SFWR ENG 4E03

Kemal Ahmed
Fall 2015
Instructor: Vincent Maccio

Note: material covered in Stats 3Y03 Summary will not be covered in this summary. To find a unit CTRL-F "[<unit>]", e.g. for Number of jobs in system, CTRL-F "[N]"

Contents

Statistics

Poisson parameter [λ]: rate
Service rate [μ]:
Continuous Random Variable (CRV):

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Think chemistry, i.e. cancelling units
Probability Density Function (PDF) [f(x)]:
Cumulative Density Function (CDF) [F(x)]:

Second Moment $E[x^2]$

Variance

$$\operatorname{var}(X) = E\left[X^2\right] - \left(E\left[X\right]\right)^2$$

- <u>Don't change probability</u>, but square X for calculation only

$$\begin{aligned}\operatorname{var}(x) &= E\left[(X-\mu)^2\right] \\ &= \sigma^2 \\ &= \int_{-\infty}^{\infty}(x-\mu)^2 f(x)\,\mathrm{d}x \\ &= \int x^2 f(x)\,\mathrm{d}x-\mu^2\end{aligned}$$

The higher your variance, the worse your system will perform.

Exponential

- **Mean** [E[X]]: 1/λ
  - a.k.a. Expected value
- **Variance**: $1/\lambda^2$
- **Probability Distribution Function (PDF)** [P(X=x)]: $\lambda e^{-\lambda x}/x!$
- **Cumulative Distribution Function (CDF)** [f(x)]: CDF = ∫PDF, i.e. $1 - e^{-\lambda x}$
- Memoryless
- not always for time

Uniform

- **Variance**: $(b-a)^2/12$
- **Mean**: (a+b)/2
- **PDF**: 1 / (b–a) , a ≤ x ≤ b
- **CDF**: x–a/b–a
- **Uniform Distribution**: no memoryless property

Binomial

- **Mean** [E[X]]: n × probability
- **Variance**: n × p × (1 – p)
- **Probability Distribution Function (PDF)** [P(X)]: $(n\ c\ x)p^x(1-p)^{n-x}$
- **Cumulative Distribution Function (CDF)** [f(x)]: $\sum_{i=0}^{\lfloor x \rfloor}\binom{n}{i}p_i(1-p)^{n-i}$

Operations Analysis

Device [i]: units that are in terms of *i* are specific to an individual device or node within a system
Total devices [k]:

Service Time [S]: time per specific job
1/μ

Visitation [V]: given or projected visits/jobs (closed system); cannot be calculated; basically a probability
[E(V)]: calculated visit/job ratio
P(visit)·total visits in previous node

Demand [D]: total service time for all jobs

$$D_i = E[S_i]\cdot V_i$$

$$D = \sum_{i=0}^{k} D_i$$

Bottleneck [D_{max}]: device with largest demand, utilization

Time in system [T]: time the job is in the system

$$E[T] = \frac{N}{X}$$

$$E[T] \geq \max\left(D, ND_{\max} - E[Z]\right)$$

If E[Z] = 0, T = R

Response Time [R]: time the job is *being processed* in the system
If E[Z] = 0, R = T

M/M/1: E[R] = 1/(μ–λ)
M/M/1/N: E[R] = E[N]/λ'
M/M/C_1/C_2: 1/μ, $C_1 \geq C_2$
M/M/C: E[R] = E[R_Q] + E[S]

Users [M]:
Optimal users [M*]:

$$M^* = \frac{D + E[Z]}{D_{\text{bottleneck}}}$$

Total Jobs [N]: N=M in a closed system

- *Little's Law*: $E[N] = \lambda E[T], \lambda = X$
- $E[N] = \lambda E[R], \lambda = X$
- M/M/1:
  - E[N] = λ/(μ–λ) = ρ/(1–ρ), if you have overall system λ
  - E[N] = $\sum_{i=0}^{\infty} i\pi_i$ ← probability × #jobs, if your λ or μ is different for each state
- M/M/1/N: E[N] is expected # jobs, N is max # jobs

  $$E[N] = \sum_{i=0}^{N} i\pi_i = \pi_0 \frac{\lambda}{\mu}\left(\frac{N\left(\frac{\lambda}{\mu}\right)^{N-1} - (N+1)\left(\frac{\lambda}{\mu}\right)^{N} + 1}{1 - \left(\frac{\lambda}{\mu}\right)^2}\right)$$

