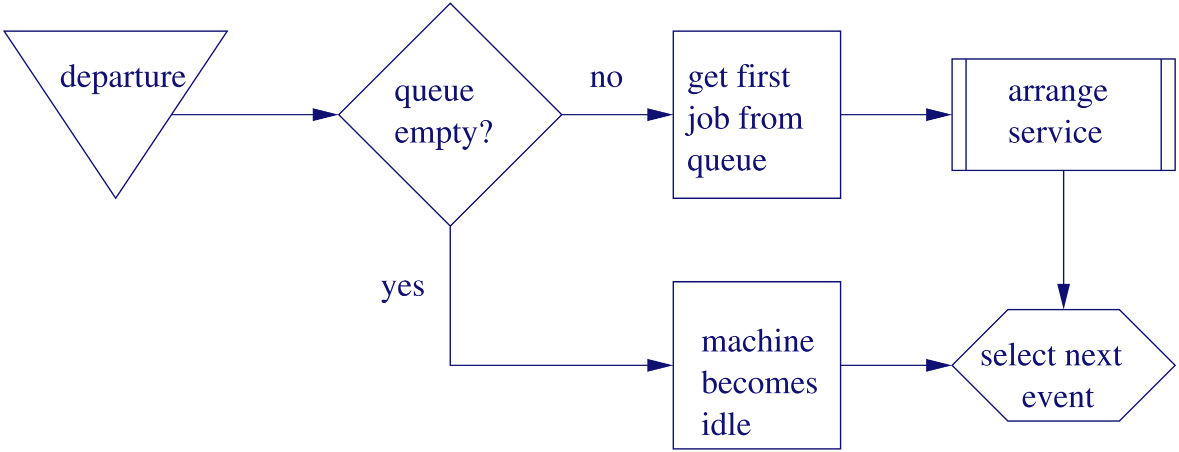
Prototypical approach



Arrival event



Departure event



Discrete-event simulation


```
type job = real;
type event = tuple (string e; real t);

func bool pred(event x, y):
    return real(x.t) < real(y.t)
end
```

Discrete-event simulation


```
model GG1():
    bool busy; string e; int n;
    real t, sumw;
    list job queue; list event fes;

    fes = [("a", t + sample exponential(1.25))];
    while n < 1000000:
        e, t = fes[0];
        fes = fes[1:];
        if e == "a":
            ...
        elif e == "d":
            ...
        else:
            writeln("error: unknown event");
            break;
        end;
    end;
    writeln("E(W) = %g", sumw / n);
```

Discrete-event simulation


```
if e == "a":
    if busy == false:
        busy = true;
        n = n + 1;
        fes = insert(fes,
              ("d", t + sample exponential(1.0)), pred);
    else:
        queue = queue + [t];
    end;
    fes = insert(fes,
          ("a", t + sample exponential(1.25)), pred);
```

Discrete-event simulation


```
elif e = "d":
    if size(queue) > 0:
        sumw = sumw + (t - queue[0]);
        queue = queue[1:];
        n = n + 1;
        fes = insert(fes,
              ("d", t + sample exponential(1.0)), pred);
    else:
        busy = false;
    end;
```

Proces-Interaction approach

This approach focusses on describing *processes*;
In the event-scheduling approach one regards a simulation as executing a sequence of events ordered in time; but *no time elapses* within an event.

The process-interaction approach provides a process for *each entity* in the system; and *time elapses* during a process.

In production systems we have processes for:

- Arrivals
- Buffers
- Machines
- Exit

Single machine model

- Generator G sends jobs to buffer B;
- Buffer B receives jobs from G and sends jobs to machine M;
- Machine M processes these jobs and sends finished jobs to exit E;
- Exit E is doing some book keeping.

Object type job


```
type job = real;
```

Generator G


```
proc G(chan! job a; dist real u):

    while true:
        a!time;
        delay sample u;
    end
end
```

G generates jobs with inter-arrival times sampled from distribution u.

Bufffer B


```
proc B(chan? job a; chan! job b):
    list job xs;
    job x;

    while true:
        select
            a?x:
                xs = xs + [x]
        alt
            size(xs) > 0, b!xs[0]:
                xs = xs[1:]
        end
    end
end
```

B receives, stores and sends jobs

Machine M


```
proc M(chan? job a; chan! job b; dist real u):
    job x;

    while true:
        a?x;
        b!x;
        delay sample u;
    end
end
```

M processes jobs with processing times sampled from distribution u.

