Chapter 8: Queueing Theory

Math 419W/519
Prof. Andrew Ross
Eastern Michigan University

In this chapter, “mu” is a rate like it was in chapter 5&6, not a mean like it was in chapter 7.

What is it?

- A queue = a line of people or things waiting to be served
- Queueing Theory: ways of predicting how long the line or wait will be, or deciding on how many servers to have.

Applications

- Telephone call centers
- Factories
- Inventory
- Health care
- Firefighters/police/ambulance
- Repair technicians
- Car/Truck Traffic
- Internet data traffic
- UPS/FedEx
- Machines waiting for repair
- etc.

Air Travel

- Wait to find a parking space
- Wait for the parking shuttle
- Wait to check your bags
- Wait to get through security
- Wait to buy some food
- Wait for your plane to arrive
- Wait to board the plane
- Wait for luggage to finish loading
- Wait to de-ice
- Wait to take off
- Wait to de-ice
- Wait to take off
- Wait for the peanuts
- Wait to land
- Wait for the gate to free up
- Wait to de-plane
- Wait for your luggage
- Wait for a taxi

Basic Notation

- Arrival pattern/Service Pattern/#servers
- Pattern specifiers:
  - M = memoryless (Poisson Process for arrivals, or exponential distribution for service durations)
  - G = General (could be any distribution)
  - D = Deterministic
  - E = Erlang distribution
  - H = Hyperexponential distribution
  - PH = Phase-Type distribution

Start Simple, Ignoring:

- Time-of-day changes in arrival rate
- Priorities
- Balking (giving up before joining the queue)
- Abandoning/reneging (giving up while in queue)
- Retrials (trying back later after balking/abandoning)
- Batch arrivals
- Batch service
- Uncertainty in arrival rate
- Bilingual/Monolingual servers (Press 1 for English...)
- Virtual Hold (Press 1 and we will call you back...)

Example notation

- M/M/1 : arrivals follow a Poisson process, service times are exponentially distributed, single server.
- M/M/c: multiple servers. The basic call-center model.
- M/G/1: Poisson arrivals, general distribution of service durations

Notation: Input measures

- lambda = arrival rate
  - e.g. 120 calls per hour=30 seconds between calls, on avg.
- mu = service rate per server
  - e.g. 4 calls per hour = 15 minutes per call, on avg.
- c = # of servers (or k, or m, or n, or s)!
- rho = lambda/mu = “traffic”
  - For example, rho=120 calls per hour/4 calls per hour = 30 (units cancel—it's unitless!)

The usual problem

- Knowing lambda, mu, and c, what will be the average waiting time or line length?
  - There are some exact formulas for this.

- The real problem: knowing lambda and mu, and having a limit on the avg. waiting time, how many servers are needed?
  - There is a simple approximate formula for this, but hardly ever an exact formula.

Notation: Basic Output Measures

- L = avg # of people or jobs in the system
  - That's in the line plus those in service
- Lq = avg # of people or jobs in the line
  - Not including those in service
- W = avg time spent in the system
  - That's time spent in line, plus time spent in service
- Wq = avg time spent in the line
- Of course, W = Wq + 1/mu

Basic Output Measures: when?

- For queues involving people, we usually care about Wq, because once they get into service, they are happy.
  - At the emergency room, you want to see a doctor right away, but once you do, you don't want that doctor to rush.
- For queues involving objects, we usually care about W, because as long as they are in the system, they aren't being used profitably elsewhere.
- Less common to care about L or Lq—only when deciding how big the waiting area should be.
  - And even then, need to plan for much more than the average.

Fancy Output Measures

- % of time that a server is busy (“utilization”)
  - Higher is good to keep costs low
  - Lower is good to keep waiting times low
  - Overall, don't try to control it, except:
  - Keep it under 95% (?) for human servers
- Pr(wait < 20 seconds) = 80% (?)
  - Adapt to context: Emergency 911 vs IRS helpline
- Pr(had to wait at all)
- % Abandonment
- Pr(blocked) if there's a finite waiting room

Little's Law

- L = lambda*W, and Lq = lambda*Wq
- Along with W=Wq+1/mu
- Given any one of L, Lq, W, Wq, you can compute the other 3 easily.
- But Little's Law doesn't actually compute any of them in the first place.
- Also applies to infinite-server systems where Wq=0, W=1/mu.
- Also applies just to servers: avg # in service = arr. rate to service * 1/mu

General Plan

- Formulate a Markov Chain (usually CTMC)
- Find steady-state probabilities
- From those, compute L or Lq

Who is doing the observing?

