Queuing Theory

Performance Modeling Lecture #5

Slides adapted from Mark Claypool

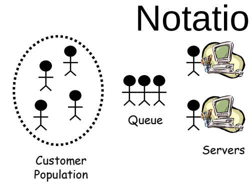
Introduction

- In computers, jobs share many resources: CPU, disks, devices
- Only one can access at a time, and others must wait in queues
- Queuing theory helps determine time jobs spend in queue
  - Can help predict response time

Outline

- Introduction
- Notation and Rules
- Little's Law
- Types of Stochastic Processes
- Analysis of a Single Queue, Single Server
- Analysis of a Single Queue, Multiple Servers

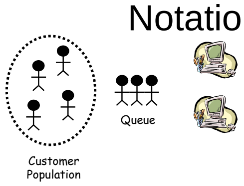
Notation (1 of 4)



- Imagine waiting for a PC in the computer lab (or checking out at a grocery store, or …)
  - Resources are "servers"
  - People are "customers"
- If all servers busy, customers wait in a "queue"

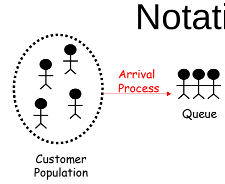
- For queuing analysis, need to specify:
  - Population size
  - Number of servers
  - System capacity
  - Arrival process
  - Service time distribution
  - Service discipline

Notation (2 of 4)



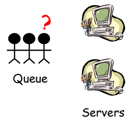
- Population size
  - Potential customers who can enter
  - Most real systems finite but easier to analyze if infinite
- Number of servers
  - Can be one or more
  - Assume identical, but if not then separate queuing system for each
- System capacity
  - Number that can wait plus be served
  - Most systems have finite queue length, but easier to analyze if infinite

Notation (3 of 4)



- Arrival process
  - Students arrive a $t_1, t_2, \ldots, t_j$
  - Interarrival times are $\tau_j = t_j - t_{j-1}$
  - Usually assume independent, identically distributed (IID)
- Arrival process (cont.)
  - Most common are Poisson arrivals
    - IID and exponentially distributed ($f(x)=\lambda e^{-\lambda x}$)
- Service time distribution
  - Amount of time each customer at server 
  - Again, usually IID
  - Most common are exponential 

Notation (4 of 4)



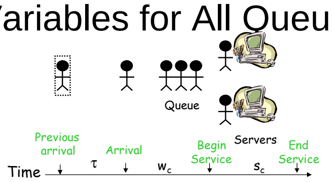
- Service discipline
  - Order customers called for servicing
  - Most common is FCFS
- Kendall notation:

  A/S/c/B/N/D
  - A is Arrival time distro
  - S is Service time distro
  - c is number of servers
  - B is number of buffers
  - N is population size
  - D is service discipline
- Some typical times used:
  - M Exponential
    - M means “memoryless” in that current arrival independent of past
  - D Deterministic
  - E Erlang
  - G General: Valid for all

Notation Example

- M/M/3/20/1500/FCFS – single queue system with:
  - Exponentially distributed arrivals
  - Exponentially distributed service times
  - Three servers
  - Capacity 20 (queue size is 20 – 3 = 17)
  - Population is 1500 total
  - Service discipline is FCFS
- Often, assume infinite queue and infinite population and FCFS, so just → M/M/3

Variables for All Queues



- τ = interarrival time
- λ = mean arrival rate

  λ = 1/E[τ]
- s = mean service time per job
- μ = mean service rate per server

  μ = 1/s, total rate cμ
- n_q = nr of jobs waiting in queue
- n_s = nr of jobs receiving service
- n = number of jobs in system

  $n = n_q + n_s$
- w =mean waiting time
- r = mean response time

  r = w + s

Rules for All Queues (1 of 4)

- *Stability Condition*
  - If the number of jobs in queue becomes infinite, the system becomes unstable. For stability, mean arrival rate less than mean service rate

    $\lambda < m\mu$
  - Does not apply to finite queue or finite population systems
    - Finite population cannot have infinite queue
    - Finite queue drops if too many arrive so never has infinite queue