Exit E


```
proc E(chan? job a; int n):
    int i;
    real sumw;
    job x;

while i < n:
        a?x;
        sumw = sumw + (time - x);
        i = i + 1;
    end;
    writeln("E(W) = %g", sumw / n);
end
```

Exit E computes mean waiitng time over first n jobs.

Single machine model $GBME$


```
model GBME():
chan job a, b, c;

run G(a, exponential(1.25)),
        B(a, b),
        M(b, c, exponential(1.0)),
        E(c, 1000)
end
```

Simulation system Arena

In Arena you can construct simulation models without programming, but simply with click, drag and drop...

Student version of Arena is available in the campus software

Book with CD-ROM:

W. David Kelton, Randall P. Sadowski, Deborah A. Sadowski:
Simulation with Arena. 2nd ed., London: McGraw-Hill, 2002

Output analysis

Method of independent replications

Example: Long-run ("steady-state") mean waiting time $E(W)$ in the single machine model

Produce n independent sample paths of waiting times $W_1^{(k)}, W_2^{(k)}, \ldots, W_N^{(k)}$ and compute

$$\overline{W}_N^{(k)} = \frac{1}{N} \sum_{j=1}^{N} W_j^{(k)}, \quad k = 1, \ldots, n.$$

Output analysis

Then, for large N, an approximate $100(1-\alpha)\%$ confidence interval for the mean waiting time $E(W)$ is

$$\overline{W}_N(n) \pm z_{1-\frac{1}{2}\alpha} \frac{S_N(n)}{\sqrt{n}}$$

where $\overline{W}_N((n)$ and $S_N^2((n)$ are the sample mean and variance of the realizations $\overline{W}_N^{(1)}, \ldots, \overline{W}_N^{(n)}$,

$$\overline{W}_N(n) = \frac{1}{n} \sum_{k=1}^{n} \overline{W}_N^{(k)}$$

$$S_N^2(n) = \frac{1}{n} \sum_{k=1}^{n} [\overline{W}_N^{(k)} - \overline{W}_N(n)]^2$$

Output analysis


```
xper X():
    int n;
    real w, sum1, sum2, smean, svar;

    n = 10;
    for i in range(n):
        w = GBME();
        sum1 = sum1 + w;
        sum2 = sum2 + w * w;
        writeln("E(W(\%d) = \%g", i, w)
    end;

    smean = sum1 / n;
    svar = sum2 / n - smean * smean;
    writeln("E(W) =  \%g +- \%g",
        smean, 1.96 * sqrt(svar / n));
end;
```

Output analysis

Results for $\lambda = 0.5$, $\mu = 1$ and 10 runs, each of $N = 10^4$ waiting times

k	$\overline{W}_N^{(k)}$
1	0.995
2	1.002
3	0.959
4	1.037
5	0.902
6	1.011
7	1.125
8	1.007
9	1.075
10	1.044

$E(W) = 1.016 \pm 0.036$ (95% confidence interval)

Output analysis

Results for $\lambda = 0.9$, $\mu = 1$ and 10 runs, each of $N = 10^4$ waiting times

k	$\overline{W}_N^{(k)}$
1	7.373
2	8.496
3	8.574
4	7.752
5	8.637
6	7.404
7	9.556
8	8.863
9	8.537
10	11.000

$E(W) = 8.619 \pm 0.632$ (95% confidence interval)

Clearly, a more congested system is harder to simulate!
To obtain a more accurate estimate we should increase the number of runs and/or the length of each run: How much?

Initialization effect

We are interested in the long-run behavior of the system and maybe the choice of the initial state of the simulation will influence the quality of our estimate.

One way of dealing with this problem is to choose N very large and to neglect this initialization effect. However, a better way is to throw away in each run the first m observations:

$$\overline{W}_N^{(k)} = \frac{1}{N-m} \sum_{j=m+1}^{N} W_j^{(k)}.$$

We call m the length of the warm-up period and it can be determined by a graphical procedure.

Disadvantage of the independent replication method:
Initialization effect in each simulation run.

Single machine model

- Arrival process of jobs:
  - Interarrival times are independent and identically distributed;
  - Jobs arrivals one-by-one.
- Processing times:
  - Processing times are independent and identically distributed;
- Processing order:
  - First come first served (FCFS);
  - Processing one-by-one.
- Processing capacity:
  - Single machine machine.
- Buffer capacity:
  - Ample (infinite).