- M/M/C: go through Little's law
  - E[N] = E[N_Q] + ρ,
- M/M/∞:
- Jackson Network: E[N] = ΣE[N_i] = ΣPλ/(μ_i–Pλ) = Σ(λ_i/(μ_i–λ_i))

Think time [Z]: time it takes the user to put a request in and start, it's kinda like the frequency that users put in requests (seconds / request)

$$E[Z] = E[T] - E[R]$$

Throughput [X]: out-rate, max jobs / hour of full system

$$X \le \min\left(\frac{1}{D_{\max}}, \frac{N}{D+E[Z]}\right)$$

Note: $\frac{1}{D_{\max}}$ and $\frac{N}{D+E[Z]}$ converge at their lowest point, so equate them

$$X_i = E[V_i]X$$

Utilization [ρ]: probability that the processor is busy

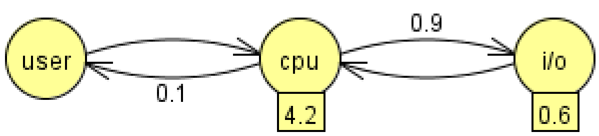
$$\rho_i = X_i E[S_i]$$

$$\rho_i = XD_i$$

$\rho = \lambda/c\mu$

Visitation Trick

If determining visitation at a node, establish a reference node from one of the incoming nodes that has a returning percentage, usually the user node



$V_{user} = 1 = 0.1 \cdot V_{CPU}$

Summation Equations

Geometric Series: $\sum_{i=0}^{\infty} r^i = \frac{1}{1-r}$, where 0 ≤ r ≤ 1 (because otherwise it would be unstable)

$$\sum_{i=0}^{\infty} r^{i+1} = \sum_{i=1}^{\infty} r^i = \frac{r}{1-r}$$

$$\sum_{i=1}^{\infty} r^i = \sum_{i=0}^{\infty} r^i - 1$$

Geometric Sequence: $S_n = \sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$

$$S_n = \sum_{i=1}^{n} r^i = \frac{r(1-r^n)}{1-r}$$

Removing the annoying factors:

$$\sum_{i=0}^{\infty} i(i+1)\rho^i$$

Take out a value so the integral takes out the i and i+1

$$= \rho \sum_{i=0}^{\infty} i(i+1)\rho^{i-1}$$

$$= \rho \frac{\mathrm{d}\rho}{\mathrm{d}i}\left(\sum_{i=0}^{\infty}(i+1)\rho^{i}\right)$$

$$= \rho \frac{\mathrm{d}\rho^2}{\mathrm{d}^2 i}\left(\sum_{i=0}^{\infty}\rho^{i+1}\right)$$

$$= \rho \frac{\mathrm{d}\rho^2}{\mathrm{d}^2 i}\left(\sum_{i=1}^{\infty}\rho^{i}\right)$$

$$= \rho \frac{\mathrm{d}\rho^2}{\mathrm{d}^2 i}\left(\sum_{i=0}^{\infty}\rho^{i} - \rho^0\right)$$

$$= \rho \frac{\mathrm{d}\rho^2}{\mathrm{d}^2 i}\left(\frac{1}{1-\rho} - 1\right)$$

DTMC

Discrete Time Markov Chains (DTMC): probability

[n]: number of tasks in queue / system

Steady state: n->∞
For discrete: use the sum of the X's, so $E[X] = \Sigma(P(X=i) \cdot X_i)$ and $E[X^2] = \Sigma(P(X=i) \cdot X_i^2)$

Balance Equations

$$\pi_n = \frac{\prod_{i=0}^{n} \lambda_i}{\prod_{i=0}^{n} \mu_i} \pi_0$$

^think of it like series / parallel, where you add multiple connections out in different directions (parallel) and multiply connections stacked onto each other (series)

OR $jobs_{in} = jobs_{out}$
OR $rate_{in} \times prob_{in} = rate_{out} \times prob_{out}$

Matrices

Rows: equations for nodes going out (add up to 1)
Columns: equations for nodes coming in

CTMC

Continuous Time Markov Chain (CTMC): rate

Poisson Process

Counting Process: a way of determining the time between consecutive occurrences of an event
Poisson Process: a *counting process*, whose time between arrivals uses Exponential Distribution