Rules for All Queues (2 of 4)

- *Number in System versus Number in Queue*
  - Number of jobs is equal to waiting and servicing

    $n = n_q + n_s$
  - Also means:

    $E[n] = E[n_q]+E[n_s]$
  - So mean number of jobs is equal to mean number in queue plus mean number being serviced

    $Var[n] = Var[n_q]+Var[n_s]$
  - Variance of jobs equal to variance of queue + service
- Also, service rate of servers independent of jobs in queue: $Cov(n_q,n_s) = 0$

Rules for All Queues (3 of 4)

- *Number versus Time*
  - If jobs not lost due to buffer overflow the mean jobs is related to response time as:

    mean jobs in system = arrival rate x mean response time
  - Similarly

    mean jobs in queue = arrival rate x mean waiting time
  - Above equations known as “Little’s Law”
  - For finite buffers can use effective arrival rate (ignoring drops)

Rules for All Queues (4 of 4)

- *Time in System versus Time in Queue*
  - Time spent in system is sum of queue and service time

$$r = w + s$$

  - If service rate independent of jobs in queue

$$Cov(w,s) = 0$$
$$Var[r] = Var[w] + Var[s]$$

Outline

- Introduction
- Notation and Rules
- Little's Law
- Types of Stochastic Processes
- Analysis of a Single Queue, Single Server
- Analysis of a Single Queue, Multiple Servers

Little's Law

Mean jobs in system = arrival rate x mean response time

- Very commonly used in theorems
- Applies if jobs entering equals jobs serviced
  - No new jobs created, no new jobs lost
  - If lost, can adjust arrival rate to mean only those not lost
- Intuition: suppose monitor system and keep log of arrival and departures. If long enough, arrivals about the same as departures.
  - Let there be N arrivals in long time T. Then:

arrival rate = total arrivals / total time = N/T

Applying Little's Law

- Can be applied to subsystem, too
  - mean time in queue = arrival rate x waiting time
  - mean time being serviced = arrival rate x service time
- Example:
  - server satisfies I/O request in average of 100 msec. I/O rate is about 100 requests/sec. What is the mean number of requests at the server?
  - Mean number at server = arrival rate x response time

    = (100 requests/sec) x (0.1 sec)

    = 10 requests

Utilization Law

- Given average arrival rate λ.
- Average utilization of a system is time busy over total time

$$U = b/T$$

- Factor into:

$$U = b/T = (b/d)\ (d/T)$$

where d is number of departures and arrivals during time T

- Notice, (b/d) is average time spent servicing each of the d jobs. Call it s (s = b/d)
- Since balanced (in == out), $\lambda = d/T$
- So:

$$U = \lambda s \qquad (\textit{Utilization Law})$$

Applying Utilization Law

- Consider I/O system with one disk and one controller. If average time required to service each request is 6 msec, what is maximum request rate it can tolerate?
- Maximum will occur when 100% utilized, so U=1
- Substituting $U = \lambda s$, we get:

$$1 = \lambda_{max} s$$

- So, $\lambda_{max} = 1 / (6 \times 10^{-3}) = 167$ requests/sec

Utilization Law

- Notice, utilization law $U = \lambda s$ can be written as:

  $$U = \lambda/\mu$$

  where μ is the average service rate
- Ratio λ/μ is often called *traffic intensity*
  - Given own symbol $\rho = \lambda/\mu$
- If ($\rho > 1$) then $\lambda > \mu$ (arrival rate greater than service rate)
  - Jobs arrive faster than can be processed
  - Queue grows to infinity
  - Unstable
- Must have ($\rho < 1$) for stability (so U never > 100%)

Operational Analysis

- Using Little's Law and Utilization Law can say things about average behavior
  - Requires no assumptions about distribution times of arrivals or servicing
  - High level view
- But can not say things about, say, maximum or worst case
- For example, cannot use it to determine needed buffer space to enqueue incoming requests
- Will use stochastic distributions and queuing theory to get more detailed analysis

Outline

- Introduction
- Notation and Rules
- Little's Law
- Types of Stochastic Processes
- Analysis of a Single Queue, Single Server
- Analysis of a Single Queue, Multiple Servers