Poisson arrival process

Interarrival times of jobs are independent and exponential with rate λ.

Because of memoryless property,

$$P(\text{arrival in } (t, t+\Delta)) = 1 - e^{-\lambda\Delta} \approx \lambda\Delta$$

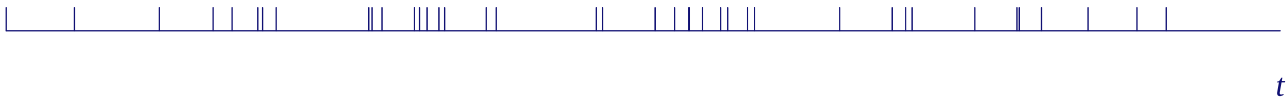
and whether or not there is an arrival is independent of any arrivals before t.

Hence, dividing $(0, t)$ into small intervals of length Δ, the number of arrivals in $(0, t)$ is binomial with $n = t/\Delta$ and $p = \lambda\Delta$. Since n is large and p is small, this number is Poisson distributed with parameter $np = \lambda t$,

$$P(k \text{ arrivals in } (0, t)) = e^{-\lambda t}\frac{(\lambda t)^k}{k!}, \quad k = 0, 1, 2, \ldots$$

Poisson arrival process

- Since density $f(x) = \lambda e^{-\lambda x}$ is maximal for $x = 0$, short interarrival times occur more frequently than long ones. So arrivals tend to cluster:



- Superposition of many independent rarely occurring arrival processes is (close to) Poisson: this is why Poisson processes often occur in practice!
- Merging of two Poisson streams with rates λ_1 and λ_2 is again Poisson with rate $\lambda_1 + \lambda_2$, since

$$P(\text{arrival in } (t, t + \Delta)) \approx (\lambda_1 + \lambda_2)\Delta.$$

- Random splitting of Poisson stream with rate λ and splitting probability p is again Poisson with rate $p\lambda$, since

$$P(\text{arrival in } (t, t + \Delta)) \approx p\lambda\Delta.$$

Exponential machine model

Since exponentials are memoryless:

- State $X(t)$ of system at time t is characterized by number in the system

Let

$$p_i(t) = P(i \text{ jobs in the system at time } t), \quad i = 0, 1, \ldots.$$

How to determine these time dependent probabilities?

Via differential equations...

Exponential machine model

Clearly

$$\begin{aligned} P(\text{arrival in } (t, t+\Delta)) &= 1 - e^{-\lambda\Delta} \approx \lambda\Delta, \\ P(\text{departure in } (t, t+\Delta)) &= 1 - e^{-\mu\Delta} \approx \mu\Delta, \\ P(\text{no arr or dep in } (t, t+\Delta)) &= e^{-(\lambda+\mu)\Delta} \approx 1 - \lambda\Delta - \mu\Delta. \end{aligned}$$

So for $i = 1, \ldots$,

$$p_i(t+\Delta) = p_{i-1}(t)\lambda\Delta + p_i(t)(1 - \lambda\Delta - \mu\Delta) + p_{i+1}(t)\mu\Delta$$

and thus

$$\frac{p_i(t+\Delta) - p_i(t)}{\Delta} = p_{i-1}(t)\lambda - p_i(t)(\lambda+\mu) + p_{i+1}(t)\mu.$$

Letting $\Delta \to 0$ yields

$$\frac{d}{dt}p_i(t) = p_{i-1}(t)\lambda - p_i(t)(\lambda+\mu) + p_{i+1}(t)\mu.$$

Exponential machine model

Hence, for $i = 1, 2, \ldots$,

$$\frac{d}{dt}p_i(t) = p_{i-1}(t)\lambda - p_i(t)(\lambda + \mu) + p_{i+1}(t)\mu$$

and similarly, for $i = 0$,

$$\frac{d}{dt}p_0(t) \;=\; -p_0(t)\lambda + p_1(t).$$

This system of differential equations can be solved, but...

It is easier to look at long-run or limiting behavior as $t \to \infty$.

Provided $\lambda < \mu$, the limits

$$p_i = \lim_{t\to\infty} p_i(t)$$

exist, and thus $\lim_{t\to\infty} \frac{d}{dt} p_i(t) = 0 \quad$ (Why?).