- $\lambda_{total} = \Sigma\lambda_i$
  - you can also split up λ into multiple λs
- Not only do you see each second as time independent, each stream of probabilities is independent
- $P(x;\lambda) = e^{-\lambda}\lambda^x/x!$
  - [x]: things will happen
  - [λ]: rate; $\lambda = \alpha t$
- [α]: expected number of events during unit interval
- [t]: time interval length
- $P_X(t) = \dfrac{e^{-\alpha t} \cdot (\alpha t)^X}{X!}$

Kendall notation

Job Processing time [μ]: rate of jobs leaving system (jobs/sec)
μ = 1/ processing_time_per_job

M/M/1 Queue
[M]: distribution of time between arrivals is Markovian (Memoryless) ~ exp(λ)
[M]: distribution of job processing times are Markovian (Memoryless) ~ exp(μ)
[1]: single server

$(\Sigma p_{out})\times\pi_i = \Sigma p_j\pi_j$, j=0..n, j≠i

π_0: percent of time that the queue is empty

Attributes:

- FIFO
- Infinite buffer

Variations

- M/M/2 Queue: same, except 2 servers
- M/M/C Queue: *C* servers
- $M/E_k/C$: Erlang k, i.e. series of exponential
- H()/M/C: hyperexpontial distribution
- PH/M/C: phase type, i.e. any combination of any number of exponentials with any rate
- M/G/C: Memoryless, general distribution of service time
- G/G/1: has not been solved yet
- M/M/1/1: 1 server, maximum 1 job in queue
- arrivals/processing/servers/jobs

[c]: number of servers

Think: one queue goes to multiple servers

$$\xrightarrow{\lambda}] \begin{matrix} \nearrow M_1 \\ \searrow M_2 \end{matrix}$$

Steady State

Steady State Probability: probability x many jobs will be in system (not just in each server!)

Blocking Probability [P_Q]: probability that a process will be blocked when entering the system and be placed in the queue. This is the same as *Steady State Probability*, since the number of jobs in a system dictates if a job will be blocked.

M/M/1

$\pi_0 = 1 - \lambda/\mu$

$\pi_i = \rho^i(1 - \rho)$

$$\pi_{n_1..n_k} = \prod_{i=1}^{k} \rho_i^{n_i}(1-\rho_i)$$

M/M/1/N

When you can only have up to N jobs in system queue.

[λ']: rate jobs enter the system, until the queue is full

$\lambda' = \lambda(1 - \pi_N)$

$$\pi_0 = \frac{1-\frac{\lambda}{\mu}}{1-\left(\frac{\lambda}{\mu}\right)^N} = \frac{1}{1+\sum_{i=1}^{N}\left(\frac{\lambda}{\mu}\right)^i}$$

$$\pi_i = \left(\tfrac{\lambda}{\mu}\right)^i \pi_0$$

Then remember, $1 = \Sigma\pi_i$ to find

Waiting: jobs put into the queue

Blocked: jobs not allowed in the queue

M/M/C

Useful if multiple jobs are sharing the same queue

Erlang-C Equation: $P_Q = \sum_{i=0}^{\infty}\pi_i = \frac{1}{c!}\left(\frac{\lambda}{\mu}\right)^c\left(\frac{1}{1-\rho}\right)\pi_0$

Given λ and μ, what should c be so $P_Q < \rho$

M/M/C/N

The following two equations are for the probability of entering the queue. Does the μ you use for equations double in M/M/2? No, but you'll see jobs coming out of a system at a rate of c·μ.

$$\pi_0 = \left[1 + \sum_{i=1}^{c-1}\frac{1}{i!}\left(\frac{\lambda}{\mu}\right)^i + \frac{1}{c!}\left(\frac{\lambda}{\mu}\right)^c\left(\frac{1}{1-\rho}\right)\right]^{-1}$$

$$\pi_i = \begin{cases} \dfrac{1}{i!}\left(\dfrac{\lambda}{\mu}\right)^i \pi_0, & i < c \\ \dfrac{1}{c!c^{i-c}}\left(\dfrac{\lambda}{\mu}\right)^i \pi_0, & i \geq c \end{cases}$$