Types of Stochastic Processes (1 of 5)

- Number of jobs at CPU of computer system at time t is a random variable (n(t))
- To specify such random variables, need probability distribution function for each t
  - Same with waiting time (w(t))
- These random functions of time or sequences are called *stochastic processes*
- Useful for describing state of queuing systems

Types of Stochastic Processes (2 of 5)

- *Discrete-State and Continuous-State Process*
  - Depends upon values its state can take
  - If finite or countable → discrete
  - Ex: jobs in system n(t) can only take values 0, 1, 2 … countable, so discrete-state process
    - Also called a *stochastic chain*
  - Ex: waiting time w(t) can take any real value, so continuous-state process
- *Markov Process*
  - If future states depend only on the present and are independent of the past then called *markov process*
  - Makes it easier to analyze since do not need past trajectory, only present state
  - Also memory-less in that don't need length of time in current state

Types of Stochastic Processes (3 of 5)

- *Birth-Death Process*
  - Markov in which transitions restricted to neighboring states only are called *birth-death process*
  - Can represent states by integers, s.t. process in state n can only go to state n+1 or n-1
  - Ex: jobs in queue with single server can be represented by birth-death process
    - Arrival (birth) causes state to change by +1 and departure after service (death) causes state to change by –1
    - Only if arrive individually, not in batch

Types of Stochastic Processes (4 of 5)

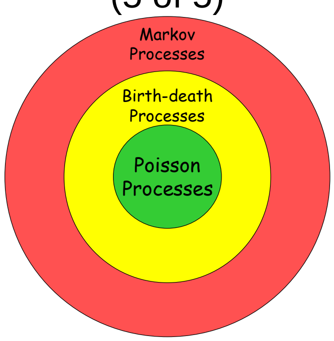
- *Poisson Processes*
  - If interarrival times are IID and exponentially distributed, then number of arrivals over interval [t,t+x] has a Poisson distribution → *Poisson Process*
  - Popular because arrivals are memoryless
  - Also:
    - Merging k Poisson streams with mean rate λ_i gives another Poisson stream with mean rate:

$$\lambda = \Sigma\lambda_i$$

Types of Stochastic Processes (4 of 5)

- *Poisson Processes* (continued)
  - Also
    - If Poisson stream split into k substreams with probability p_i, each substream is Poisson with mean rate λp_i
    - If arrivals to single server with exponential service times are Poisson with mean λ, departures are also Poisson with mean λ, if ($\lambda<\mu$)
      - Same relationship holds for m servers as long as total arrival rate less than total service rate

Types of Stochastic Processes (5 of 5)



Questions

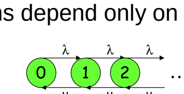
- M/D/10/5/1000/LCFS
  - What can you say about it?
  - What is bad about it?
- Which has better performance: M/M/3/300/100 or M/M/3/100/100?
- During 1 hour, name server received 10,800 requests. Mean response time 1/3 second.
  - What is the mean number of queries in system?

Outline

- Introduction
- Notation and Rules
- Little's Law
- Types of Stochastic Processes
- Analysis of a Single Queue, Single Server
- Analysis of a Single Queue, Multiple Servers

Single Queue, Single Server - M/M/1 Queue (1 of 6)

- Only one queue, exponentially distributed arrivals and service time
  - Ex: CPU in a system, processes in queue
- No buffer or population limitations
- Can often be modeled as birth-death process
  - Jobs arrive individually (not batch)
  - Changes state to n+1 (birth), n-1 (death)
- Transitions depend only on current state



Notation: probability of being in state n is P_n

Single Queue, Single Server - M/M/1 Queue (2 of 6)

- At any state, probability of going up same as probability coming down (balanced)

$$\lambda P_{n-1} = \mu P_n \text{, or}$$
$$P_n = (\lambda/\mu) P_{n-1} = \rho P_{n-1}$$

- We have: $P_1 = \rho P_0$, $P_2 = \rho P_1$, ..
- In general, probability of exactly n jobs in the system is:

$$P_n = \rho^n P_0$$

- We want a closed form for P_n (with no P_0)