- Suppose we have 1 arrival every hour exactly on the hour.
- And service takes exactly 59 minutes.
- This is a D/D/1 queue—simple, but boring.
- The server says: I'm busy 59/60=98.33% !
- Arriving customers say: we never saw the server busy!

When does that not happen?

- This is avoided if arrivals are Poisson:

  Poisson

  Arrivals

  See

  Time

  Averages

  =PASTA (proposition 8.2)

Chapter 8.3.1: M/M/1

- L = rho/(1-rho)
  - doesn't depend on lambda & mu separately, just their ratio
- Calculate in your head:

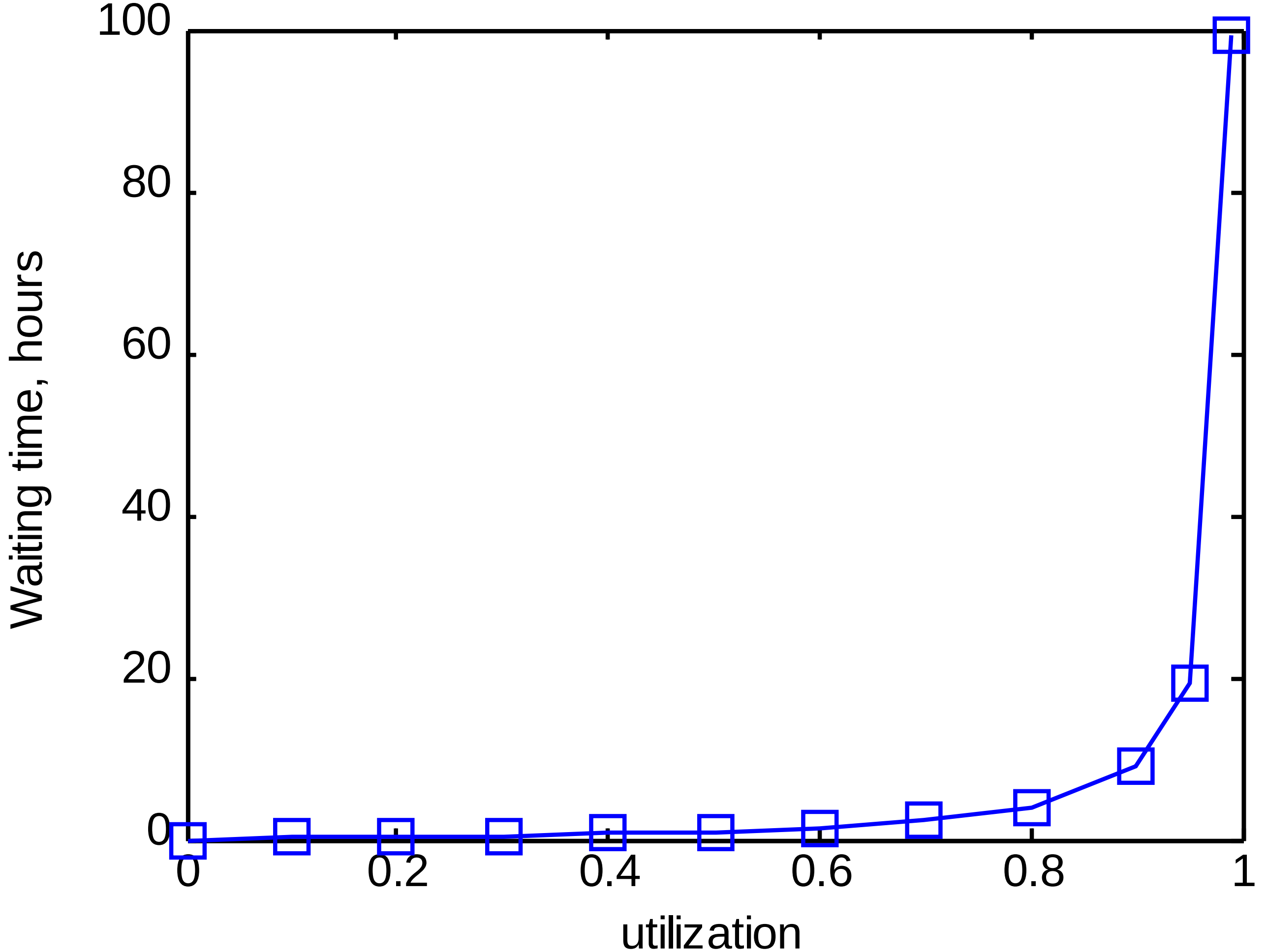
rho	L
0.5	
0.8	
0.9	
0.99	

Make a spreadsheet & graph

- L = rho/(1-rho) for an M/M/1 queue
- Use: rho=0, 0.25, 0.5, 0.75, 0.9, 0.99
- Use markers-with-connecting-straight lines
- Now try markers-with-connecting-smooth-lines

- If rho=0.99 and you spend 10% more money to make the server go 10% faster, now rho=0.9
- What % does L decrease?

M/M/1 Wq graph



Other M/M/1 facts

- Waiting time (if you are delayed) has an exponential distribution
  - If you can't see the queue, the time you've spent waiting gives no information about how much longer you will have to wait!
  - CoV is 100%: waiting time is, say, 5 minutes plus or minus 5 minutes.
- # of people in the system has a geometric distribution
  - So can't just plan on the average line length!
- Pr(system empty) = 1-rho

Read for yourself if interested:

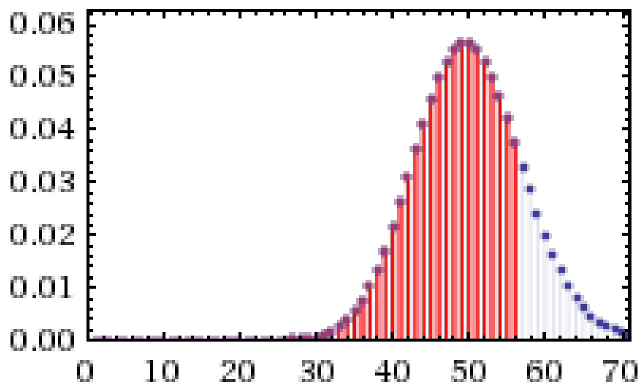
- Ch 8.3.2: finite buffer M/M/1/N
- Ch 8.3.4: Shoeshine Shop
- Ch 8.3.5: Bulk service

Ch 8.3.3: M/M/k

- Need lambda/mu = rho < k
  - Otherwise work piles up faster than we can serve it!
- Some books/web sites use r=lambda/mu, rho=r/c so rho<1 is needed.
- Formula shown in the book is:
  - Commonly repeated elsewhere
  - Hard to use—an ungainly sum
  - Impossible to use for more than 170 servers, though real call centers can have thousands of servers.
- Instead use web-based calculators:
  - Search for “Erlang-C”, a synonym for M/M/k
  - http://www.math.vu.nl/~koole/ccmath/ErlangC/
- Or Excel packages like QTS Plus (from the same place you get our class videos)

Other M/M/k facts

- Waiting time (if you are delayed) has an exponential distribution – similar to M/M/1
- # of people in the system has a combined Poisson/geometric distribution
  - Poisson for n<k, geometric for n>k



- Pr(system empty) = miniscule
- Pr(arrival must wait) = “Erlang-C” function

Approximate as single-server?

- Let mu=1 call per minute, lambda=50 calls per minute, and k=57 servers.

  Erlang-C calculator gives:

  - Wq=2.11 seconds (! not minutes)
  - Pr(not delayed) = 75.35%

- Approximate with a single really fast server?

  mu=57, lambda=50, k=1 server? rho=50/57,

  Wq=(1/mu)*rho/(1-rho)= 0.1253 minutes=7.5 seconds

  Pr(not delayed)=1-rho=12%

- Not a good approximation at all.