Exponential machine model

Thus we get the following equations for the limiting probabilities p_i,

$$\begin{aligned} p_0\lambda &= p_1\mu, \\ p_i(\lambda+\mu) &= p_{i-1}\lambda + p_{i+1}\mu, \quad i = 1, \ldots \end{aligned}$$

Remarks:

- p_i can be interpreted as long-run fraction of time system is in state i.
- The equations for p_i are balance of flow equations:

  Flow out of state i = Flow into state i.

- Adding equations from state 0 to state i gives easier equations:

  $$p_i\lambda = p_{i+1}\mu, \quad i = 0, 1, \ldots,$$

  so also

  Flow from state i to $i+1$ = Flow from state $i+1$ to i.

Exponential machine model

Solution

$$p_i = p_0 \left(\frac{\lambda}{\mu}\right)^i, \quad i = 0, 1, \ldots,$$

and p_0 follows from

$$1 = \sum_{i=0}^{\infty} p_i = p_0 \frac{1}{1 - \lambda/\mu},$$

so

$$p_0 = 1 - \lambda/\mu.$$

The machine utilization is $\rho = 1 - p_0 = \lambda/\mu$ and thus

$$p_i = (1 - \rho)\rho^i, \quad i = 0, 1, \ldots,$$

which is the geometric distribution.

Fundamental relations

Little's law

Consider system in equilibrium

- $E(L)$ is mean number in system
- $E(S)$ is mean time spent in system
- λ is arrival rate (or departure rate)

Then:

$$E(L) = \lambda E(S)$$

The definition of system is flexible (e.g. queue, server, queue+server)

PASTA: Poisson Arrivals See Time Averages

Poisson arrivals see the system in equilibrium,
that is, they see the same as random outside observer!

Exponential machine model

- Poisson arrivals with rate λ
- Exponential service times with mean $1/\mu$
- Stability: $\lambda < \mu$ or $\rho = \lambda/\mu < 1$
- Single server
- FCFS service

Then:

- $p_k = P(k \text{ jobs in system}) = (1-\rho)\rho^k$, $k = 0, 1, \ldots$ (Geometric)
- $E(L) = \sum_{k=0}^{\infty} k p_k = \frac{\rho}{1-\rho}$
- $E(S) = E(L)/\lambda = \frac{1/\mu}{1-\rho}$
- $E(Q) = \sum_{k=1}^{\infty} (k-1) p_k = \frac{\rho^2}{1-\rho}$
- $E(W) = E(Q)/\lambda = \frac{\rho/\mu}{1-\rho}$

or via PASTA+Little...

Exponential machine model

- Poisson arrivals with rate λ
- Exponential service times with mean $1/\mu$
- Stability: $\lambda < \mu$ or $\rho = \lambda/\mu < 1$
- Single server
- FCFS service

Via PASTA+Little...

$$E(S) = E(L^a)\frac{1}{\mu} + \frac{1}{\mu},$$

where L^a is the number on arrival. By PASTA, $E(L^a) = E(L)$, so

$$E(S) = E(L)\frac{1}{\mu} + \frac{1}{\mu}$$

and thus by Little's law, $E(L) = \lambda E(S)$,

$$E(S) = \frac{1/\mu}{1-\rho}.$$

General machine model

- Poisson arrivals with rate λ
- General service times B with distribution $F_B(\cdot)$
- Stability: $\rho = \lambda E(B) < 1$
- Single server
- FCFS service

Then:

$$E(W) = E(Q^a)E(B) + \rho E(R) = E(Q)E(B) + \rho E(R)$$

where Q (Q^a) is number in queue (on arrival) and R is residual service time,

$$E(R) = \frac{E(B^2)}{2E(B)} = \frac{1}{2}\, E(B)\, (1 + c_B^2).$$

So with Little's law $E(Q) = \lambda E(W)$, we get...

General machine model

- Poisson arrivals with rate λ
- General service times B with distribution $F_B(\cdot)$
- Stability: $\rho = \lambda E(B) < 1$
- Single server
- FCFS service

Then:

$$E(W) = \frac{\rho E(R)}{1-\rho}$$

where R is residual service time,

$$E(R) = \frac{E(B^2)}{2E(B)} = \frac{1}{2} E(B)\,(1 + c_B^2)$$

so

$$E(W) = \frac{\rho}{1-\rho} \frac{1}{2} E(B)\,(1 + c_B^2)$$