Single Queue, Single Server - M/M/1 Queue (3 of 6)

- All probabilities add to 1, so:

$$\Sigma P_n = \Sigma \rho^n P_0 = 1 \qquad n=0,1,\ldots,\infty$$

- Expanding:

$$\rho^0 P_0 + \rho^1 P_0 + \rho^2 P_0 + \ldots = 1$$
$$P_0 = 1 / (\rho^0 + \rho^1 + \rho^2 \ldots) = 1/\Sigma\rho^n$$

- Since $\rho < 1$ for stability, can be shown that sum converges:

$$P_0 = 1-\rho \quad \text{and} \quad P_n = (1-\rho)\rho^n$$

- Can now derive many useful performance parameters for M/M/1 queue

Single Queue, Single Server - M/M/1 Queue (4 of 6)

- Mean jobs in system

$$E[n] = \Sigma n P_n = \Sigma n(1-\rho)\rho^n = \rho/(1-\rho) \qquad n=0,\ldots,\infty$$

- Variance of jobs in system

$$Var[n] = E[(n - E[n])^2] = E[n^2] - (E[n])^2$$
$$\Sigma n^2 (1-\rho)\rho^{n^2} - (\Sigma n(1-\rho)\rho^n)^2 = \rho/(1-\rho)^2$$

- Probability of n or more jobs

$$Pr(\geq n \text{ jobs in system}) = \Sigma P_j \quad j=n,\ldots,\infty$$
$$= \Sigma(1-\rho)\rho^j = \rho^n$$

- Mean response time
  - Using Little's law
    - mean jobs = mean arrv rate x mean resp time

Single Queue, Single Server - M/M/1 Queue (5 of 6)

- Mean jobs in queue (use n-1 since at most one serviced)

$$E[n_q] = \Sigma(n-1)P_n = \Sigma(n-1)(1-\rho)\rho^n = \rho^2/(1-\rho)$$

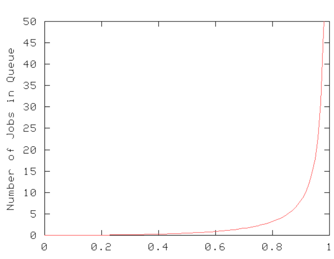
  - When no jobs in system, idle
  - When jobs in system, busy
- Utilization
  - Server is busy when 1 or more jobs in system
  - Average load, or average utilization

$$U = 1 - P_0 = 1-(1-\rho) = \rho$$

  - (Note, same as before)

Single Queue, Single Server - M/M/1 Queue (6 of 6)

- As utilization increases beyond 85%, queue rises sharply
  - Corresponding sharp rise in response time
- Utilization must be under 100%, but often lower
  - Ex: OS CPU scheduler often has 60-80% heuristic



Example of M/M/1 Queue Analysis (1 of 2)

- Network gateway, 4 Mbps, packet size 1000 bytes, Arrival rate of 125 packets/sec
  - What is the probability of overflow with only 12 buffers?
  - How many buffers are needed to keep packet loss to 1 in 1,000,000?

Example of M/M/1 Queue Analysis (2 of 2)

- Arrival rate λ=125 pps
- Service rate:
  - 4000000/8
  - = 500000 Mbytes/sec
  - 500000/1000 = 500 pps
  - So, μ=500 pps
- Utilization (traffic intensity):
  - $\rho = \lambda/\mu = 125/500 = .25$
- Mean packets in gateway:
  - $\rho/(1-\rho) = .25/.75 = .33$
- Probability of n packets in gateway
  - $Pr(n) = (1-\rho)\rho^n = .75(.25)^n$
- Mean time in gateway:
  - $(1/\mu) / (1-\rho)$
  - $= (1/500)/(1-.25) = 2.66$ms
- Prob of overflow = Pr(13+)
  - $= \rho^{13} = .25^{13} = 1.49 \times 10^{-8}$
  - ≈ 15 packets/billion
- To limit to less than 10^{-6}
  - $\rho^n \le 10^{-6}$
  - $n > \log(10^{-6})/\log(.25)$
  - > 9.96
  - – So, 10 buffers

Another Example of M/M/1 Queue Analysis (1 of 2)

- Web server. Time between requests exponential with mean time between 8 ms. Time to process exponential with average service time 5 ms.
  - A) What is the average response time?
  - B) How much faster must the server be to halve this average response time?
  - C) How big a buffer so only 1 in 1,000,000,000 requests are lost?