Single vs Multi-Server

- Single-server intuition still applies:
  - as rho approaches #servers, L&W go to infinity
  - But the numbers aren't the same for single vs multi
- Single-server: most people are in the queue
- Multi-server: most people are in service

3 Laws of Applied Queueing Theory

- Get there before the queue forms
- At the grocery store, stay to the far left or right (but not at tollbooths)
- For M/M/c, you need approximately

  #servers = rho + z*sqrt(rho)

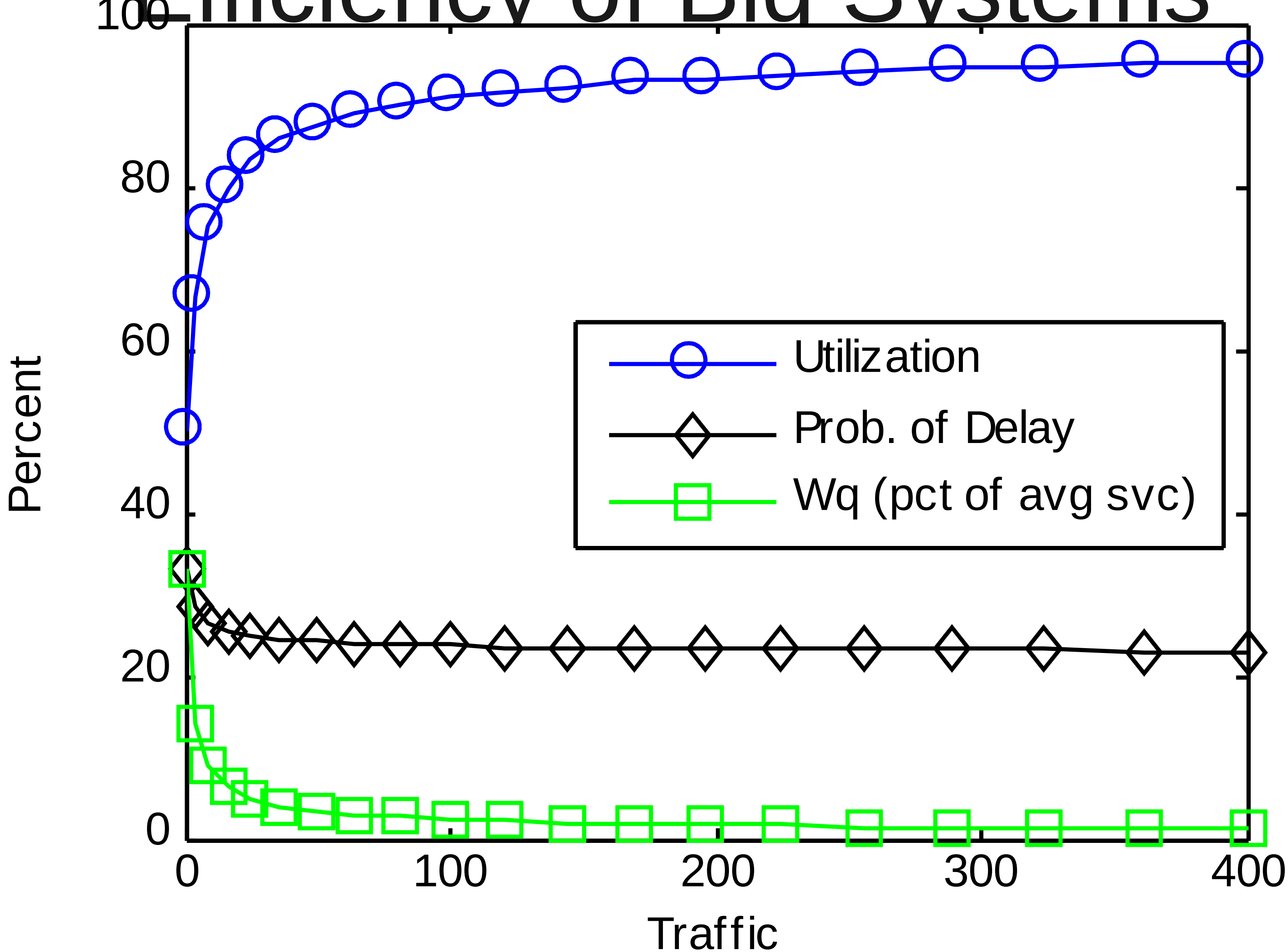
  Where z is 1 or 2: 1=good service, 2=great service.

  Technically, z is the Normal Distribution cutoff for Pr(not delayed). For example, if Pr(not delayed)=85%, then z=1

Practice with the 3rd Law

- Also called “Square-root staffing”
- If rho=10, you need 10+1*sqrt(10)=13.16 or 14 servers,
  - which is 31% more than rho alone.
- If rho=100, you need...
  - Which is ??% more than rho alone.
- If rho=1000, you need...
  - Which is ??% more than rho alone

Efficiency of Big Systems



- Wq falls off like 1/sqrt(rho)

More on efficiency

I stopped here at traffic=400, but biggest physical call centers are about 2000 people (can get bigger by virtual grouping)

Old hospital guideline: aim for 85% utilization. Bad!

Infomercial “operators are standing by”? They are consolidated & cross-trained.

In 1978 there were 661 Poison Control Centers in the US, now there are 51, with a national 1-800 number

Chapter 8.4: Networks of Queues

- If jobs arrive from outside & eventually leave,
- If all nodes have exponential service,
- And if a backup at one queue doesn't jam service at another
  - One study showed ER backups due to low # staff to move patients from ER to main hospital
- (and a few more assumptions)
- Then we can treat each queue independently.

- If jobs just circulate without arriving/leaving,
  - Like pallets in a factory
  - “closed” queueing network. Software can solve.

Chapter 8.5: M/G/1

- Service time not necessarily exponential.
- Need to know Squared Coefficient of Variation (SCV) of service times.
- The Variability-Utilization-Time (VUT) equation:
- Wq = (1+SCV)/2 * rho/(1-rho) * 1/mu
- Also called: Pollaczek-Khintchine formula
- Is exact, not an approximation, for M/G/1
- Recognize (1+SCV)/2 ? Inspection paradox!
- Variability hurts! Want SCV=0.

G/G/1 Approximation

- Include SCVa = SCV of inter-arrival times into VUT,
- Write service SCV as SCVs to avoid confusion.
- Kingman's equation (approximate):
- Wq = (SCVa+SCVs)/2 * rho/(1-rho) * 1/mu

G/G/k Approximation

- The “rho” term in the numerator

  = Pr(server busy) for single-server system.
- Replace with Pr(all servers busy)=Erlang-C for multiserver system
- Wq = (SCVa+SCVs)/2 * ErlangC/(1-rho) * 1/mu
- Approximating General service with Exponential when calculating ErlangC

Except for Data Networks

- Arrival of data packets isn't even a renewal process, let alone a Poisson Process
- Shows fractal patterns!
- Usually, averaging over a longer timespan reduces variability, but not for data networks.
- “Where Mathematics Meets the Internet” Walter Willinger and Vern Paxson

Service Ordering

- First-Come-First-Serve (FCFS) or FIFO
- Last-Come-First-Serve (LCFS) or LIFO
- Service in Random Order (SIRO) or RSS
- All have same averages (L, Lq, W, Wq)
- FCFS has lowest wait-time variance, LCFS highest.

Service Ordering: lower mean wait!

- Shortest Job First
  - needs estimate of service time for each job
- Shortest Remaining Processing Time
  - Also needs ability to interrupt jobs
- But either can really slow down long jobs.

- Round-Robin
  - Each job gets a little slice of time, e.g. 5ms-30ms

Appointment-based queueing

- E.g. dentist's office, doctor's office
- No-shows are a problem: forgetfulness, etc.
  - Some clinics with low-income customers will triple-book appointment slots!
    - Car breakdowns, Can't get time off, Can't get a babysitter
- Much less academic work done on this.
- A tiny trend toward only making same-day appointments: “Advanced Access”

Time-of-Day arrivals?

- Improving the SIPP Approach for Staffing Service Systems That Have Cyclic Demands. Linda V. Green, Peter J. Kolesar and João Soares. Operations Research, Vol. 49, No. 4 (Jul. - Aug., 2001), pp. 549-564
- For call center models, if rho/mu < 1, can break it into hour-long segments and treat each independently.
- If it's any worse, hire a queueing theorist.
- Procedure:
  - Forecast the arrival rate curve
  - Decide how many servers in each time block
  - Decide how many people on each shift (watch out for lunch breaks, coffee breaks, etc), “scheduling” (Math 560)
  - Decide which people work which shift (“rostering”)

Software

- Already mentioned:
  - Erlang-C calculators on the web & for Excel
  - QTS Plus
- Discrete-Event Simulation:
  - Arena, SIMUL8, GPSS, etc.
  - http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation1.html
- Can hack multiserver queues in excel:
  - http://www.informs.org/Pubs/ITE/Archive/Volume-7/Simpler-Spreadsheet-Simulation-of-Multi-Server-Queues

Bigger Issues

- If you add servers to improve service, fewer people will balk/abandon, and your servers might get busier.
- Game Theory—where is the equilibrium?