M/M/∞

Same as M/M/C, except:

$$\pi_0 = e^{-\frac{\lambda}{\mu}}$$

and just find the unit

M/G/1

General Distribution of service time

Heaviside function: 1 if not zero

[E[A]]: arrivals

E[A] = ρ

Response Time

Waiting time in queue [R_Q]: response time of queue

$$E\left[R_Q\right] = \frac{1}{\lambda} P_Q \left(\frac{\rho}{1-\rho}\right)$$

M/M/1: $E\left[R_Q\right] = \dfrac{1}{\mu - \lambda} - \dfrac{1}{\mu}$

M/M/C: $E\left[R_Q\right] = \left(\dfrac{(\lambda/\mu)^c \mu}{(c-1)!(c\mu - \lambda)^2}\right)\pi_0$

M/M/∞: $E[R_Q] = 0$

Job Queue Size

Number of jobs in queue [N_Q]:

M/M/1: $\rho^2/(1 - \rho)$

M/M/C: $E\left[N_Q\right] = \pi_0 \dfrac{\lambda\mu\rho^c}{(c-1)!(c\mu - \lambda)^2}$

M/M/∞: $E[N_Q] = 0$

You need to know what is in the progression of each step

e.g.

When you have varying

$$\pi_n = (n+1)\left(\frac{\lambda}{\mu}\right)^n \pi_0$$

$$1 = \sum_{i=0}^{\infty} (i+1)\rho^i \pi_0$$

$$= \pi_0 \sum_{i=1}^{\infty} (i+1)\rho^i$$

$$= \pi_0 \frac{\mathrm{d}}{\mathrm{d}\rho}\left(\sum_{i=0}^{\infty} \rho^{i+1}\right)$$

$$= \pi_0 \frac{\mathrm{d}}{\mathrm{d}\rho}\left(\sum_{i=1}^{\infty} \rho^{i}\right)$$

$$= \pi_0 \frac{\mathrm{d}}{\mathrm{d}\rho}\left(\sum_{i=0}^{\infty} \rho^{i+1}\right)$$

Cost

Jobs incur a cost only if they're waiting
Hourly Cost of job [h]:

Square Root Staffing Rule

Given an M/M/c queue with arrival rate, λ, server speed, μ, and ρ is *large* (assume this means over 100, but we don't actually know what it means), α is a bound on P_Q, let c_α^* denote the least # of servers needed to ensure that $P_Q < \alpha$. Then

$c_\alpha^* \approx \rho + k\sqrt{\rho}$, where k = is the solution to

$\frac{k\Phi(k)}{\phi(k)} = \frac{1-\alpha}{\alpha}$, where Φ(·) is the CDF of the standard normal and ϕ() is its pdf

[K]: minimum # servers to stay stable λ/μ or ρ
[k]: a constant…just assume 1 for now

Essentially, the perfect number of servers is ρ + √ρ

e.g.)

α	k	ρ + k√ρ
0.8	0.178	10, 018
0.5	0.506	10, 051
0.2	1.06	10, 106
0.1	1.42	10, 142

[Q]: transition matrix

$$q_{ii} = -\sum_{j=i} q_{ij}$$

$$q_{ij} = \lim_{\Delta t \to 0} \frac{P\{X_{t+\Delta t} = j \mid X_t = i\}}{\Delta t}$$

Replace i <--> j to get q_{jj} and q_{ji}.

Jackson Networks

Open Loop

$P(N_1 = n_1) =$
balance pick

$$\pi_{\vec{n}} = \rho_1^{n_1}(1-\rho_1)\rho_2^{n_2}(1-\rho_2)\cdots\rho_k^{n_k}(1-\rho_k)$$

$\pi_{n\sim}$ = P(state of system n~) $= \prod_{i=1}^{k}$ P(n jobs at node i)

$$= \prod_{i=1}^{k} \rho_i^{n_i}(1-\rho_i)$$

Poisson Arrivals See Time Averages property (PASTA): the probability of a state (i.e. π_i) as seen by an outside random observer is the same as the probability of the state seen by an arriving customer. It is the open loop counterpart to arrival theorem

$\lambda_{total} = \Sigma\lambda_{in,i}$

Traffic Equations

For each node, what is the number of jobs entering?
$\lambda_x = R + \Sigma P_{i,entering} \cdot \lambda_{i,entering}$

response rate + probability of each job entering

Closed Loop

Since your values will become linearly independent, you cannot simply use your regular traffic equations. You need to estimate a fake value for one of your λ's and evaluate your probabilities using them.
Correction Variable [C]:
Jobs in system [M]:
1 = C(Σstates) = C(Σρ , such that sum of powers for each state = M)

Mean Value Analysis

(MVA): Finds E[R] of each node of a **closed Jackson network**.