Another Example of M/M/1 Queue Analysis (2 of 2)

- Request rate
  - $\lambda = 1000 / 8 = 0.125$ requests per ms
- Service rate
  - $\mu = 1000 / 5 = 0.2$ requests per ms
- Utilization
  - $\rho = \lambda/\mu = .125/.2 = .625$
  - – So, 62.5% of capacity
- A) Avg response time
  - $(1/\mu) / (1-\rho)$
  - $= (1/.2)/(1-.625)$
  - = 13.33 ms
- To halve, want
  - $(1/\mu) / (1-\rho) = 6.665$
  - – Assume λ fixed, so change μ
  - $\mu = 1/6.665 + 0.125 = .257$
  - B) So, (.275-.2)/.2 * 100 = 37.5% faster
- 1 in 1 billion errors
  - – Buffer size k:
  - $Pr(k) \le 10^{-9}$
  - $\rho^k \le 10^{-9}$
  - C) So,
  - $k > \log(10^{-9}) / \log(.625)$
  - $k \approx 44$

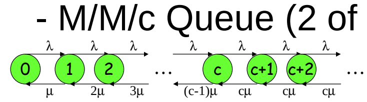
Outline

- Introduction
- Notation and Rules
- Little's Law
- Types of Stochastic Processes
- Analysis of a Single Queue, Single Server
- Analysis of a Single Queue, Multiple Servers

Single Queue, Multiple Servers - M/M/c Queue (1 of 4)

- Model multiple servers
  - – Model multiprocessor (SMP) systems
    - All "ready to run" processes in one queue
  - – Model Web server "farm"
  - – Model grocery store with single queue
- 'c' is the number of servers (Jain uses 'm')
- Assume arrival rate λ is the same
- Each server now can serve μ jobs per time
  - – Mean service rate $c\mu$
  - – Note, assumes no "cost" for determining server
- If any server idle (fewer than c jobs in system, say n), job serviced immediately
- If all c servers are busy, job waits in queue

Single Queue, Multiple Servers - M/M/c Queue (2 of 4)



- From above:
  - $\lambda_n = \lambda$ n=0,…,∞
  - $\mu_n = n\mu$ n=1,…,c-1
  - $\mu_n = c\mu$ n=c,…,∞
- Using balanced equations:
  - $P_n = [(c\rho)^n/n!]P_0$ n=1,…,c
  - $P_n = [(c\rho)^n/(c!c^{n-c})]P_0$ n>c
- Where ρ is traffic intensity
  - – Also, utilization of each server
- Find P_0 since sum must be 1
- $\Sigma P_n = \Sigma[(c\rho)^n/n!]P_0$ (1 to c) $+ \Sigma[(c\rho)^n/(c!c^{n-c})]P_0$ (c+1 to ∞) = 1
- Solve for P_0 =

$$\frac{1}{\Sigma[(c\rho)^n/n!] + (c\rho)^c/(c!(1-\rho))}$$

(n=1 to c in first term)

Single Queue, Multiple Servers - M/M/c Queue (3 of 4)

- Newly arriving job will wait if all servers are busy. Happens if more than c jobs.
  - Pr(>c jobs) $= \rho_c + \rho_{c+1} + \rho_{c+2} + \ldots = \Sigma P_n$ (n from c+1 to ∞)
    $= P_0 (c\rho)^c / c! \times \Sigma \rho^{n-c}$ (n from c+1 to ∞)
    $= [(c\rho)^c]/[c!(1-\rho)]\ P_0$
  - Known as *Erlang's C formula* (κ)
- Mean jobs in system
  - $E[n] = \Sigma n P_n = [P_0 (c\rho)^c]/[c!(1-\rho)^2] + c\rho$
    $= c\rho + \rho\kappa/(1-\rho)$

Single Queue, Multiple Servers - M/M/c Queue (4 of 4)

- Mean jobs in queue
  - $E[n_q] = \Sigma (n-c) P_n = P_0 (c\rho)^c / c! \times \Sigma (n-c) \rho^{n-c}$
    $= [P_0 \rho (c\rho)^c]/[c!(1-\rho)^2] = \rho\kappa/(1-\rho)$
- Mean response time
  - Using Little's law
    - mean jobs = mean arrv rate x mean resp time

$$E[n] = \lambda E[r]$$
$$E[r] = E[n]/\lambda$$
$$E[r] = 1/\mu + \kappa/[c\mu(1-\rho)]$$

- Mean waiting time $E[w] = E[n_q]/\lambda$
  $= [\rho\kappa/(1-\rho)]/\lambda = \kappa/[c\mu(1-\rho)]$

M/M/c Example (1 of 2)

- How does response time for previous M/M/1 Web server change if number of servers increased to 4?
  - Can model as M/M/4

M/M/c Example (2 of 2)

- Request rate λ=0.125
- Service rate μ=0.2
- Traffic intensity
  - $\rho = \mu/(c\lambda) = 0.1563$
- Probability of idle
- $P_0 = \dfrac{1}{\Sigma[(c\rho)^n/n!] + (c\rho)^c/(c!(1-\rho))\ P_0}$
  $= 0.532$
- Erlang's C formula
  - $\kappa = \dfrac{(4 \times 0.1563)^4}{4!(1-.01563)} (0.5352)$
    $= .0040326$
- So, average response time:

$$E[r] = 1/\mu + \kappa/c\mu(1-\rho)$$
$$= 1/.2 + .004326/(4)(.2)(1-.1563)$$
$$= 5.01 \text{ ms}$$

- Thus, increasing servers by 4 reduces response time by appx 62%

Another M/M/c Example (1 of 2)

- Students arrive at computer lab, 10 per hour. Spend 20 minutes at a terminal (assume exponentially distributed) and then leave. Center has 5 terminals.
  - A) How many terminals can go down and still be able to service the students?
  - B) What is the probability all terminals are busy?
  - C) How long is the average student in center?

Another M/M/c Example (2 of 2)

- Arrival rate λ=.167 per min, μ=.05 per min
- Utilization = $\lambda/(\mu c) = .167/(.05 \times 5) = .67$

A) Find c s.t. U > 1, so $1 > \lambda/(\mu c) \rightarrow c > \lambda/u \approx 4$
  - One terminal only can go down
- Prob all idle, P_0
  $= [1 + (5 \times .67)^5/[5!(1-.67)] + (5 \times .67)^1/1!$
  $+ (5 \times .67)^2/2! + (5 \times .67)^3/3! + + (5 \times .67)^4/4!]^{-1}$
  $= 0.0318$

B) Prob busy → *Erlang's C formula* (κ)
  Pr(>c jobs) $= [(cp)^c]/[c!(1-\rho)]\ P_0$
  $= [(5 \times .67)^5] / [5!(1-.67)] \times .0318 = .33$
  - So, 1/3 of the time you'll need to wait upon arriving

C) Time to wait: $E[w] = \kappa/[m\mu(1-\rho)]$
  $= .33/(5 \times .05 \times (1-.67)) = 4$ minutes

M/M/c versus M/M/1 (1 of 2)

- Consider what would happen if the terminals were distributed in separate labs, one per lab, across campus.
  A) Would you wait longer?
- Can model as separate M/M/1 systems and compare to M/M/c system

M/M/c versus M/M/1 (2 of 2)

- For M/M/1 λ=.167 / 5 = .0333 and μ=.05
  – ρ=.0333/.05 = .67
- Expected waiting time:
  $E[w] = E[n_q]/\lambda = [\rho^2 / (1-\rho)] / \lambda$
  $= [(.67)^2/(1-.67)] / (.033)$
  $\approx$ 41 minutes
  A) Yes. A *lot* longer.
- What is the model ignoring that may make the answer seem better?