I think it is n^2, whereas other methods are n^n
Visit Ratio [v]: based on a reference node, usually set $v_{ref} = c$

$p_i = \frac{v_i}{\sum_{j=1}^{k} v_j}$, e.g. $p_i = \frac{c}{c + 0.3c + 0.7c}$

1. Base case:
    a. $R^{(1)} = 1/\mu$
    b. $\lambda^{(1)} = M/(p_i \cdot R^{(1)})$
2. For k = 1, ..., K, compute:

a. $E\left[R_i^{(M)}\right] = \frac{1}{\mu_i} + \frac{p_i \lambda^{(M-1)} E\left[R_i^{(M-1)}\right]}{\mu_i}$

b. Little's Law: $\lambda^{(M)} = \frac{M}{\sum_{i=1}^{k} p_i E\left[R_i^{(M)}\right]}$

3. Plug it in: $E[N_i] = \lambda_i E[R_i]$

- Performs better than balance equations or Jackson Network, but can't find steady state distribution or PDF
- Recursive algorithm
- Only finds E[N], i.e. mean queue length

The higher your variance, the worse your system will perform.

Arrival Theorem: when a job arrives at a node within a closed Jackson network, there will be a number of jobs at the node, M – 1, where M is the expected number of jobs in the given node.

Inspection Paradox:

General Distribution

[X]:
Baskett, Chandy, Muntz and Palacios (BCMP) theorem: named after the authors of the paper
w.p. Width Probability
o.w. OtherWise

First Come, First Serve

First Come First Serve (FCFS): normal
optimal if IFR
If exponential, then same as M/M/1

Last Come, First Serve

Last Come First Serve (LCFS):

- Problems:
  - Context switch/ overhead
  - Isn't fair!
- Assume stable
- Inspection paradox: could be good when you have few larger jobs

E[N] = M/M/1

$$X = \begin{cases} 1, & w.p.9999 \\ 100000, & w.p.0.0001 \end{cases}$$

LCFS-PRe-emptive (LCFS-PR):

Shortest Remaining Processing Time

Shortest Remaining Processing Time (SRPT):

- # of jobs low
- response time low (optimal)
- need job size info
- Overhead
    - Starvation (fairness)

Processor Sharing

Processor Sharing (PS):

- Everyone is equal
- Constantly switch between all the jobs
- a.k.a. **thrashing**
- e.g. X = 5s, μ = 1/5
- Problems: overhead / switching costs
- a.k.a. Round Robin
- E[N] = M/M/1

$$E\left[R_{PS}\right] = E\left[R_{LCFS}\right] - E\left[R_{FCFS}\right] = \frac{1}{\mu - \lambda}$$

Longest Remaining Processing Time

Longest Remaining Processing Time (LRPT):

- only useful if highest priority jobs
- would eventually become PS because the length of time remaining will reach the next longest processing time

$E[R_{PS}] = E[R_{LCFS}] = \frac{1}{\mu - \lambda}$

Random

- Can be unfair
- Problems:
    - Large jobs starved

Failure

Pareto Power [α]: 0 < α < 2

[K]:

Pareto distribution: an exponential which doesn't start at 0 (a.k.a. **zipfian**)

CDF: $F(x) = 1 - \left(\frac{K_{\min}}{x}\right)^{a}$

PDF: $f(x) = \frac{\alpha K_{\min}^{\alpha}}{x^{\alpha+1}}, x > K_{\min}$

$$Var = \begin{cases} \frac{K_{\min}^{2}\alpha}{(\alpha-1)^{2}(\alpha-2)}, & \alpha > 2 \\ \infty, & \alpha \le 2 \end{cases}$$

Just think: 99% controls 50% and 1% controls the rest integral of the density function between k and p come out to 1

Failure / Hazard Rate [h]:

$$h(t) = \frac{f(t)}{1 - F(t)}$$

- **Uniform**: 1/(b–t)
- **Exponential**: λ

Increasing Failure Rate (IFR):
Decreasing Failure Rate (DFR):
Both: constant, since it's memoryless
Neither: when there are parts that increase and parts that decrease

Reaming Processing Time: