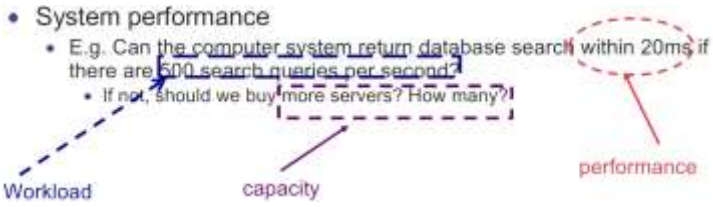
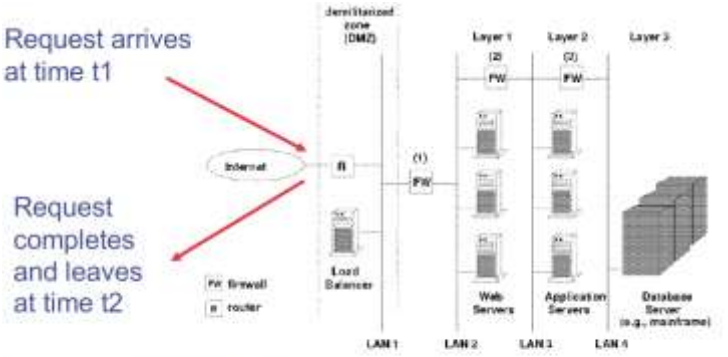
COMP9334 19T1 CONTENTS

Week 1A: Introduction to Capacity Planning

Performance metrics: response time, waiting time, throughput, availability



Response time of a system



Response time = t2 - t1.
Measured in seconds. Can be expressed as mean, standard deviation, probability distribution etc.

Response time = Waiting time + Processing time

Response time = Departure time - arrival time

Generally we need to know

- The arrival pattern
  - Ex: The arrival rate
  - Ex: The inter-arrival time statistical distribution
- The service time distribution
  - The time required to process the job

Since we are interested in response time, our models capture the time related aspects of the real systems e.g. queueing, processing units

We will learn different methods to determine response time in this course

Availability

Fraction of time the system is up and useable by users

- Ex: It is common for Internet Service Providers (ISP) to sign Service Level Agreement (SLA) with their commercial customers. One ISP guarantees that its network outage is less than 6 hours per 30 days. The network availability is 1 - 6/(30*24) = 99.17%

Probability

Probability A and Pro B =Pro (A and B) = Pro B * Pro (A | B)

Pro(A|B) means 在 B 发生的情况下，A 发生的概率

Pro (A or B) = Pro(A) + Pro (B) – Pro (A and B)

Throughput

The rate at which requests are completed

Ex: For network routers, throughput can be measured in

- Packets per second (pps)
  - Ex: 10 Mpps for 40-byte packets
  - Note: Should specify packet size
- Mb/s

Other throughput measures

- Web site: HTTP requests/s, bytes/s
- CPU: MIPS, FLOPS

Throughput is a function of the load

- A disk takes 0.01s to perform an I/O operation
- Maximum number of I/O operation per s = 100
- If 50 I/O operations arrive per second, the throughput = 50 I/O operations/s
- If 110 I/O operations arrive per second, the throughput = 100 I/O operations

- Can you find a formula relating throughout, offered load and max capacity?
- Throughput = min(offered load, max capacity)

If you find it difficult to do the previous page, you can try this real life analogy.

Throughput is a function of the load

- A barister can make a cup of coffee every 30 seconds
- Maximum number of cups of coffee the barister can make in an hour = 120
- If 50 customers arrive in an hour and each customer orders a coffee, the barister's throughput = 50 coffees / hour
- If 150 customers arrive in an hour and each customer orders a coffee, the barister's throughput = 120 coffees / hour

Utilization

Definition: Utilisation = Percentage of time over which the server is busy

What is the utilisation of the server over the first 12s?

- 8/12 = 66.7%

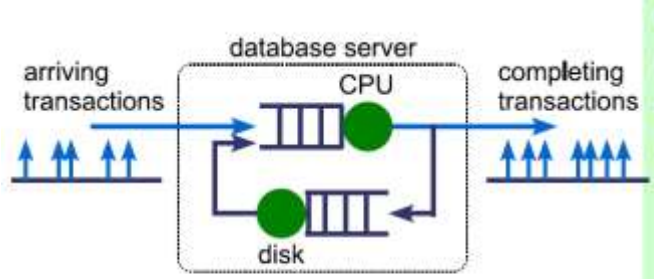
Service time

Time require to process a request at a resource

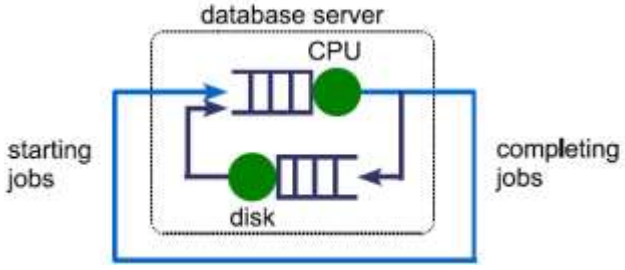
- Ex: The service time to send a 1000 byte packet over a 10 kbps link is 0.8s. In this case,
  - Service time = packet size / transmission rate
- Ex: The service time for to get a X byte large file from a disk is
  - Seek time + X / transfer rate
- For a class of resources, we have
  - Service time = Overhead + Job size / Processing rate

Week 1B: Queuing networks. Operational analysis

Open vs. closed queueing networks



Open queueing network
- External arrivals
- Workload intensity specified by arrival rate



Closed queueing network
- No external arrivals
- Workload intensity specified by customer population

Open queueing network
- Possibly unbounded #customers
- For stable equilibrium Throughput = arrival rate

Work in an open queueing network is called transaction

Closed queueing network
- Known #customers
- Throughput depends on #customers etc.

Work in a closed queueing network is called jobs

DB server – Multi-programming level

Some database server management systems (DBMS) set an upper limit on the number of active transactions within the system

This upper limit is called multi-programming level (MPL)

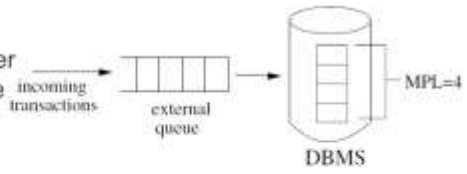
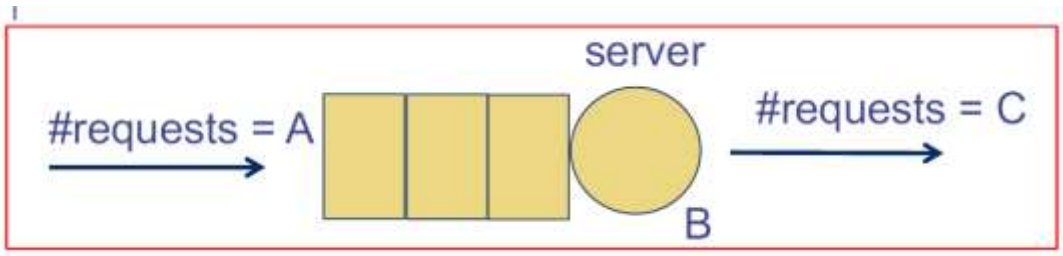


Figure 1. *Simplified view of the mechanism used in external scheduling. A fixed limited number of transactions (MPL=4) are allowed into the DBMS simultaneously. The remaining transactions are held back in an external queue. Response time is the time from when a transaction arrives until it completes, including time spent queueing externally to the DBMS.*

Single-queue example (Arrival rate, Output rate, etc.)



In an observational period of T, server busy for time B
A requests arrived, C requests completed

Deductions: Arrival rate λ = A/T
Output rate X = C/T
Utilisation U = B/T

Mean service time per completed request = B/C

Motivating example

Given

- Observation period = 1 minute
- CPU
  - Busy for 36s.
  - 1790 requests arrived
  - 1800 requests completed
- Find
  - Mean service time per completion = 36/1800 = 0.02s
  - Utilisation = 36/60 = 60%
  - Arrival rate = 1790/60 = 29.83 requests /s
  - Output rate = 1800/60 = 30 requests/s

Utilization law

The operational quantities are inter-related

Consider

- Utilisation U = B / T
- Mean service time per completion S = B / C
- Output rate X = C / T

Utilisation law – Can you relate U, S and X?

- U = S X

Utilisation law is an example of operational law.

If U ≠ S X, something is wrong

Equilibrium assumption

OA makes the assumption that

- C = A
- Or at least C ≈ A

This means that

- The devices and system are in equilibrium
  - Arrival rate of requests to a device = Output rate of requests for that device = Throughput of the device
  - The above statement also applies to the system, i.e. replace the word "device" by "system"

Signal in OA

We measure the basic operational quantities for each device (or other equivalent quantities) over a time of T

- A(j) = Number of request arriving at device j
- B(j) = Busy time for device j
- C(j) = Number of completed requests for device j

In addition, we have

- A(0) = Number of arrivals at the system
- C(0) = Number of completions for the system

Question: What is the relationship between A(0) and C(0) for a closed QNs?

Visit ratios

A job arriving at the system may require multiple visits to a device in the system

- Example: If every job (or transaction) arriving at the system will require 3 visits to the disk (= device j), what is the ratio of C(j) to C(0)?
  - We expect C(j)/C(0) = 3.
- V(j) = Visit ratio of device j
  = Number of times a job (transaction) visits device j
  - We have V(j) = C(j) / C(0)

Forced Flow Law

Since $$V(j) = \frac{C(j)}{C(0)}$$

$$X(j) = \frac{C(j)}{T} \text{ and } X(0) = \frac{C(0)}{T}$$

The forced flow law is

$$V(j) = \frac{X(j)}{X(0)}$$

Service time versus service demand

Ex: A job requires two disk accesses to be completed. One disk access takes 20ms and the other takes 30ms.

Service time = the amount of processing time required *per visit* to the device

- The quantities "20ms" and "30ms" are the individual service times.

D(j) = Service demand of a job at device j is the total service time required by that job

- The service demand for this job = 20ms + 30 ms = 50ms

Service demand

Service demand can be expressed in two different ways

- Ex: A job requires three disk accesses to be completed. One disk access takes 20ms and the others take 30ms and 28ms.
  - What is D(j)? 78ms.
  - What are V(j) and S(j)?
    - Recall that S(j) = mean service time of device j
  - V(j) = 3. S(j) = 26ms.

- Service demand D(j) = V(j) S(j)

Service demand law

Given D(j) = V(j) S(j)

Since $V(j) = \frac{X(j)}{X(0)}$

$\Rightarrow D(j) = \frac{X(j)S(j)}{X(0)}$

- What is X(j) S(j)?
- It is U(j)

Service demand law $D(j) = \frac{U(j)}{X(0)}$

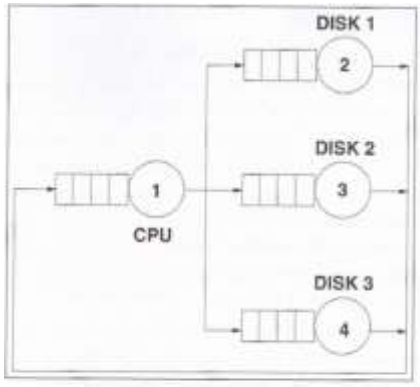
Service demand law D(j) = U(j) / X(0)

- You can determine service demand without knowing the visit ratio
- Over measurement period T, if you find
  - B(j) = Busy time of device j
  - C(0) = Number of requests completed
- You've enough information to find D(j)

The importance of service demand

- You will see that service demand is a fundamental quantity you need to determine the performance of a queueing network
- You will use service demand to determine system bottleneck in Lecture 2A

Server example exercise



Measurement time = 1 hr		
	# I/O/s	Utilisation
Disk 1	32	0.30
Disk 2	36	0.41
Disk 3	50	0.54
CPU		0.35
Total # jobs=13680		

What is the service time of Disk 2?
What is the service demand of Disk 2?
What is its visit ratio?

Service time = U2/X2 = 0.41/36 = 11.4ms
System throughput = 13680/3600 = 3.8 jobs/s
Service demand = 0.41/3.8 = 108ms
Visit ratio = 36/3.8 = 108 / 11.4 = 9.47

Little's law

Due to J.C. Little in 1961

- A few different forms
  - The original form is based on stochastic models
- An important result which is non-trivial
  - All the other operational laws are easy to derive, but Little's Law's derivation is more elaborate.

Consider a single-server device

- Navg = Average number of requests in the device
  - When we count the number of requests in a device, we include the one being served and those in the queue waiting for service

- X = Throughput of the device
- Ravg = Average response time of the requests
- Navg = Average number of requests in the device
- Little's Law (for OA) says that

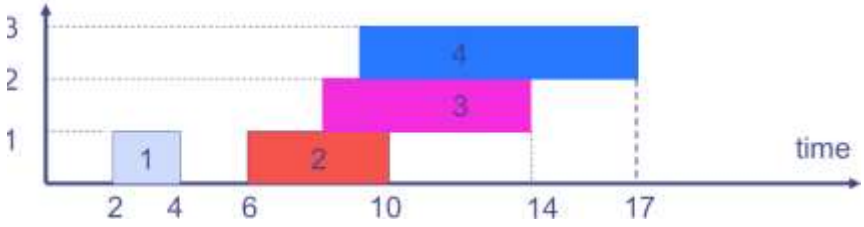
$$Navg = X * Ravg$$

We will argue the validity of Little's Law using a simple example.

Consider the single sever queue example

Request index	Arrival time	Service time	Departure time
1	2	2	4
2	6	4	10
3	8	4	14
4	9	3	17

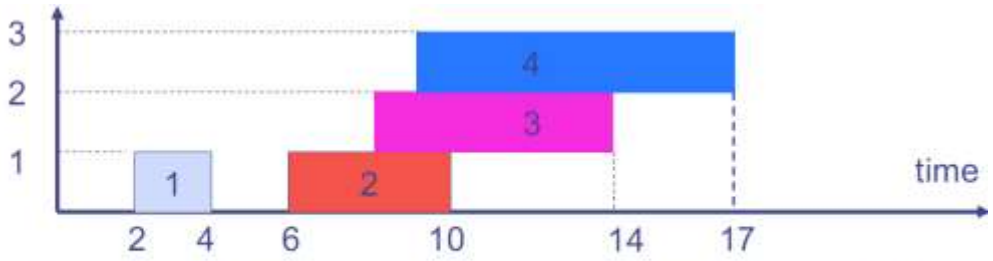
Let us use blocks of height 1 to show the time span of the requests, i.e. width of each block = response time of the request



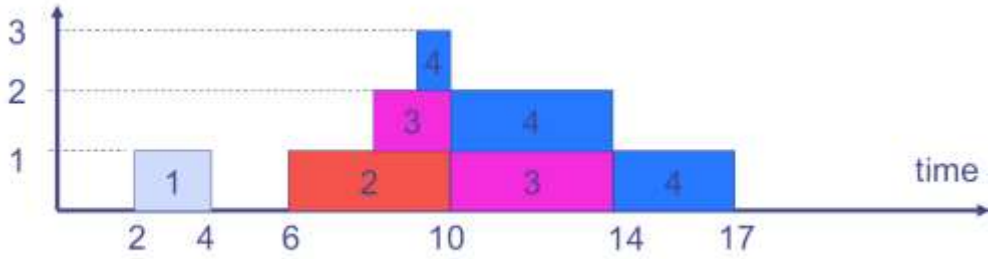
Assuming that in the measurement time interval [0,20] these 4 requests arrive and depart from this device, i.e. the device is in equilibrium.

Total area of the blocks
= Response time of request 1 + Response time of request 2 + Response time of request 3 + Response time of request 4
= Average response time over the measurement interval * Number of requests completed over the measurement interval

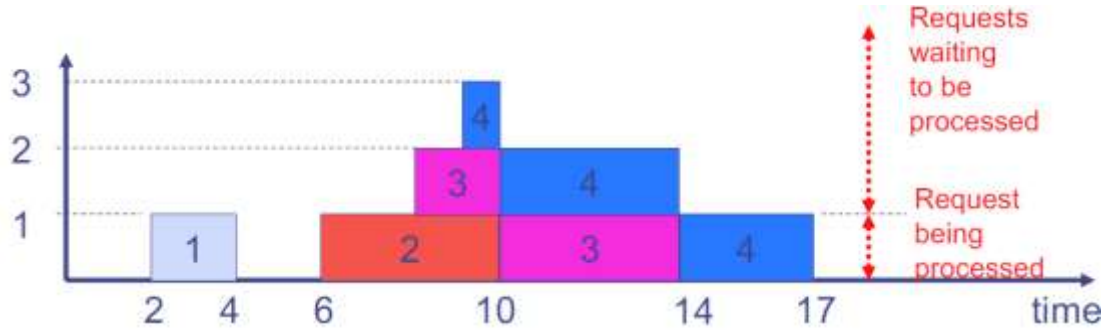
This is one interpretation. Let us look at another.



Let us assume these blocks are "plasticine" and let them fall to the ground. Like this.



There is an interpretation of the height of the graph.



Interpretation: Height of the graph = #requests in the device
E.g. Number of requests in [9,10] = 3
E.g. Number of requests in [11,12] = 2 etc.

Again, consider the measurement time interval of [0,20].

Area under the graph in [0,20]
= Height of the graph in [0,1] + Height of the graph in [1,2] + …
Height of the graph in [19,20]
= #reqs in [0,1] + #reqs in [1,2] + … + #reqs in [19,20]
= Average number of requests in [0,20] in the device * 20

Area = Average response time over [0,T] *
Number of requests completed in [0,T]

Area = Average number of requests in [0,T] * T

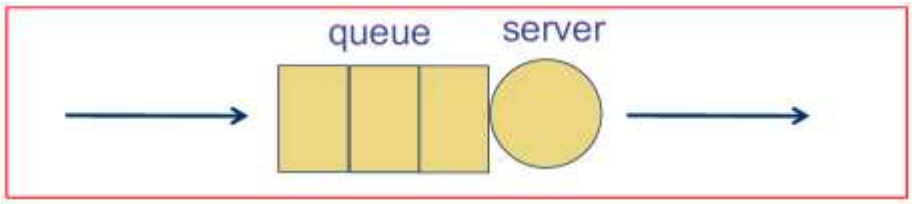
Area = Average response time of all jobs *
Number of requests completed in [0,T] (Interpretation #1)
= Average #requests in [0,T] * T (Interpretation #2)

Since Number of requests completed in [0,T] / T
= Device throughput in [0,T]

We have Little's Law.

Average number of requests in [0,T]
= Average response time of all reqs * Device throughput in [0,T]

Using Little's Law



- A device consists of a server and a queue
- The device completes on average 8 requests per second
- On average, there are 3.2 requests in the device
- What is the response time of the device?
- Mean throughput X = 8 requests/s
- Mean number of requests Navg = 3.2 requests
- By Little's Law, average response time = Navg/X = 3.2 / 8 = 0.4 s

Little's Law

- Mean #requests = Mean response time * Mean throughput

If #requests in the device , then response time

- And vice versa

Little's Law can be applied at many different levels

Little's law can be applied to a device

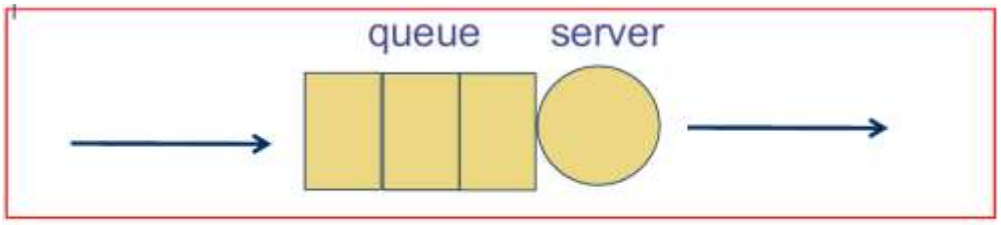
- Navg(j) = Ravg(j) * X(j)

A system with K devices

- Navg(j) = #requests in device j
- Average number of requests in the system Navg = Navg(1) + ... + Navg(K)
- Average response time of the system = Ravg

We can also apply it to an entire system

- Navg = Ravg * X(0)



- The device completes on average 8 requests per second
- On average, there are
  - 3.2 requests in the device
  - 2.4 requests in the queue
  - 0.8 requests in the server
- What is the mean waiting time and mean service time?
- Hint: You need to draw "boxes" around certain parts of the device and interpret the meaning of response time for that box.

Mean throughput X = 8 requests/s

Mean waiting time = 2.4 / 8 = 0.3 s

Mean service time = 0.8 / 8 = 0.1 s

Week 1A: Revision problems

Question 1

An important part of performance analysis is to model the workload. In this question, you will look at a very simple model and we will generalise it to a very well known model in performance analysis in the lecture in Week 2.

Consider a user who may send HTTP requests to a web server. In the time interval $[k\,\delta, (k+1)\,\delta)$ where k is a non-negative integer, there is a probability of p that this user will send an HTTP request to a web server and there is a probability of $(1-p)$ that this user will not send. Assuming that the probability the user sends (or not send) in each time interval is independent. Assuming that the current time is 10δ, what is the probability that this user will not send an HTTP request to the web server before 30δ?

Prob (the user will not send before 30δ)

= Prob (the user will not send in $[10\delta,11\delta)$) x

Prob (the user will not send in $[11\delta,12\delta)$) x ….

Prob (the user will not send in $[29\delta,30\delta)$)

(note: the probability to send is independent for each time)

= (1-p)^20

Question 2

This is a revision question on probability distribution which you should be able to solve if you have the pre-requisites.

Consider a continuous probability distribution with sample space is [1,∞) and probability density function

- f(x) = a / x^3 for x ≥ 1

What is the value of *a* in order that f(x) be a valid probability density function?

What is the probability the probability that a number drawn from this distribution is exactly 10?

Given this probability density function, what is the probability that a number drawn from this distribution has a value greater than 10?

- In order that the probability density function be valid, the probability that the number is drawn between [1,∞) is 1.

$$\int_1^\infty \frac{a}{x^3} = 1 \Rightarrow \left[\frac{ax^{-2}}{-2}\right]_1^\infty = 1 \Rightarrow \frac{a}{2} = 1 \Rightarrow a = 2$$

- Probability that a number drawn is exactly 10 is zero
  - Explanation: The numbers that can come from this distribution is in the range [1,∞) and there are infinite numbers, hence the probability of getting just one number (in this case 10), is zero.
- Probability that a number drawn is greater than 10 =

$$\int_{10}^\infty \frac{2}{x^3} = \left[\frac{2x^{-2}}{-2}\right]_{10}^\infty = 0.01$$

Note: The probability distribution that you've worked with is called a Pareto distribution. It has what is known as a heavy tail properties. This probability distribution appears very often in modern computer performance analysis.

Week 1B: Revision problems

5. A transaction processing system is monitored for one hour. During this period, 5,400 transactions are processed. What is the utilization

of a disk if its average service time is equal to 30 msec per visit and the disk is visited three times on average by every transaction?

6. The average delay experienced by a packet when traversing a computer network is 100 msec. The average number of packets that cross the network per second is 128 packets/sec. What is the average number of concurrent packets in transit in the network at any time?

7. A file server is monitored for 60 minutes, during which time 7,200 requests are completed. The disk utilization is measured to be 30%. The average service time at this disk is 30 msec per file operation request. What is the average number of accesses to this disk per file request?

Question 5

The service demand law says that utilisation = service demand x system throughput. Since the average service time is 30ms and every transaction visits the disk 3 times, each transaction requires a service demand of 3 x 30ms = 90ms. In one hour, 5400 transactions mean a throughput of 5400/3600 = 1.5 transactions/second. The utilisation is therefore 0.09*1.5 = 0.135.

Question 6

This question is solved by applying Little's Law to the computer network. The throughput of the network is 128 packets/s. The response time is the average delay experienced by the network = 100ms = 0.1s. Thus, the average number of packets (by Little's Law) is 128 * 0.1 = 12.8 packets.

Question 7

Method 1: Apply the operational Law

The system throughput was 7200/(60*60) = 2 requests per second.
By service demand law, the service demand at the disk is utilisation of the disk / system throughput = 0.3 / 2 = 0.15s.
The average number of accesses (= visit ratio) was 0.15ms/30ms = 5.

Method 2: From first principles

The monitoring period was 60 minutes and the disk utilisation was 30%, that means the disk was busy for 60 * 0.3 = 18 minutes. During this 18 minutes, 7200 requests are completed, so each request took on average 18*60/7200 = 0.15s. Since each file operation took 30ms, thus the average number of visits was 0.15s/30ms= 5.

Derivative Formulas(求导公式)

$y = C \ , \ y' = 0$

$y = x^n \ , \ y' = nx^{n-1}$

$y = sinx \ , \ y' = cosx$

$y = cosx \ , \ y' = -sinx$

$y = tanx \ , \ y' = \frac{1}{cos^2 x} = sec^2 x$

$y = cotx \ , \ y' = -\frac{1}{sin^2 x} = -csc^2 x$

$y = secx \ , \ y' = secx \cdot tanx$

$y = cscx \ , \ y' = -cscx \cdot cotx$

$y = ln|x| \ , \ y' = \frac{1}{x}$

$y = log_a x \ , \ y' = \frac{1}{xlna}$

$y = e^x \ , \ y' = e^x$

$y = a^x \ , \ y' = a^x lna \quad (a > 0, a \neq 1)$

$y = arcsinx \ , \ y' = \frac{1}{\sqrt{1 - x^2}}$

$y = arctanx \ , \ y' = \frac{1}{1 + x^2}$

$y = arccotx \ , \ y' = -\frac{1}{1 + x^2}$

$(arccotx)' = -\frac{1}{1 + x^2}$

$$\int f[\varphi(x)]\varphi'(x)\mathrm{d}x = \int f(\varphi(x))\mathrm{d}\varphi(x)$$

(也称配元法，凑微分法)

分部积分:

$$\int u(x)v'(x)dx = u(x)v(x) - \int u'(x)v(x)dx$$

①$(u\pm v)'=u'\pm v'$

②$(uv)'=u'v+uv'$

③$(u/v)'=(u'v-uv')/\ v\text{^}2$

积分号下的求导法

$d(\int f(x,t)dt\ \ \varphi(x),\psi(x))/dx=f(x,$

$\psi(x))\psi'(x)-f(x,\varphi(x))\varphi'(x)+\int[f\ 'x(x,t)dt\ \ \varphi(x),\psi(x)]$

Week 2A: Operational Analysis (2). Workload Characterisation

We have derived these operational laws

- Utilisation law U(j) = X(j) S(j)
- Forced flow law X(j) = V(j) X(0)
- Service demand law D(j) = V(j) S(j) = U(j) / X(0)
- Little's law N = X R

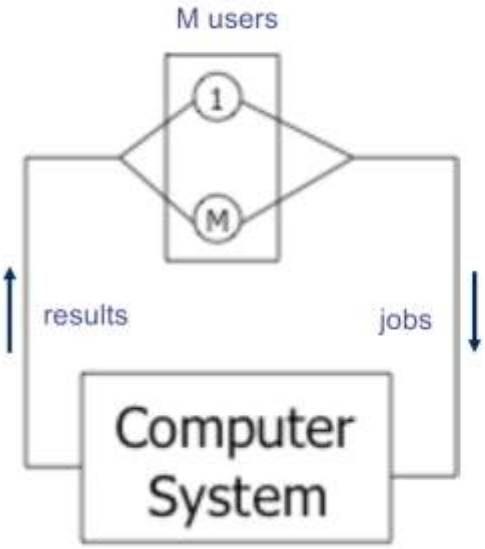
Operational analysis (Continued)

- Using operational law for
  - Performance analysis
  - Bottleneck analysis

Workload characterisation

- Poisson process and its properties

Interactive systems



- An interactive system is used to model the interaction between humans (users) and computers
- The system consists of
  - A number of users
  - A computer system

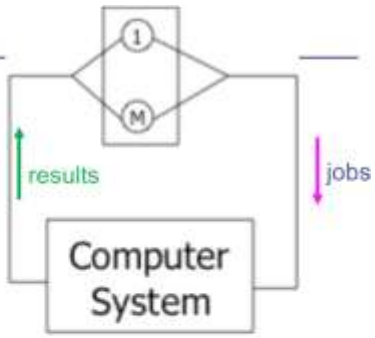
Interactions

- Users send jobs to computer systems
- After finishing processing a job, the computer system returns the result to the user
- A user, after inspecting the results from the computer system, will send another job to the system

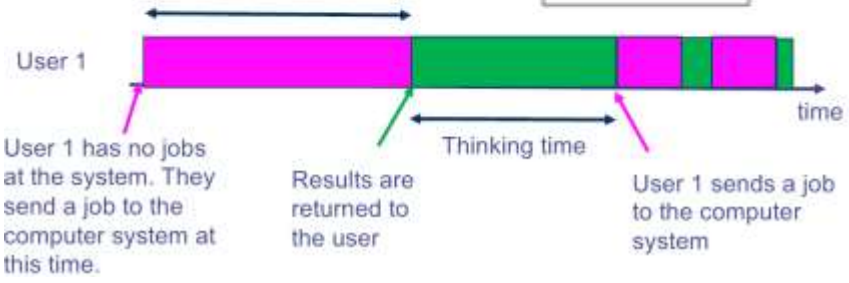
Analyze interactive systems with specific assumptions

- Fixed number of users denoted by M
- Each user can have at most 1 job at the computer system
- Each user goes through a cycle consisting of
  - Thinking time
  - Waiting for result time

Interactive cycle



- The time the job from User 1 spends in the computer system
- User 1 waiting for the results

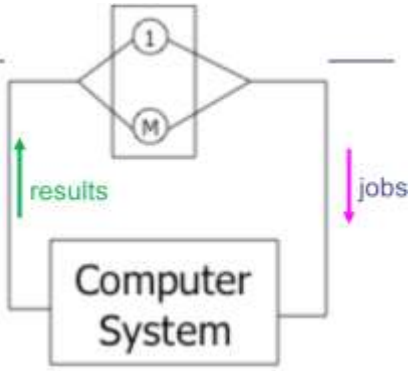


User 1 has no jobs at the system. They send a job to the computer system at this time.

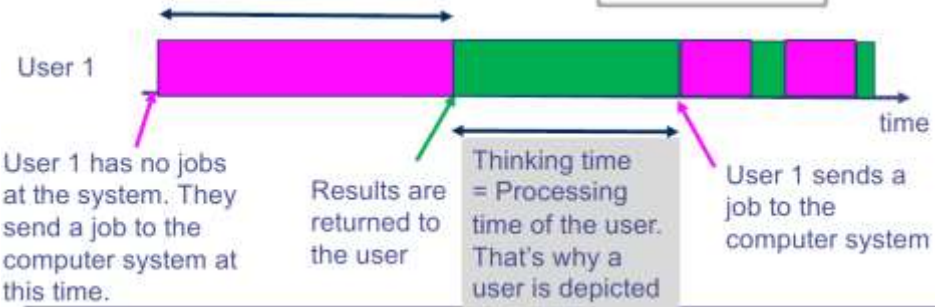
Results are returned to the user

User 1 sends a job to the computer system

Interactive cycle



- User 1's perspective: waiting for the result of their job
- Computer system's perspective: Response time of the job from User 1

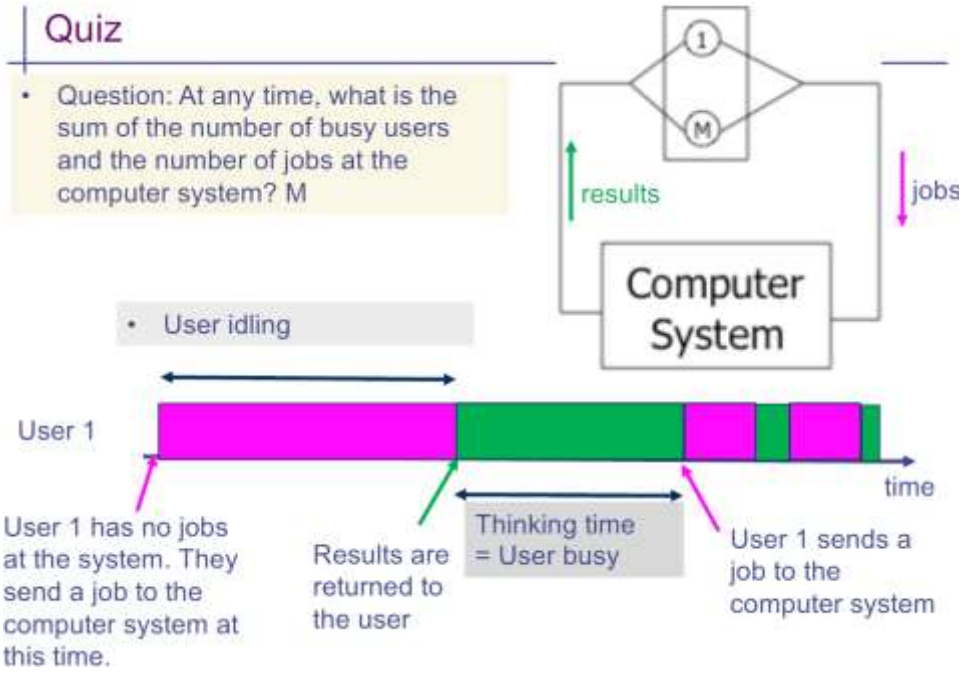


User 1 has no jobs at the system. They send a job to the computer system at this time.

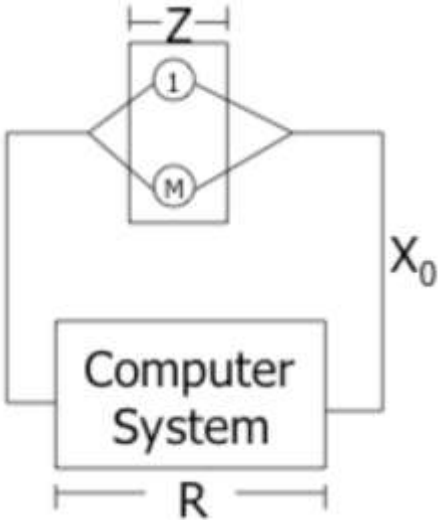
Results are returned to the user

Thinking time = Processing time of the user. That's why a user is depicted as a circle

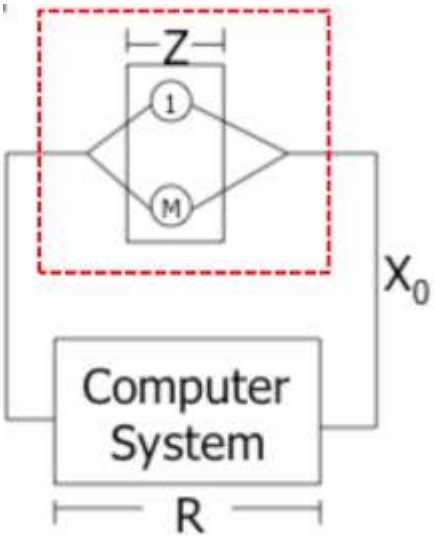
User 1 sends a job to the computer system



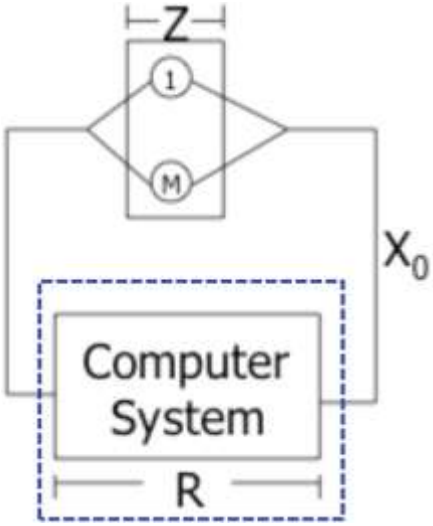
Interactive system: Parameters and analyzing (Thinking time, etc.)



- M interactive users
- Z = mean thinking time
- R = mean response time of the computer system
- X0 = throughput



- Mavg = mean # busy users
- Z = mean thinking time
- X0 = throughput
- Apply Little's Law to the red box. What do you get?
  - Mavg = Z * X0



- Navg = average # jobs in the computer system
- R = mean response time at the computer system
- X0 = throughput
- Apply Little's Law to the computer system (i.e. the blue box), what do you get?
  - Navg = R * X0

Quiz 1: Mavg = X0 * Z

Quiz 2: Navg = X0 * R

What is Mavg + Navg?

- M = Mavg + Navg

Interactive response time law

- M = X0 * (Z+R)

The operational laws

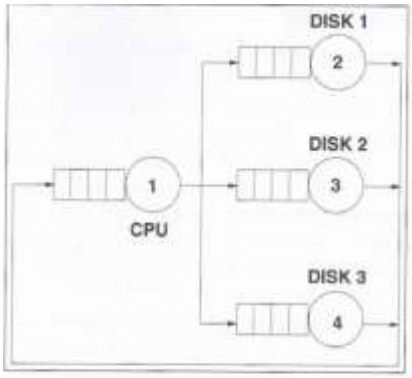
These are the operational laws

- Utilisation law U(j) = X(j) S(j)
- Forced flow law X(j) = V(j) X(0)
- Service demand law D(j) = V(j) S(j) = U(j) / X(0)
- Little's law N = X R
- Interactive response time M = X(0) (R+Z)

Applications

- Mean value analysis (later in the course)
- Bottleneck analysis
- Modification analysis

Bottleneck analysis - motivation



	D(j)	Utilisation
Disk 1	79ms	0.30
Disk 2	108ms	0.41
Disk 3	142ms	0.54
CPU	92ms	0.35

Service demand law: D(j) = U(j) / X(0)
==> U(j) = D(j) X(0)
Utilisation increases with increasing throughput and service demand

- Disk 3 has the highest service demand
- It is the bottleneck of the whole system

Operational law: $X(0) = \frac{U(j)}{D(j)}$

Utilisation limit: $U(j) \leq 1$

$$\Rightarrow X(0) \leq \frac{1}{D(j)}$$

$$X(0) \leq \frac{1}{D(j)}$$ Should hold for all K devices in the system

$$i.e. X(0) \leq \frac{1}{D(1)}, ..., X(0) \leq \frac{1}{D(K)}$$

$$\Rightarrow X(0) \leq \min \frac{1}{D(j)}$$

$$\Rightarrow X(0) \leq \frac{1}{\max D(j)}$$ Bottleneck throughput is limited by the maximum service demand

The maximum system throughput is 1 / 0.142 = 7.04 jobs/s. What if we upgrade Disk 3 by a new disk that is 2 times faster, which device will be the bottleneck after the upgrade? You can assume that service time is inversely proportional to disk speed.

Another throughput bound

- Little's law

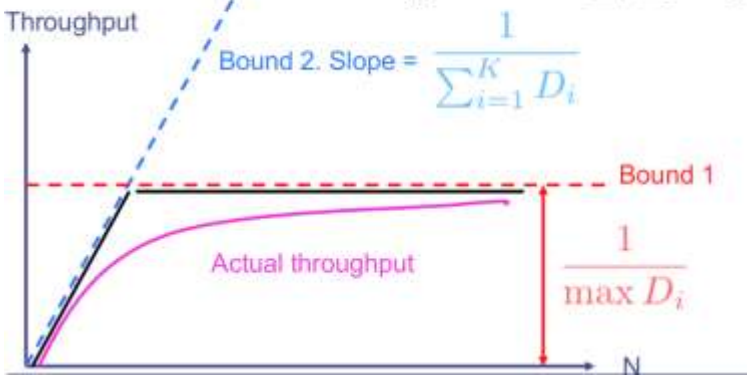
$$N = R \times X(0) \geq (\sum_{i=1}^{K} D_i) \times X(0)$$

$$\Rightarrow X(0) \leq \frac{N}{\sum_{i=1}^{K} D_i}$$

Previously, we have $$X(0) \leq \frac{1}{\max D(j)}$$

Therefore: $$X(0) \leq \min \left[\frac{1}{\max D_i}, \frac{N}{\sum_{i=1}^{K} D_i} \right]$$

$$X(0) \leq \min\left[\frac{1}{\max D_i}, \frac{N}{\sum_{i=1}^{K} D_i}\right]$$



Simple to use

- Needs only utilisation of various components

Assumes service demand is load independent

Modification analysis

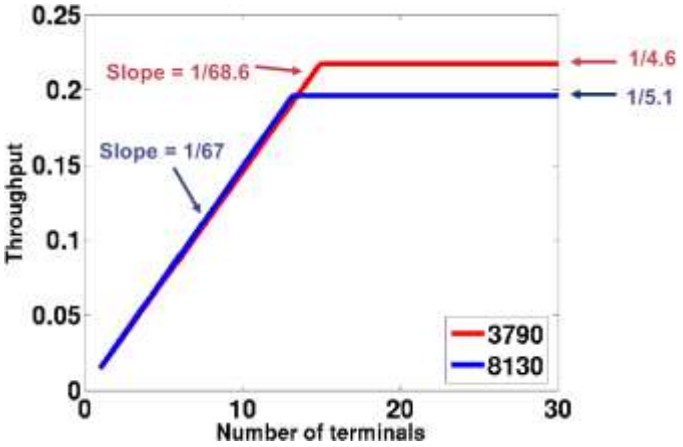
A company currently has a system (3790) and is considering switching to a new system (8130). The service demands for these two systems are given below:

	Service demand (seconds)	
System	CPU	Disk
3790	4.6	4.0
8130	5.1	1.9

The company uses the system for interactive application with a think time of 60s.

Given the same workload, should the company switch to the new system?

Exercise: Answer this question by using bottleneck analysis. For each system, plot the upper bound of throughput as a function of the number of interactive users.



Operational analysis

Operational analysis allows you to bound the system performance but it does NOT allow you to find the throughput and response time of a system

To order to find the throughput and response time, we need to use queueing analysis

To order to use queueing analysis, we need to specify the workload

Workload analysis

Performance depends on workload

- When we look at the performance bound earlier, the bounds depend on number of users and service demand
- Queue response time depends on the job arrival probability distribution and job service time distribution
  - Recall from Lecture 1A:
    - Uniform arrival times and uniform processing times result in zero waiting time
    - But non-uniform distributions give non-zero waiting time

Need to specify workload by using probability distribution.

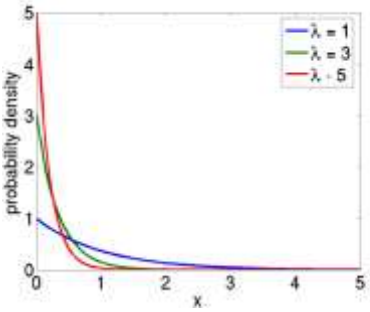
We will look at a well-known arrival process called Poisson process today.

Poisson process has a close relationship with exponential distribution and this is our starting point

Exponential distribution

- A continuous random variable is exponentially distributed with rate λ if it has probability density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

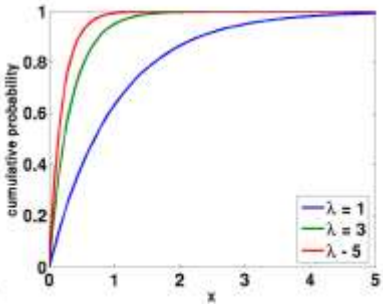


Probability that $x \leq X \leq x + \delta x$ is

$f(x)\,\delta x = \lambda \exp(-\lambda x)\,\delta x$

- The cumulative distribution function F(x) = Prob(X ≤ x) is:

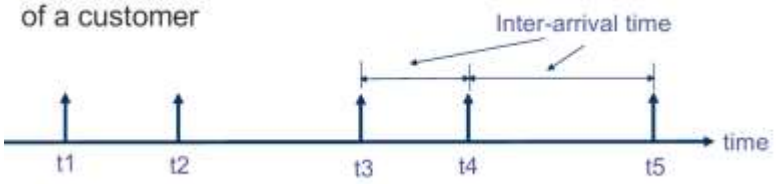
$$F(x) = \int_0^x \lambda e^{-\lambda z} dz = 1 - e^{-\lambda x} \text{ for } x \geq 0$$



What is Prob(X ≥ x)?

Arrival process

- Each vertical arrow in the time line below depicts the arrival of a customer



An arrival can mean

- A telephone call arriving at a call centre
- A transaction arriving at a computer system
- A customer arriving at a checkout counter
- An HTTP request arriving at a web server

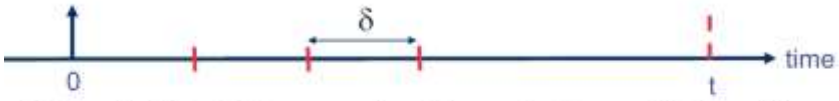
The inter-arrival time distribution will impact on the response time.

We will study an inter-arrival distribution that results from a large number of **independent** customers.

Many independent arrivals

- Assume there is a large pool of N independent customers
- Behaviour of each customer: Within a small time interval of δ, a customer makes a request (or arrives) with a probability of $p\delta$
  - p is a constant
- Quiz: If there are 2 (= N) customers, what is the probability that both of them do not send any request in the time interval δ
  - Answer: $(1 - p\,\delta)^2$
- If a customer arrives at time 0, we want to calculate the probability that the next customer does not arrive before time t
  - Why is this interesting? No arrival before time t = Inter-arrival time is at least t

- Aim: Want to find the probability of no arrivals in [0,t]
- Divide the time t into intervals of width δ



- No arrival in [0,t] = no arrival in each interval δ from N users
- Probability of no arrival in $\delta = (1 - p\,\delta)^N \approx 1 - Np\delta$
- There are t / δ intervals
- Probability of no arrival in [0,t] is

$$(1 - Np\delta)^{\frac{t}{\delta}} \rightarrow e^{-Npt} \text{ as } \delta \rightarrow 0$$

Exponential inter-arrival time

We have showed

Probability(no arrival in [0,t]) = exp(- N p t)

Since we assume that there is an arrival at time 0, this means

Probability(inter-arrival time > t) = exp(- N p t)

This means

Probability(inter-arrival time $\leq$ t) = 1 - exp(- N p t)

What this shows is the inter-arrival time distribution for independent arrival is exponentially distributed

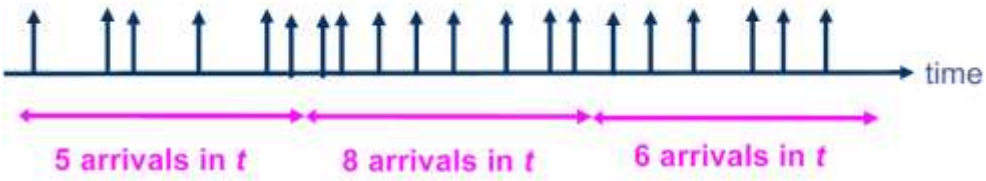
Define: λ = Np

- λ is the mean arrival rate of customers

Two different methods to describe arrivals

Method 1: Continuous probability distribution of inter-arrival time

Method 2: Use a fixed time interval (say t), and count the number of arrivals within t.

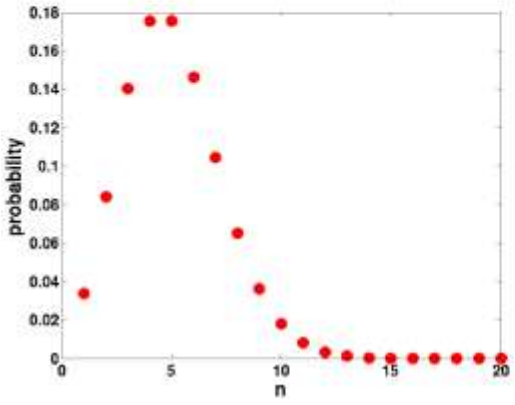


- The number of arrivals in t is random
- The number of arrivals must be an non-negative integer
- We need a discrete probability distribution:
  - Prob[#arrivals in t = 0]
  - Prob[#arrivals in t = 1]
  - etc.

Poisson process

- Definition: An arrival process is Poisson with parameter λ if the probability that n customer arrive in any time interval t is

$$\frac{(\lambda t)^n e^{-\lambda t}}{n!}$$



Example:

Example:
$\lambda = 5$ and t = 1

Note: Poisson is a discrete probability distribution.

Theorem: An exponential inter-arrival time distribution with parameter λ gives rise to a Poisson arrival process with parameter λ

How can you prove this theorem?

- A possible method is to divide an interval t into small time intervals of width δ. A finite δ will give a binomial distribution and with $\delta \rightarrow 0$, we get a Poisson distribution.

Customer arriving rate

- Given a Poisson process with parameter λ, we know that the probability of n customers arriving in a time interval of t is given by:

$$\frac{(\lambda t)^n e^{-\lambda t}}{n!}$$

- What is the mean number of customers arriving in a time interval of t?

$$\sum_{n=0}^{\infty} n \frac{(\lambda t)^n e^{-\lambda t}}{n!} = \lambda t$$

- That's why λ is called the arrival rate.

Customer inter-arrival time

You can also show that if the inter-arrival time distribution is exponential with parameter λ, then the mean inter-arrival time is $1/\lambda$

Quite nicely, we have

Mean arrival rate = 1 / mean inter-arrival time

Application of Poisson process

Poisson process has been used to model the arrival of telephone calls to a telephone exchange successfully

Queueing networks with Poisson arrival is tractable

- We will see that in the next few weeks.

Beware that not all arrival processes are Poisson! Many arrival processes we see in the Internet today are not Poisson. We will see that later.

Poisson distribution & exponential distribution & binomial distribution

从 e 得定义可以推导出：

$$e = \sum_{i=0}^{\infty} \frac{1}{i!}$$

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = \lim_{n->\infty} (1 + \frac{x}{n})^n$$

从二项分布推导泊松分布

二项分布：$P(X = k) = C_n^k \cdot p^k \cdot (1-p)^{n-k}$

抛硬币

以抛硬币为例，p可以表示抛一次硬币，朝上的概率，$P(X = k)$表示抛n次硬币后，k个硬币朝上的概率。

车流量估计

将抛硬币的场景转换为估计车流量的场景：在一个小时内，经过路口A的车辆数目为k的概率$P(X = k)$。车流量场景于抛硬币场景相比，加入了时间约束。如果我们现在继续用二项分布来解决该场景，需要**对时间进行离散化**。

- 假设一个小时平均车流量是λ
- 将一个小时的观测近似为3600次一秒的观测，即$n = 3600$
- 则一秒钟，车辆通过的概率$p = \frac{\lambda}{3600}$ (PS: 此处不够严谨，稍后会介绍)

一小时内，经过路口A的车辆数目为k的概率：

$$P(X = k) = C_{3600}^k \cdot (\frac{\lambda}{3600})^k \cdot (1 - \frac{\lambda}{3600})^{3600-k}$$

然而，$p = \frac{\lambda}{3600}$表示的是一辆汽车在一次观测(我们假设的是一秒)中通过路口A的概率。一秒中，可能会有多辆车经过，因此，时间间隔必须足够小以保证一个时间间隔内只有一辆车通过，即$n \to \infty$。

$$\begin{aligned} P(X = k) &= \lim_{n->\infty} C_n^k \cdot (\frac{\lambda}{n})^k \cdot (1 - \frac{\lambda}{n})^{n-k} \\ &= \lim_{n->\infty} \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{k!} \cdot \frac{\lambda^k}{n^k} \cdot (1 - \frac{\lambda}{n})^n \cdot (1 - \frac{\lambda}{n})^{-k} \\ &= \left\{ \lim_{n->\infty} \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{n^k} \right\} \cdot \left\{ \frac{\lambda^k}{k!} \right\} \cdot \left\{ \lim_{n->\infty} (1 - \frac{\lambda}{n})^n \right\} \cdot \left\{ \lim_{n->\infty} (1 - \frac{\lambda}{n})^{-k} \right\} \\ &= 1 \cdot \frac{\lambda^k}{k!} e^{\lambda} \cdot 1 = \frac{\lambda^k}{k!} e^{\lambda} \end{aligned}$$

日常生活中，大量事件是有固定频率的。

- 某医院平均每小时出生 3 个婴儿
- 某公司平均每 10 分钟接到 1 个电话
- 某超市平均每天销售 4 包 xx 牌奶粉
- 某网站平均每分钟有 2 次访问

它们的特点就是，我们可以预估这些事件的总数，但是没法知道具体的发生时间。已知平均每小时出生 3 个婴儿，请问下一个小时，会出生几个？有可能一下子出生 6 个，也有可能一个都不出生。这是我们没法知道的。

泊松分布就是描述某段时间内，事件具体的发生概率。

$$P(N(t)=n)=\frac{(\lambda t)^n e^{-\lambda t}}{n!}$$

上面就是泊松分布的公式。等号的左边，P 表示概率，N 表示某种函数关系，t 表示时间，n 表示数量，1 小时内出生 3 个婴儿的概率，就表示为 P(N(1) = 3) 。等号的右边，λ 表示事件的频率。

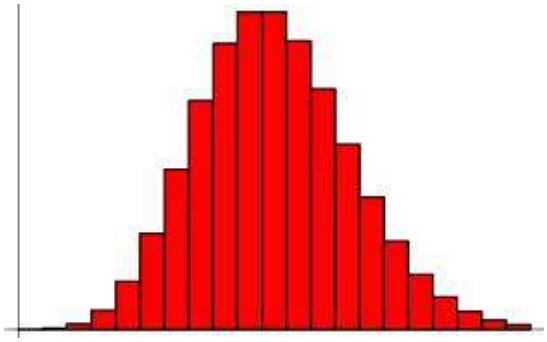
两个小时内，一个婴儿都不出生的概率是 0.25%，基本不可能发生。

$$P(N(2)=0)=\frac{(3\times 2)^0 e^{-3\times 2}}{0!}\approx 0.0025$$

一个小时内，至少出生两个婴儿的概率是 80%。

$$\begin{aligned}P(N(1)\geq 2)&=1-P(N(1)=1)-P(N(1)=0)\\&=1-\frac{(3\times 1)^1 e^{-3\times 1}}{1!}-\frac{(3\times 1)^0 e^{-3\times 1}}{0!}\\&=1-3e^{-3}-e^{-3}\\&=1-4e^{-3}\\&\approx 0.8009\end{aligned}$$

泊松分布的图形大概是下面的样子。



可以看到，在频率附近，事件的发生概率最高，然后向两边对称下降，即变得越大和越小都不太可能。每小时出生 3 个婴儿，这是最可能的结果，出生得越多或越少，就越不可能。

泊松分布使用范围

Poisson 分布主要用于描述在单位时间(空间)中稀有事件的发生数. 即需满足以下四个条件：

1、给定区域内的特定事件产生的次数，可以是根据时间，长度，面积来定义；

2、各段相等区域内的特定事件产生的概率是一样的；

3、各区域内，事件发生的概率是相互独立的；

4、当给定区域变得非常小时，两次以上事件发生的概率趋向于 0。

例如：

1、放射性物质在单位时间内的放射次数；

2、在单位容积充分摇匀的水中的细菌数；

3、野外单位空间中的某种昆虫数等。

泊松分布的期望和方差

泊松分布是一个离散型随机变量分布，其分布律是：

$$P(X=k)=\frac{\lambda^k e^{-\lambda}}{k!}$$

根据离散型随机变量分布的期望定义，泊松分布的期望：

$$E(X)=\sum_{k=0}^{\infty}k\cdot\frac{\lambda^k e^{-\lambda}}{k!}$$

因为k=0时：

$$k\cdot\frac{\lambda^k e^{-\lambda}}{k!}=0$$

所以：

$$E(X)=\sum_{k=1}^{\infty}k\cdot\frac{\lambda^k e^{-\lambda}}{k!}$$

做一下变换：

$$E(X)=\sum_{k=1}^{\infty}k\cdot\frac{\lambda^k e^{-\lambda}}{k!}=\sum_{k=1}^{\infty}\frac{\lambda^k e^{-\lambda}}{(k-1)!}=\sum_{k=1}^{\infty}\frac{\lambda^{k-1}\lambda e^{-\lambda}}{(k-1)!}=\lambda e^{-\lambda}\sum_{k=1}^{\infty}\frac{\lambda^{k-1}}{(k-1)!}$$

这里需要用到泰勒展开式，我们知道常用的泰勒展开式中：

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}+\ldots=\sum_{k=1}^{\infty}\frac{x^{k-1}}{(k-1)!}$$

因此，泊松分布的期望为：

$$E(X)=\lambda e^{-\lambda}\sum_{k=1}^{\infty}\frac{\lambda^{k-1}}{(k-1)!}=\lambda e^{-\lambda}e^{\lambda}=\lambda$$

对于方差$D(X)$，先求出$E(X^2)$：

$$E(X^2)=\sum_{k=0}^{\infty}k^2\cdot\frac{\lambda^k e^{-\lambda}}{k!}=\lambda e^{-\lambda}\sum_{k=1}^{\infty}\frac{k\lambda^{k-1}}{(k-1)!}=\lambda e^{-\lambda}\sum_{k=1}^{\infty}\frac{(k-1+1)\lambda^{k-1}}{(k-1)!}$$

$$=\lambda e^{-\lambda}(\sum_{m=0}^{\infty}\frac{m\cdot\lambda^m}{m!}+\sum_{m=0}^{\infty}\frac{\lambda^m}{m!})(m=k-1)$$

$$=\lambda e^{-\lambda}(\lambda\cdot\sum_{m=1}^{\infty}\frac{\lambda^{m-1}}{(m-1)!}+\sum_{m=0}^{\infty}\frac{\lambda^m}{m!})$$

$$=\lambda e^{-\lambda}(\lambda e^{\lambda}+e^{\lambda})=\lambda(\lambda+1)$$

所以：

$$D(X)=E(X^2)-(E(X))^2=\lambda(\lambda+1)-\lambda^2=\lambda$$

因此，泊松分布的期望和方差为：

$$E(X)=\lambda$$

$$D(X)=\lambda$$

泊松过程的强度 lambda （常数）等于单位长时间间隔内出现的质点数目的期望值。即对泊松分布有：E(X) = D(X) = λ

泊松分布的特征

1、泊松分布是一种描述和分析稀有事件的概率分布。要观察到这类事件，样本含量 n 必须很大。

2、λ 是泊松分布所依赖的唯一参数。λ 值愈小，分布愈偏倚，随着 λ 的增大，分布趋于对称。

3、当 λ = 20 时，分布泊松接近于正态分布；当 λ = 50 时，可以认为泊松分布呈正态分布。在实际工作中，当$\lambda \geq 20$时就可以用正态分布来近似地处理泊松分布的问题。

指数分布是事件的时间间隔的概率。下面这些都属于指数分布。

- 婴儿出生的时间间隔
- 来电的时间间隔
- 奶粉销售的时间间隔
- 网站访问的时间间隔

指数分布的公式可以从泊松分布推断出来。如果下一个婴儿间隔时间 t ，就等同于 t 之内没有任何婴儿出生。

$$P(X>t)=P(N(t)=0)=\frac{(\lambda t)^0 e^{-\lambda t}}{0!}$$

$$=e^{-\lambda t}$$

反过来，事件在时间 t 之内发生的概率（至少出生一个的概率），就是 1 减去上面的值。

$$P(X \leq t) = 1 - P(X > t) = 1 - e^{-\lambda t}$$

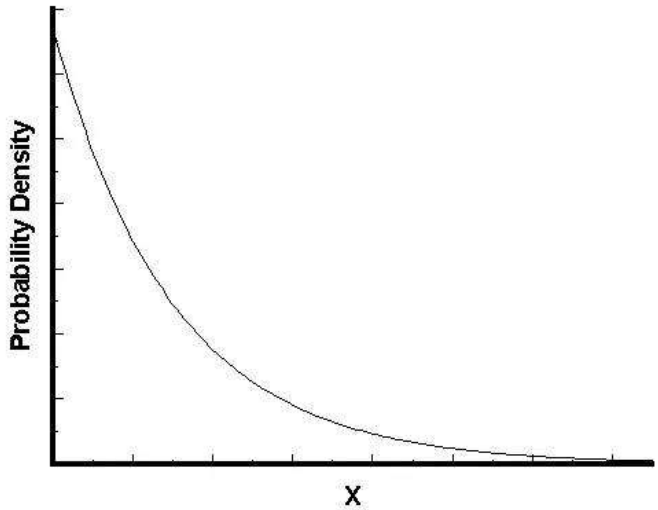
接下来 15 分钟，会有婴儿出生的概率是 52.76%。

$$\begin{aligned} P(X \leq 0.25) &= 1 - e^{-3 \times 0.25} \\ &\approx 0.5276 \end{aligned}$$

接下来的 15 分钟到 30 分钟，会有婴儿出生的概率是 24.92%。

$$\begin{aligned} P(0.25 \leq X \leq 0.5) &= P(X \leq 0.5) - P(X \leq 0.25) \\ &= (1 - e^{-3 \times 0.5}) - (1 - e^{-3 \times 0.25}) \\ &= e^{-0.75} - e^{-1.5} \\ &\approx 0.2492 \end{aligned}$$

指数分布的图形大概是下面的样子。



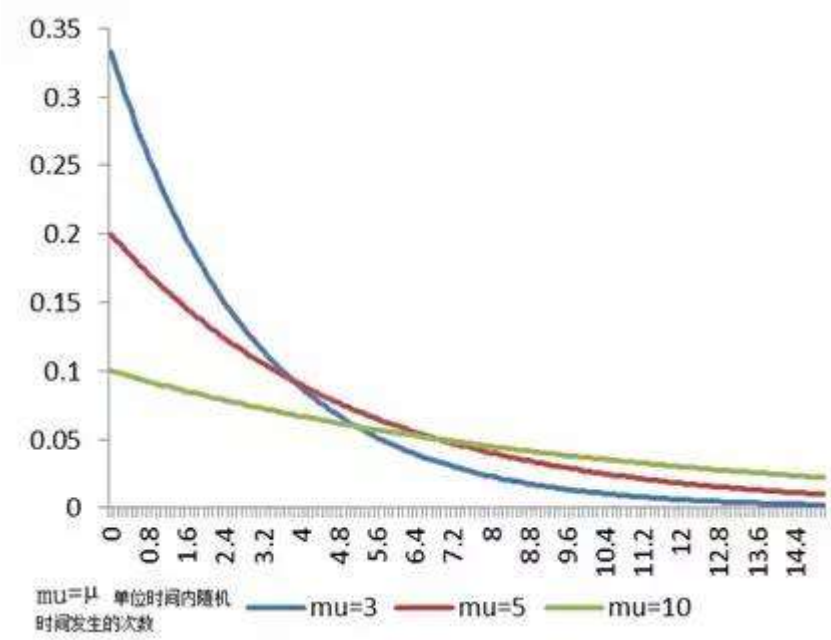
可以看到，随着间隔时间变长，事件的发生概率急剧下降，呈指数式衰减。想一想，如果每小时平均出生 3 个婴儿，上面已经算过了，下一个婴儿间隔 2 小时才出生的概率是 0.25%，那么间隔 3 小时、间隔 4 小时的概率，是不是更接近于 0？

指数分布的概率密度为：

$$f(x)=\begin{cases}\lambda e^{-\lambda x} & x\geq 0,\\ 0, & x<0,\end{cases}$$

式中：x 是给定的时间；λ 为单位时间事件发生的次数；e＝2.71828。

指数分布概率密度曲线如下图：



指数分布具有以下特征：

（1）随机变量 X 的取值范围是从 0 到无穷；

（2）极大值在 x＝0 处，即 f(x)＝λ；

（3）函数为右偏，且随着 x 的增大，曲线稳步递减；

（4）随机变量的期望值和方差为 μ＝1/λ，$\sigma^2=1/\lambda^2$。

通过对概率密度函数的积分，就可以得到相应的概率，其表达式有两种

$P(X\geq x)=e^{-\lambda x}$

$P(X\leq x)=1-e^{-\lambda x}$

从前期的文章《泊松分布》中，我们知道泊松分布的分布律是：

$$P(X(t)=k)=\frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

λ是单元时间内事件发生的次数。如果时间间隔t内事件发生的次数为0，则：

$$P(X>t)=\frac{(\lambda t)^0 e^{-\lambda t}}{0!}=e^{-\lambda t}$$

反过来，在时间间隔t内发生事件的概率，就是1减去上面的值：

$$P(X<=t)=1-e^{-\lambda t}$$

这就变成了时间间隔t在参数λ下的分布函数。根据概率论知识，我们知道，分布函数是概率密度函数从负无穷到正无穷上的积分，对上述的分布函数进行求导，得到：

$$f(t)=\lambda e^{-\lambda t}$$

这就是《指数分布》的概率密度函数。也就是说指数分布是可以从泊松分布推导出来的。

对于指数分布的期望和方差，推导如下：
首先，指数分布属于连续型随机分布，因此，其期望E(X)为：

$$E(X)=\int_{-\infty}^{\infty}|x|f(x)dx=\int_0^{\infty}xf(x)dx=\int_0^{\infty}x\cdot\lambda e^{-\lambda x}dx=\frac{1}{\lambda}\int_0^{\infty}\lambda x e^{-\lambda x}d\lambda x$$

令$u=\lambda x$，则：

$$E(X)=\frac{1}{\lambda}\int_0^{\infty}ue^{-u}du=\frac{1}{\lambda}[(-e^{-u}-ue^{-u})|(\infty,0)]=\frac{1}{\lambda}$$

对于指数分布的方差D(X)：

$$D(X)=E(X^2)-(E(X))^2$$

其中：

$$E(X^2)=\int_{-\infty}^{\infty}|x^2|f(x)dx=\int_0^{\infty}x^2f(x)dx=\int_0^{\infty}x^2\cdot\lambda e^{-\lambda x}dx$$

$$E(X^2)=\frac{1}{\lambda^2}\int_0^{\infty}\lambda x\lambda x e^{-\lambda x}d\lambda x$$

令$u=\lambda x$，则：

$$E(X^2)=\frac{1}{\lambda^2}\int_0^{\infty}u^2e^{-u}du=\frac{1}{\lambda^2}[(-2e^{-u}-2ue^{-u}-u^2e^{-u})|(\infty,0)]=\frac{1}{\lambda^2}\cdot 2=\frac{2}{\lambda^2}$$

所以：

$$D(X)=E(X^2)-(E(X))^2=\frac{2}{\lambda^2}-(\frac{1}{\lambda})^2=\frac{1}{\lambda^2}$$

例：某电视机生产厂生产的电视机平均10年出现大的故障，且故障发生的次数服从泊松分布。

问（1）该电视机使用15年后还没有出现大故障的比例；（2）如果厂家想提供大故障免费维修的质量担保，但不能超过全部产量的20%，试确定提供担保的年数。

解：

（1）设X为电视机出现大故障的时间。已知μ = 10年，则λ = 1/μ = 0.1，于是，$P(X \geq x) = e^{-\lambda x} = e^{-0.1*15} \approx 0.223$。则15年后，没有出现大故障的电视机的占22.3%。

（2）问题要求比例不超过20%，这是求X的右侧概率面积，现在根据公式确定适当的x值。

电视机具有累计出现的故障比例	
担保年数X	累计概率$P(X \leq x) = 1 - e^{-\lambda x}$
1	0.095
2	0.181
3	0.259

从表中可以看到：担保2年时，出现大故障的比例是18.1%，不超过20%。担保3年时，出现大故障的比例为25.9%，已经超过20%。所以，厂家应以2年为担保期。

Week 2B: Queues with Poisson arrivals

Notation:

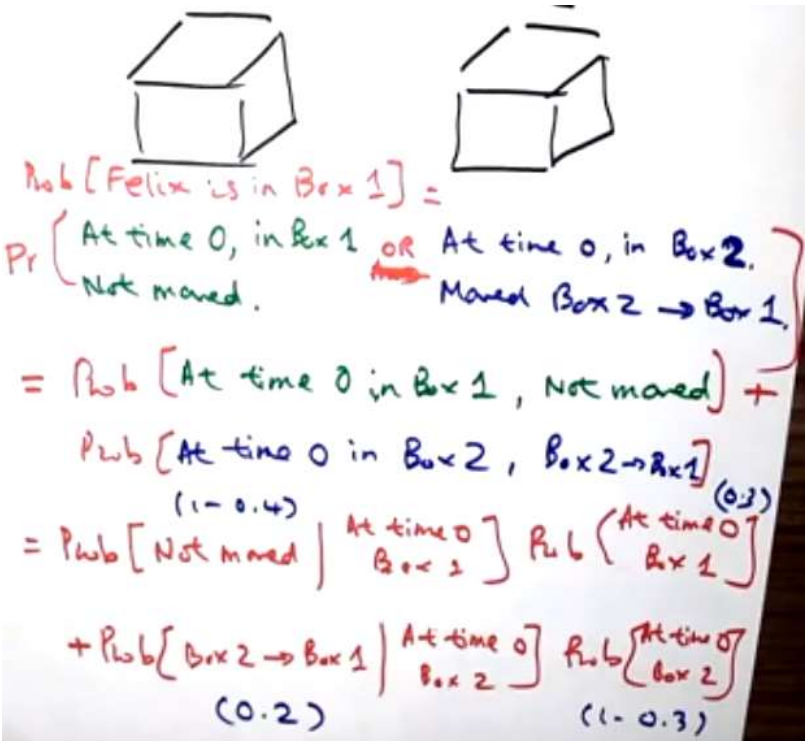
- Prob[A] = probability that event A occurs
- Prob[A | B] = probability that event A occurs given event B

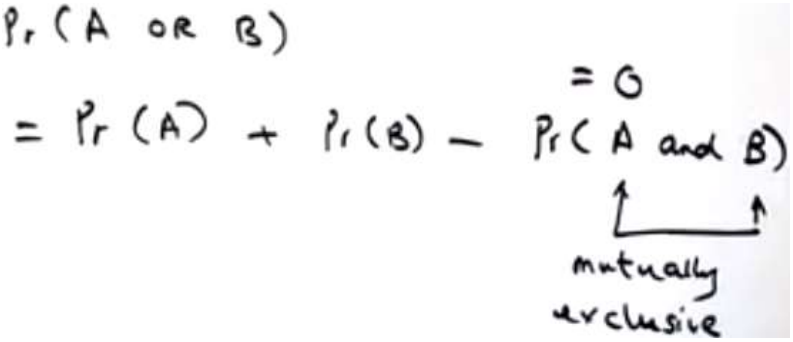
You do know

- Felix is in one of the boxes at times 0 and 1
- Prob[Felix is in Box 1 at time 0] = 0.3
- Prob[Felix will be in Box 2 at time 1| Felix is in Box 1 at time 0] = 0.4
- Prob[Felix will be in Box 1 at time 1| Felix is in Box 2 at time 0] = 0.2

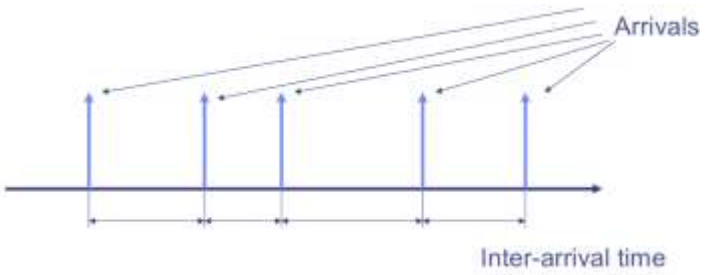
Calculate

- Prob[Felix is in Box 1 at time 1]
- Prob[Felix is in Box 2 at time 1]





Exponential inter-arrival with rate λ



We assume that successive arrivals are independent

Probability that inter-arrival time is between x and $x + \delta x$
$= \lambda \exp(-\lambda x)\, \delta x$

Poisson distribution

- The following are equivalent
  - The inter-arrival time is independent and exponentially distributed with parameter λ
  - The number of arrivals in an interval T is a Poisson distribution with parameter λ

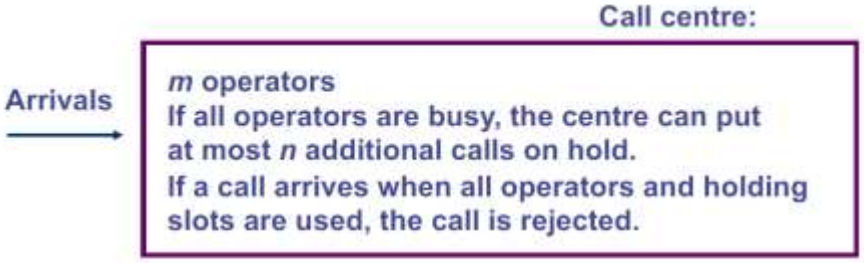
$$Pr[k \text{ arrivals in a time interval } T] = \frac{(\lambda T)^k exp(-\lambda T)}{k!}$$

- Mean inter-arrival time = $1 / \lambda$
- Mean number of arrivals in time interval $T = \lambda T$
- Mean arrival rate = λ

Sample queueing problems

Consider a call centre

- Calls are arriving according to Poisson distribution with rate λ
- The length of each call is exponentially distributed with parameter μ
  - Mean length of a call is $1/\mu$ (in, e.g. seconds)

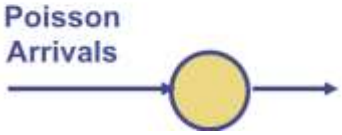


Queueing theory will be able to answer these questions:

- What is the probability that a call is rejected? (This lecture)
- What is the mean waiting time for a call? (Next lecture)

We will start by looking at a call centre with one operator and no holding slot

- This may sound unrealistic but we want to show how we can solve a typical queueing network problem



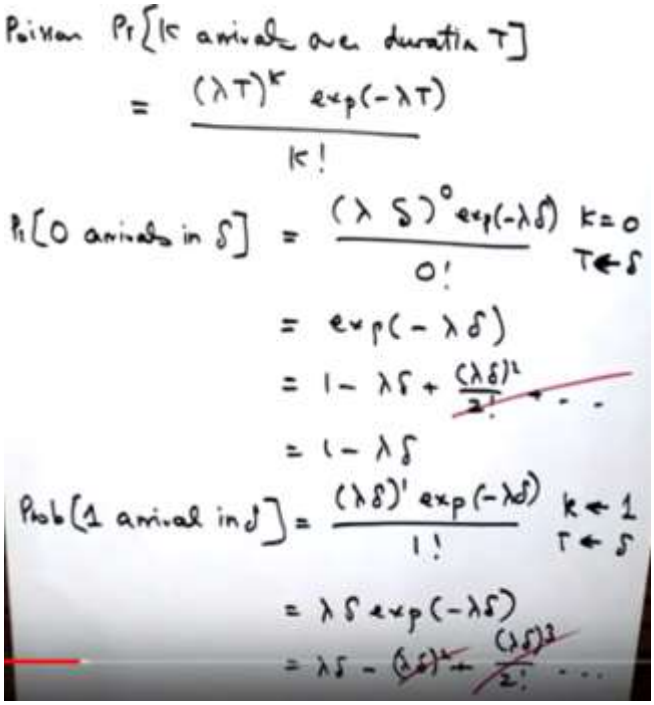
The analysis will consider what happens over a small time interval δ

This is so that we can consider only two possibilities in each time interval

Poisson distribution

- Consider a small time interval δ
  - This means δ^n (for n >= 2) is negligible
- An interpretation of Poisson arrival:
  - Probability [no arrival in δ] = 1 - $\lambda\,\delta$
  - Probability [1 arrival in δ] = $\lambda\,\delta$
  - Probability [2 or more arrivals in δ] $\approx$ 0
- This interpretation can be derived from:

$$Pr[k \text{ arrivals in a time interval } T] = \frac{(\lambda T)^k exp(-\lambda T)}{k!}$$



Service time distribution

- Service time = the amount of processing time a job requires from the server
- We assume that the service time distribution is exponential with parameter μ
  - The probability that the service time is between t and t + δt is:

$$\mu \exp(-\mu t)\ \delta t$$

Here: μ = service rate = 1 / mean service time

Another interpretation of exponential service time:

- Consider a small time interval δ
- Probability [a job will finish its service in next δ seconds] = $\mu\ \delta$
- Probability [a job will **not** finish its service in next δ seconds] = 1 - $\mu\ \delta$

Call centre with 1 operator and no holding slots

Let us see how we can solve the queuing problem for a very simple call centre with 1 operator and no holding slots

What happens to a call that arrives when the operator is busy?

- The call is rejected

What happens to a call that arrives when the operator is idle?

- The call is admitted without delay.

We are interested to find the probability that an arriving call is rejected.

- There are two possibilities for the operator:
  - Busy or
  - Idle
- Let
  - State 0 = Operator is idle (i.e. #calls in the call centre = 0)
  - State 1 = Operator is busy (i.e. #calls in the call centre = 1)

$P_0(t) = \text{Prob. 0 call in the call centre at time } t$

$P_1(t) = \text{Prob. 1 call in the call centre at time } t$

We try to express $P_0(t + \Delta t)$ in terms of $P_0(t)$ and $P_1(t)$

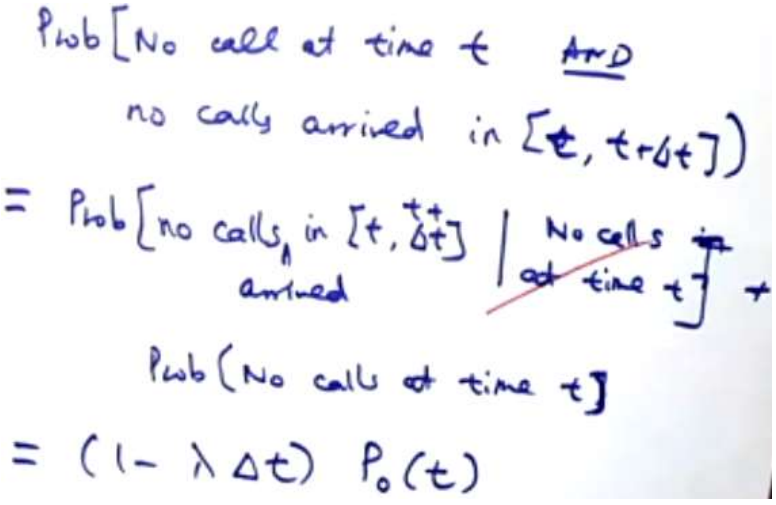
No call at call centre at t + Δt can be caused by

- **No call at time t** and **no call arrives in [t, t + Δt]**, **or**
- **1 call at time t** and **the call finishes in [t, t + Δt]**

$$P_0(t + \Delta t) = P_0(t)(1 - \lambda\Delta t) + P_1(t)\mu\Delta t$$

Question: Why do we NOT have to consider the following possibility:
No customer at time t & 1 customer arrives in [t, t + Δt] &
the call finishes within [t, t + Δt].

所以上面的 question 不用考虑



- Similarly, we can show that

$$P_1(t+\Delta t) = P_0(t)\lambda\Delta t + P_1(t)(1-\mu\Delta t)$$

- If we let $\Delta t \rightarrow 0$, we have

$$\frac{dP_0(t)}{dt} = -P_0(t)\lambda + P_1(t)\mu$$

$$\frac{dP_1(t)}{dt} = P_0(t)\lambda - P_1(t)\mu$$

- We can solve these equations to get

$$P_0(t) = \frac{\mu}{\lambda+\mu} - \frac{\mu}{\lambda+\mu}e^{-(\mu+\lambda)t}$$

$$P_1(t) = \frac{\lambda}{\lambda+\mu} + \frac{\mu}{\lambda+\mu}e^{-(\mu+\lambda)t}$$

This is too complicated, let us look at **steady state** solution

$$P_0 = P_0(\infty) = \frac{\mu}{\lambda+\mu}$$

$$P_1 = P_1(\infty) = \frac{\lambda}{\lambda+\mu}$$

From the steady state solution, we have

- The probability that an arriving call is rejected
- = The probability that the operator is busy
- =

$$P_1 = \frac{\lambda}{\lambda+\mu}$$

Let us check whether it makes sense

- For a constant μ, if the arrival rate rate λ increases, will the probability that the operator is busy go up or down?
- Does the formula give the same prediction?

An alternative interpretation

- We have derived the following equation:

$$P_0(t+\Delta t) = P_0(t)(1-\lambda\Delta t) + P_1(t)\mu\Delta t$$

- Which can be rewritten as:

$$P_0(t+\Delta t) - P_0(t) = -P_0(t)\lambda\Delta t + P_1(t)\mu\Delta t$$

- At steady state:

$$\text{Change in Prob in State } 0 = 0$$

$$\Rightarrow 0 = -P_0\lambda\Delta t + P_1\mu\Delta t$$

Rate of leaving state 0 Rate of entering state 0

Faster way to obtain steady state solution

Transition from State 0 to State 1

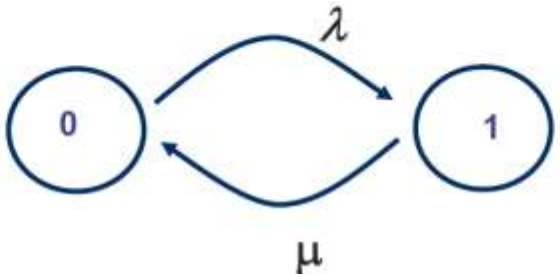
- Caused by an arrival, the rate is λ

Transition from State 1 to State 0

- Caused by a completed service, the rate is μ

State diagram representation

- *Each circle is a state*
- *Label the arc between the states with transition rate*

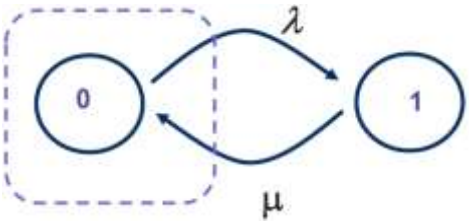


Steady state means

- **rate of transition out of a state = Rate of transition into a state**

We have for state 0:

$$\lambda P_0 = \mu P_1$$



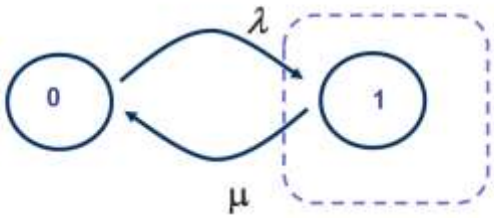
We can do the same for State 1:

Steady state means

- Rate of transition into a state = rate of transition out of a state

We have for state 1:

$$\lambda P_0 = \mu P_1$$



We have one equation $\lambda P_0 = \mu P_1$

We have 2 unknowns and we need one more equation.

Since we must be either one of the two states:

$$P_0 + P_1 = 1$$

Solving these two equations, we get the same steady state solution as before

$$P_0 = \frac{\mu}{\lambda + \mu} \qquad P_1 = \frac{\lambda}{\lambda + \mu}$$

- Change in probability in State 0

$$P_0(t + \Delta t) - P_0(t) = -P_0(t)\lambda\Delta t + P_1(t)\mu\Delta t$$

$$\Rightarrow 0 = -P_0\lambda\Delta t + P_1\mu\Delta t$$

Rate of leaving state 0 **Rate of entering state 0**

$$\Rightarrow 0 = -P_0\lambda\Delta t + P_1\mu\Delta t$$

Prob[Leaving State 0 | State 0] **Prob[Entering State 0 | State 1]**

A call centre with 1 operator and 1 holding slot

We want to determine the probability that an arriving call will be rejected

The system can be in one of the following three states

- State 0 = 0 call in the system (= the operator is idle)
- State 1 = 1 call in the system (= Operator busy. Holding slot empty.)
- State 2 = 2 calls in the system (= Operator busy. Holding slot occupied.)

Define the probability that a certain state occurs

$$P_0 = \text{Probability in State } 0$$

$$P_1 = \text{Probability in State } 1$$

$$P_2 = \text{Probability in State } 2$$

Consider a small time interval δ

- Given the system is in State 1
  - What is the probability that it will move to State 0?
  - What is the probability that it will move to State 2?

Transiting from *State 1* ➔ *State 0*

- This can only occur when a call finishes in time interval δ
- Conditional probability for this to occur = $\mu\,\delta$

Transiting from *State 1* ➔ *State 2*

- This can only occur when a call arrives in time interval δ
- Conditional probability for this to occur = $\lambda\,\delta$

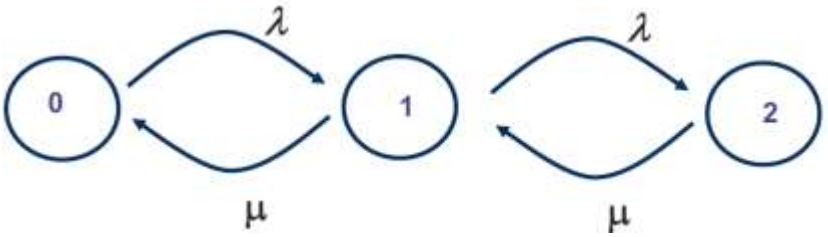
Prob [*State 1* ➔ *State 0* | *State 1*] = $\mu\,\delta$

Prob [*State 1* ➔ *State 2* | *State 1*] = $\lambda\,\delta$

Can you work out the following transition probabilities

- Prob [*State 0* ➔ *State 1* | *State 0*] = $\lambda\,\delta$
- Prob [*State 0* ➔ *State 2* | *State 0*] = *0*
- Prob [*State 2* ➔ *State 0* | *State 2*] = 0
- Prob [*State 2* ➔ *State 1* | *State 2*] = $\mu\,\delta$

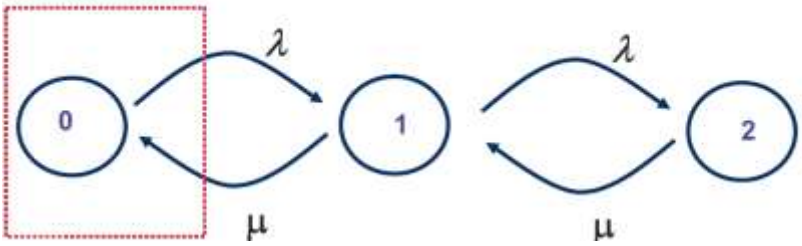
- Given the following transition probabilities (over a small time interval δ)
  - Prob [*State 0* ➔ *State 1* | *State 0*] = $\lambda\,\delta$
  - Prob [*State 0* ➔ *State 2* | *State 0*] = *0*
  - Prob [*State 1* ➔ *State 0* | *State 1*] = $\mu\,\delta$
  - Prob [*State 1* ➔ *State 2* | *State 1*] = $\lambda\,\delta$
  - Prob [*State 2* ➔ *State 0* | *State 2*] = 0
  - Prob [*State 2* ➔ *State 1* | *State 2*] = $\mu\,\delta$
- We draw the following state transition diagram
  - Note 1: We label the arc with transition rate = transition probability / δ
  - Note 2: Arcs with zero rate are not drawn



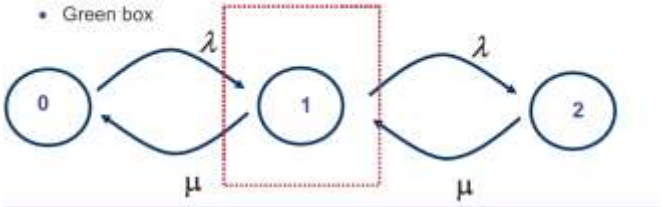
Setting up the balance equations

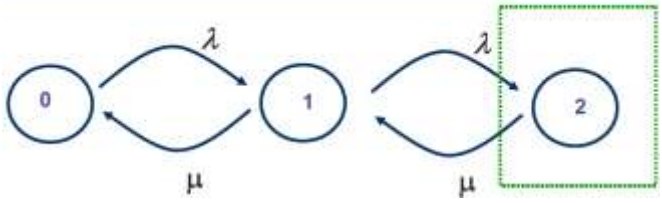
- For steady state, we have
  - Prob of transiting into a "box" = Prob of transiting out of a "box"
  - Rate of transiting into a "box" = Rate of transiting out of a "box"
- Note a "box" can include one or more state
- The "box" is the dotted square shown below

$$\left.\begin{array}{l}\text{Prob out of "box"} = P_0\lambda\delta \\ \text{Prob into "box"} = P_1\mu\delta\end{array}\right\} \Rightarrow \lambda P_0 = \mu P_1$$



- Set up the balance equations for the
  - Red box
  - Green box





- There are three balance equations

$$\begin{aligned} \lambda P_0 &= \mu P_1 \\ \lambda P_0 + \ \mu P_2 &= (\mu + \lambda) P_1 \\ \mu P_2 &= \lambda P_1 \end{aligned}$$

Note that these three equations are not linearly independent

- First equation + Third equation = Second equation

There are 3 unknowns (P_0, P_1, P_2) but we have only 2 equations

We need 1 more equation. What is it?

An addition equation: Sum(Probabilities) = 1

Solve the following equations for the steady state probabilities P_0, P_1, P_2:

$$\lambda P_0 = \mu P_1$$

$$\mu P_2 = \lambda P_1$$

$$P_0 + P_1 + P_2 = 1$$

By solving these 3 equations, we have

- By solving the equations on the previous slide, we have the steady state probabilities are:

$$P_0 = \frac{1}{1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2}$$

$$P_1 = \frac{\frac{\lambda}{\mu}}{1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2}$$

$$P_2 = \frac{\left(\frac{\lambda}{\mu}\right)^2}{1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2}$$

- If we know the values of λ and μ, we can find the numerical values of these probabilities
- Do the expressions make sense?

Week 2A: Revision problems

10. An interactive system has 50 terminals and the user's think time is equal to 5 seconds. The utilization of one of the system's disk was measured to be 60%. The average service time at the disk is equal to 30 msec. Each user interaction requires, on average, 4 I/Os on this disk. What is the average response time of the interactive system?

Question 10

We know that the system response time + think time = number of terminals / system throughput. The system throughput can be derived from the service demand law. The service demand is 4 * 30ms = 120ms. The system throughput equals to utilisation of the device / service demand = 0.6 / 0.12 = 5 user interaction per second. Therefore the system response time is 5s.

Question 1. If the inter-arrival time of requests at a server is exponentially distributed with a mean rate of 20 requests per second, answer the following questions.

a) What is the mean inter-arrival time?
b) Over a duration of 1 minute, what is the mean number of requests arriving at the server?
c) Over a duration of 1 minute, what is the probability of having no arrivals at the server?
d) Over a duration of 1 minute, what is the probability of having 10 arrivals at the server?

Question 1

(a) Since the mean arrival rate is 20 requests per second. The mean inter-arrival time is $\frac{1}{20}$ = 50ms.

(b) The mean number of requests arriving in 1 minute = 20 requests per seconds × 60 seconds / minute = 1200 requests per minute.

(c) and (d) Recalling that for Poisson arrivals with mean arrival rate λ and time interval t, the probability of n arrivals is

$$\frac{(\lambda t)^n \exp(-\lambda t)}{n!}. \tag{1}$$

For this question, $\lambda = 20$ and $t = 60$, so $\lambda t = 1200$.

In order to calculate the probability of no arrivals in a minute, we put $n = 0$ to obtain

$$\exp(-\lambda t) = \exp(-1200) \tag{2}$$

In order to calculate the probability of 10 arrivals in a minute, we put $n = 10$ to obtain

$$\frac{(1200)^{10} \exp(-1200)}{10!} \tag{3}$$

Question 2. This question is about Poisson Process. A server receives requests from two arrival processes. Both arrival processes are Poisson. The rates of these two processes are r_1 and r_2 respectively. Assuming these two processes are independent, prove that the aggregation of these two arrival processes is also Poisson. What is the aggregated arrival rate?

Question 2

In order to refer to the two Poisson processes in a convenient way, I call them P_1 and P_2. The Poisson processes P_1 and P_2, have rates r_1 and r_2, respectively.

Consider a time interval T. Since P_1 is a Poisson process with rate r_1, we know that the probability that there are k arrivals in time interval T is

$$\frac{e^{-r_1 T}(r_1 T)^k}{k!} \tag{4}$$

Similarly, the probability that there are j arrivals in time interval T from P_2 is

$$\frac{e^{-r_2 T}(r_2 T)^j}{j!} \tag{5}$$

Let us consider the aggregation of the two Poisson processes P_1 and P_2 over the time interval T. The arrivals can come from P_1 or P_2. Let us find the probability that there are n arrivals in T. If there are n arrivals from P_1 and P_2 together, this can be resulted from

- 0 arrivals from P_1 and n arrivals from P_2
- 1 arrivals from P_1 and $(n-1)$ arrivals from P_2
- 2 arrivals from P_1 and $(n-2)$ arrivals from P_2

...

- $(n-1)$ arrivals from P_1 and 1 arrivals from P_2
- n arrivals from P_1 and 0 arrivals from P_2

Therefore

Probability that there are n arrivals over time T from P_1 and P_2 together

$$= \sum_{i=0}^{n} \text{Probability of } i \text{ arrivals over time } T \text{ from } P_1 \times \text{Probability of } (n-i) \text{ arrivals over time } T \text{ from } P_2$$

$$= \sum_{i=0}^{n} \frac{e^{-r_1 T}(r_1 T)^i}{i!} \frac{e^{-r_2 T}(r_2 T)^{n-i}}{(n-i)!}$$

$$= \frac{1}{n!} e^{-(r_1+r_2)T} \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} (r_1 T)^i (r_2 T)^{(n-i)}$$

$$= \frac{1}{n!} e^{-(r_1+r_2)T} ((r_1+r_2)T)^n$$

This shows that the aggregation of P_1 and P_2 is a Poisson process with rate $r_1 + r_2$.

最后的复杂的式子是多项式(x+y)^n 展开之后的式子，因此可以化简成(x+y)^n

Week 2B: Revision problems

Question 1

In the lecture, we used queueing analysis to study two call centre problems. In this revision question, you will apply queueing analysis to a call centre with 2 operators and no holding slots. Calls arriving at the call centre are Poisson distributed with a mean arrival rate of λ. Each call requires the processing of one operator. If a call arrives when both operators are idle, the call will be directed to any one of the two operators. If a call arrives when exactly one operator is idle, the call will be directed to the idle operator. If both operators are busy, then the call will me rejected. The mean time of processing a call by any one of the operators is exponentially distributed with rate μ.

You can model this call centre as a system with 3 states: S_0, S_1, S_2 where in state S_k, the number of calls in the system is k.

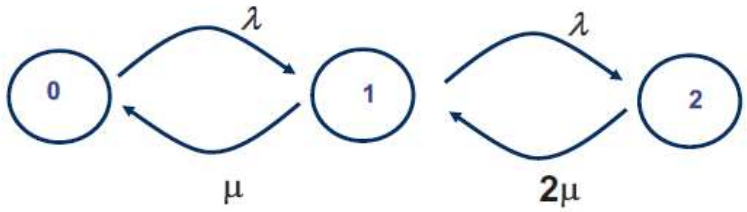
(a) Draw the state transition diagram for this system. Also, label the arcs with the appropriate state transition rate.

In particular, we ask you to pay attention to the transition rate from State 2 to State 1. You can find the answer at the bottom of Page 2 but we ask you to try to find it first. If your answer is different from the the correct answer, you may want to reconsider your mathematical argument.

(b) Write down the state balance equations for States 1, 2 and 3.

Question 1

(a) The state transition diagram is:



The state transition rate from S_2 and S_1 should be 2μ. Note that in order for the state to move from S_2 to S_1, any one of the calls in the call centre has to finish. Consider a small time interval δ,

$$
\begin{aligned}
& \text{Prob[Transiting from } S_2 \text{ to } S_1\text{]} \\
= \; & \text{Prob[Call at Operator 1 has finished OR Call at Operator 2 has finished]} \\
= \; & \text{Prob[Call at Operator 1 has finished]} + \text{Prob[Call at Operator 2 has finished]} - \\
& \text{Prob[Call at Operator 1 has finished AND Call at Operator 2 has finished]} \\
= \; & \mu\delta + \mu\delta - (\mu\delta)^2 \\
\approx \; & 2\mu\delta
\end{aligned}
$$

(b) The state balance equations for States 1, 2 and 3 are respectively:

$$
\begin{aligned}
\lambda P_0 &= \mu P_1 && (1) \\
\mu P_1 + \lambda P_1 &= \lambda P_0 + 2\mu P_2 && (2) \\
\lambda P_1 &= 2\mu P_2 && (3)
\end{aligned}
$$

Week 3A: Queues with Poisson arrivals

Single-server queue

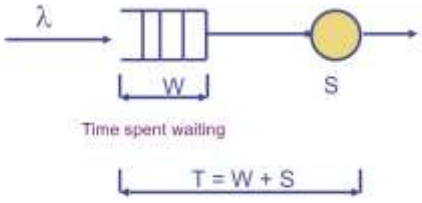
Open, single server queues

How to find:

- Waiting time
- Response time
- Mean queue length etc.

The technique to find waiting time etc. is called *Queueing Theory*

Single Server Queue: Terminology

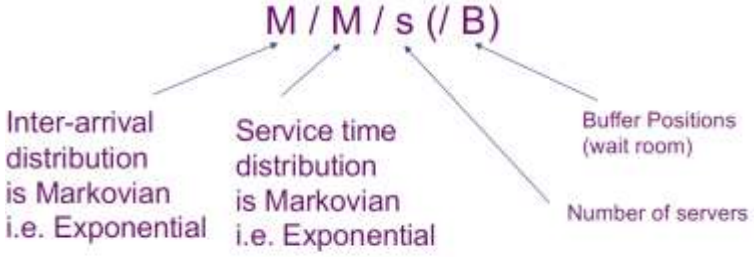


Response Time T
= Waiting time W + Service time S

Note: We use T for response time because this is the notation in many queueing theory books. For a similar reason, we will use ρ for utilisation rather than U.

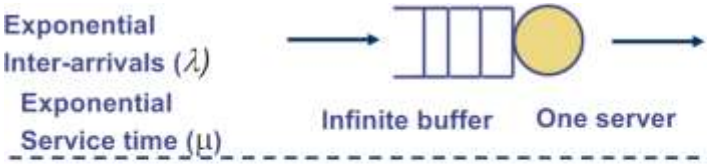
Kendall's notation

- To represent different types of queues, queueing theorists use the Kendall's notation
- The call centre example on the previous page can be represented as:



The call centre example on the last page is a M/M/m/(m+n) queue
If n = ∞, we simply write M/M/m

M/M/1 queue



- Consider a call centre analogy
  - Calls are arriving according to Poisson distribution with rate λ
  - The length of each call is exponentially distributed with parameter μ
    - Mean length of a call is $1/\mu$

Arrivals →

Call centre with *1* operator
If the operator is busy, the centre will put the call on hold.
A customer will wait until his call is answered.

- Queueing theory will be able to answer these questions:
  - What are the mean waiting time, mean response time for a call?

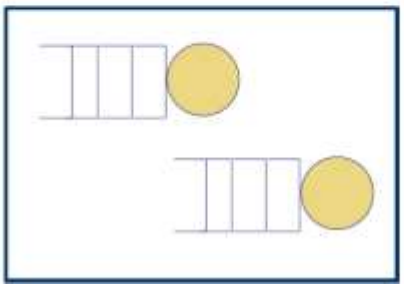
Applicable to any "box" that contains some queues or servers

Mean number of jobs in the "box" =

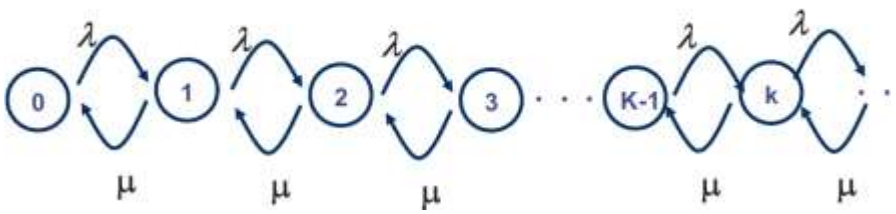
Mean response time x Throughput

We will use Little's Law in this lecture to derive the mean response time

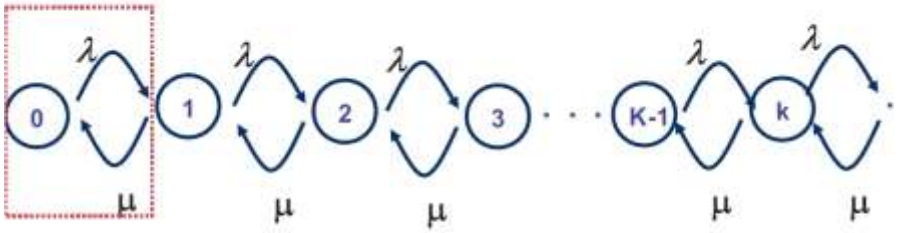
- We first compute the mean number of jobs in the "box" and throughput



- We will solve for the steady state response
- Define the states of the queue
  - State 0 = There is zero job in the system (= The server is idle)
  - State 1 = There is 1 job in the system (= 1 job at the server, no job queueing)
  - State 2 = There are 2 jobs in the system (= 1 job at the server, 1 job queueing)
  - State *k* = There are *k* jobs in the system (= 1 job at the server, *k-1* job queueing)
- The state transition diagram

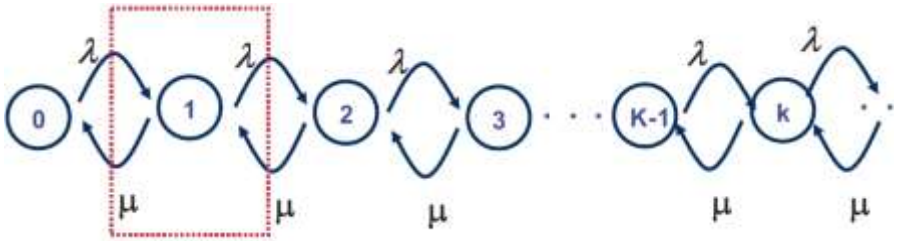


P_k = Prob. k jobs in system



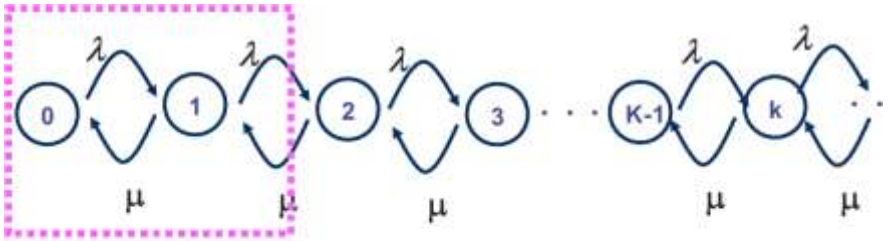
$$\lambda P_0 = \mu P_1$$

$$\Rightarrow P_1 = \frac{\lambda}{\mu} P_0$$



- Exercise: Write the state balance equation for State 1

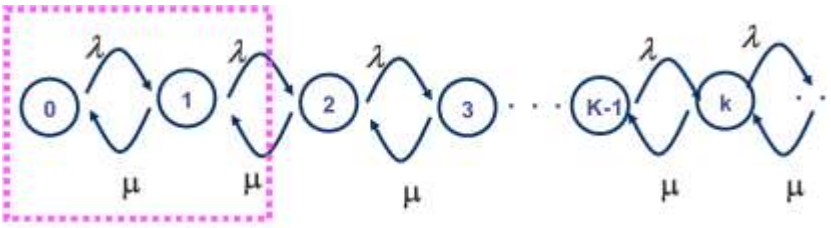
$$\lambda P_0 + \mu P_2 = (\lambda + \mu) P_1$$



- Exercise: Write the state balance equation for magenta box, i.e.

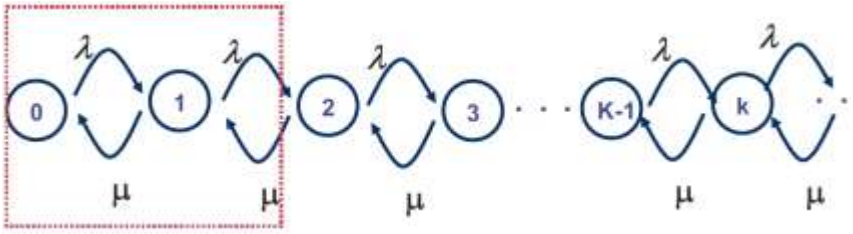
Rate of transiting out of the magenta box
= Rate of transiting into the magenta box

$$\lambda P_1 = \mu P_2$$



State balance for State 1 $\lambda P_0 + \mu P_2 = (\lambda + \mu) P_1$

State balance for State 1 and State 2 combined $\lambda P_1 = \mu P_2$



$$\lambda P_0 = \mu P_1 \qquad \lambda P_1 = \mu P_2$$

$$\Rightarrow P_2 = \frac{\lambda}{\mu} P_1 \quad \Rightarrow P_2 = \left(\frac{\lambda}{\mu}\right)^2 P_0$$

In general $P_k = \left(\frac{\lambda}{\mu}\right)^k P_0$

Let $\rho = \frac{\lambda}{\mu}$

We have $P_k = \rho^k P_0$

With $P_k = \rho^k P_0$ and

$$P_0 + P_1 + P_2 + P_3 + ... = 1$$

$$\Rightarrow (1 + \rho + \rho^2 + ...)P_0 = 1$$

$$\Rightarrow P_0 = 1 - \rho \text{ if } \rho < 1$$

$$\Rightarrow P_k = (1 - \rho)\rho^k$$

ρ = utilisation
= Prob server is busy
= 1 - P_0
= 1- Prob server is idle

Since $\rho = \frac{\lambda}{\mu}$ $, \rho < 1 \Rightarrow \lambda < \mu$

Arrival rate < service rate

上面 prob server is busy 其实就是使用率（utilization）的意思，并且可以用等比数列求出它的值

Geometric progression

$$S_n = \frac{a_1(1-q^n)}{1-q} \ (q \neq 1)$$

Recall that $P_k = \text{Prob. } k \text{ jobs in system}$

and we have calculated that $P_k = (1 - \rho)\rho^k$

Determine the mean number of jobs in the system.

Hint 1: Look at pre-lecture exercise 1.

You can use the following formula to help you.

For $0 \leq x < 1$,

$$p + x(p+q) + x^2(p+2q) + x^3(p+3q) + ... = \frac{p}{1-x} + \frac{xq}{(1-x)^2}$$

Mean number of jobs

$$P_k = \text{Prob. } k \text{ jobs in system}$$
$$P_k = (1-\rho)\rho^k$$

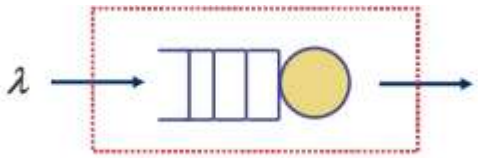
The mean number of jobs in the system =

$$\sum_{k=0}^{\infty} kP_k = \sum_{k=0}^{\infty} k(1-\rho)\rho^k$$

$$= \frac{\rho}{1-\rho}$$

用上面给的公式，讲 p=0, x=p(字母 ro)，以及 q=1-p(字母 ro) ro 带进去，就可以得到这个结果

M/M/1: mean response time



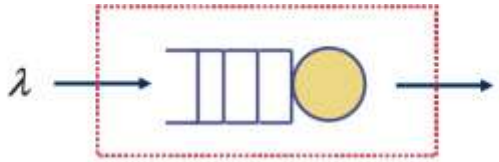
Little's law:
mean number of customers = throughput x response time

Throughput is λ *(why?)*

$$\text{Response time } T = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu-\lambda}$$

因为是平衡状态，所以 throughput 等于他的到达率

M/M/1 mean waiting time



What is the mean waiting time at the queue?

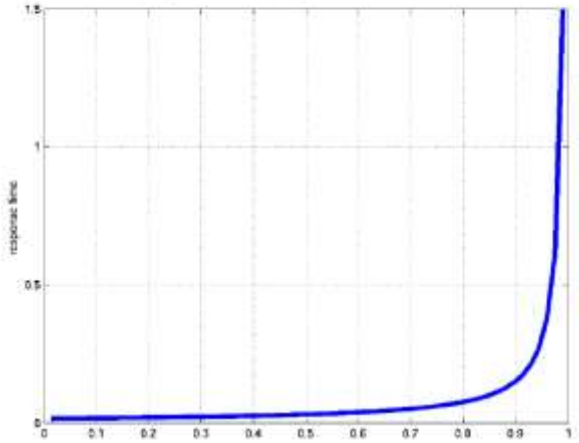
Mean waiting time = mean response time - mean service time

We know mean response time (from last slide)

Mean service time is = 1 / μ

Using the service time parameter (1/μ = 15ms) in the example, let us see how response time T varies with λ

$$T = \frac{1}{\mu(1-\rho)}$$



Observation: Response time increases sharply when ρ gets close to 1

Infinite queue assumption means $\rho \rightarrow 1$, $T \rightarrow \infty$

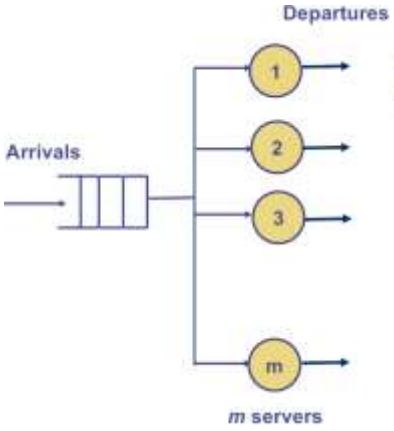
Non-linear effect on response time

- The response time of an M/M/1 queue $= \frac{1}{\mu - \lambda}$
- Assuming the mean arrival rate is 10 requests/s
- We will calculate the effect of service rate on response time
- Complete the following table and see what you can conclude

Service rate	Utilisation λ/μ	Response time
11	10/11 = 0.909	1
22	10/22 = 0.454	0.08

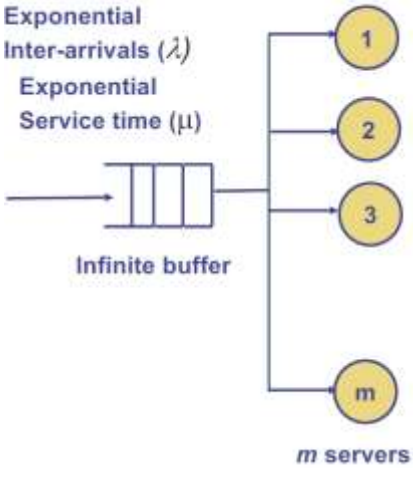
- Doubling the service rate can sometimes reduce by response time by a factor more than 2.

Multiple server queue



- Open, multi-server queue
- How to find:
  - Waiting time
  - Response time
  - Mean queue length etc.

Multi-server queues M/M/m



All arrivals go into one queue.

Customers can be served by any one of the *m* servers.

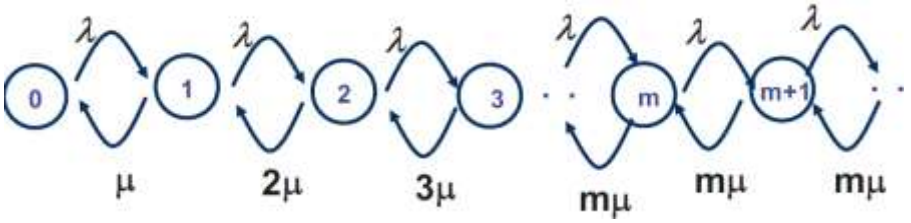
When a customer arrives
- If all servers are busy, it will join the queue
- Otherwise, it will be served by one of the available servers

A call centre analogy of M/M/m queue

- Consider a call centre analogy
  - Calls are arriving according to Poisson distribution with rate λ
  - The length of each call is exponentially distributed with parameter μ
    - Mean length of a call is $1/\mu$

Arrivals →

Call centre with *m* operators
If all *m* operators are busy, the centre will put the call on hold.
A customer will wait until his call is answered.

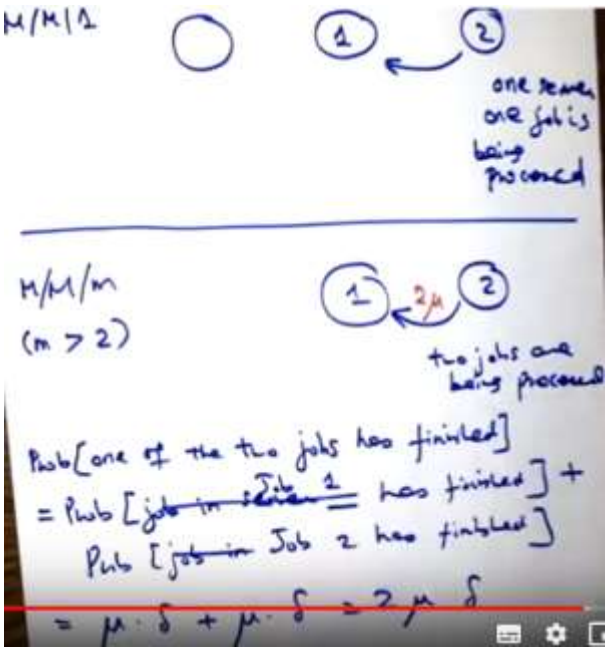


Following the same method, we have mean response time T is

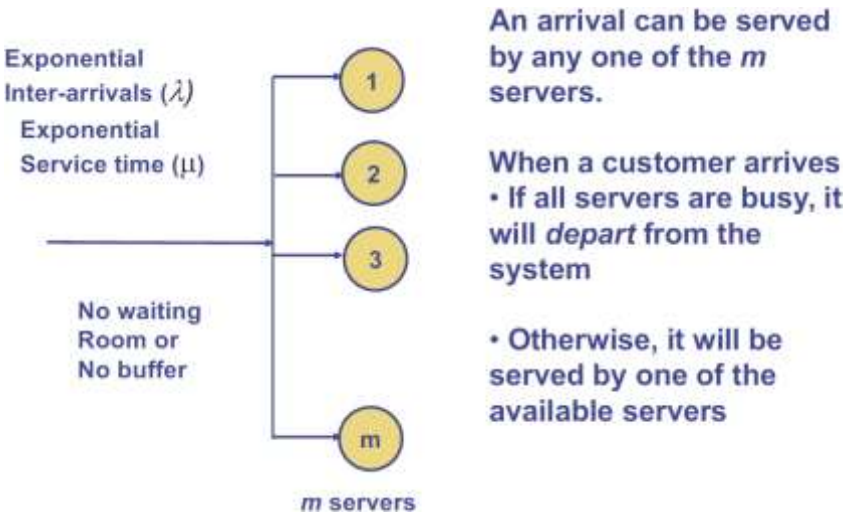
$$T = \frac{C(\rho, m)}{m\mu(1-\rho)} + \frac{1}{\mu}$$

where $$\rho = \frac{\lambda}{m\mu}$$

$$C(\rho, m) = \frac{\frac{(m\rho)^m}{m!}}{(1-\rho)\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}}$$



Multi-server queues M/M/m/m with no waiting room

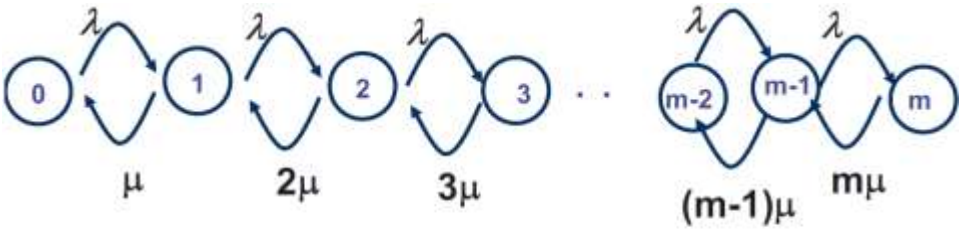


A call centre analogy of M/M/m/m queue

- Consider a call centre analogy
  - Calls are arriving according to Poisson distribution with rate λ
  - The length of each call is exponentially distributed with parameter μ
    - Mean length of a call is $1/\mu$

Arrivals →

Call centre with *m* operators
If all *m* operators are busy, the call is dropped.

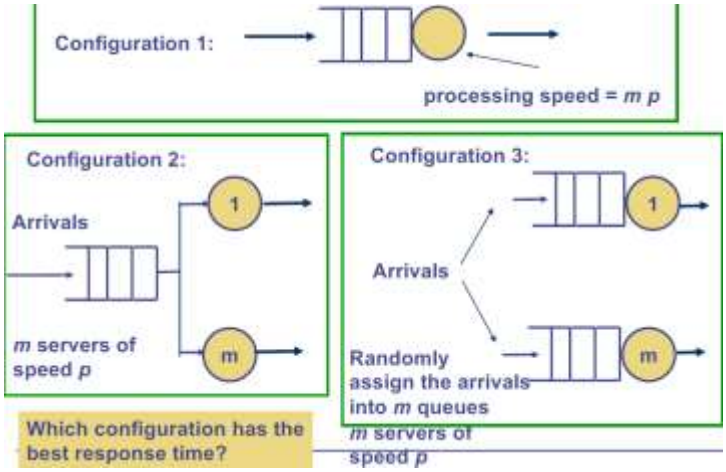


Probability that an arrival is blocked
= Probability that there are m customers in the system

$$P_m = \frac{\frac{\rho^m}{m!}}{\sum_{k=0}^{m} \frac{\rho^k}{k!}} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

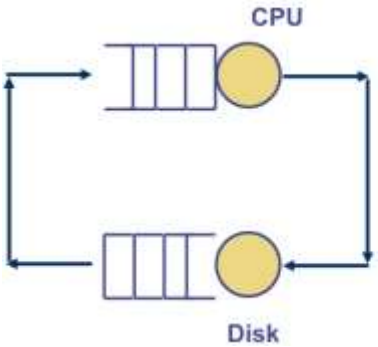
"Erlang B formula"

What will you be able to do with the results



Week 3B: Markov Chain

- You can use Markov Chain to analyse
  - Closed queueing network (see example below)
  - Reliability problem



- There are *n* jobs in the closed system
- What is the response time of one job?
- What is the response time if we replace the CPU with one that is twice as fast?

- The state-transition model that we have used is called a continuous-time Markov chain
  - There is also discrete-time Markov chain
- The transition from a state of the Markov chain to another state is characterised by an exponential distribution
  - E.g. The transition from State p to State q is exponential with rate r_{pq}, then consider a small time interval δ
  - Prob [Transition from State p to State q in time δ | State p] = $r_{pq}\,\delta$

A Markov chain can be solved by

- Identifying the states
- Find the transition rate between the states
- Solve the steady state probabilities

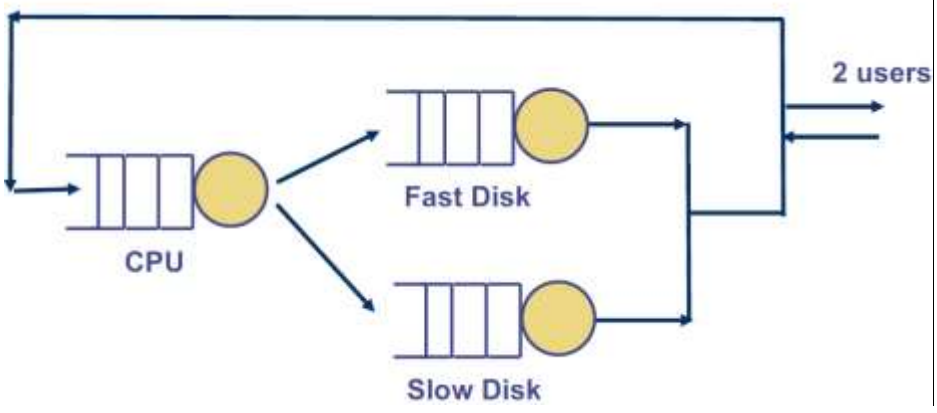
You can then use the steady state probabilities as a stepping stone to find the quantity of interest (e.g. response time etc.)

We will study two Markov chain problems in this lecture:

- Problem 1: A Database server
- Problem 2: Data centre reliability problem

Problem 1: A DB server

- A database server with a CPU, a fast disk and a slow disk
- At peak demand, there are always two users in the system
- Transactions alternate between the CPU and the disks
- The transactions will equally likely find the file on either disk



Fast disk is twice as fast as the slow disk
Typical transactions take on average 10s CPU time
Fast disk takes on average 15s to serve all files for a transactions
Slow disk takes on average 30s to serve all files for a transactions
The time that each transaction requires from the CPU and the disks is exponentially distributed

Typical capacity planning questions

What response time can a typical user expect?
What is the utilisation of each of the system resources?
How will performance parameters change if number of users are doubled?
If fast disk fails and all files are moved to slow disk, what will be the new response time?

Choice of states

Use a 2-tuple (A,B) where

- A is the location of the first user
- B is the location of the second user
- A, B are drawn from {CPU,FD,SD}
  - FD = fast disk, SD = slow disk
- Example states are:
  - (CPU,CPU): both users at CPU
  - (CPU, FD): 1st user at CPU, 2nd user at fast disk
- Total 9 states

Question: If there are *n* users,

- What are the states?
- How many states are there?

We use a 3-tuple (X,Y,Z)

- X is # users at CPU
- Y is # users at fast disk
- Z is # users at slow disk

Examples

- (2,0,0): both users at CPU
- (1,0,1): one user at CPU and one user at slow disk

There are six possible states. Can you list them?

- (2,0,0) (1,1,0) (1,0,1) (0,2,0) (0,1,1) (0,0,2)

If there are n users, how many states are there?

$$\frac{(n+1)(n+2)}{2}$$

Choice #2 requires less #states but loses certain information.

Identifying state transitions

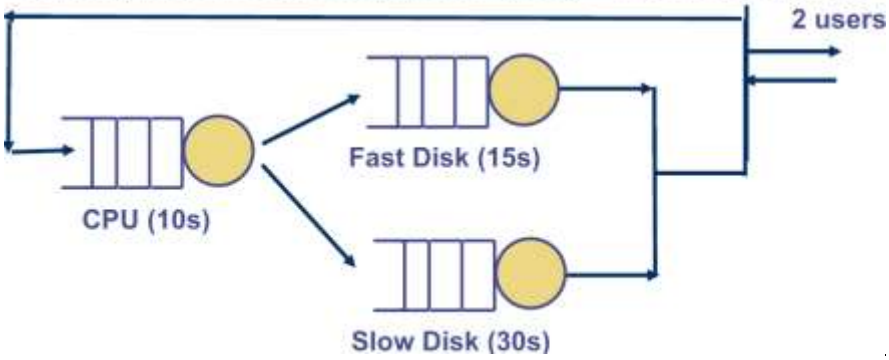
A state is: (#users at CPU, #users at fast disk, #users at slow disk)

What is the rate of moving from State (2,0,0) to State (1,1,0)?

- This is caused by a job finishing at the CPU and move to fast disk
- Jobs complete at CPU at a rate of 6 transactions/minute
- Half of the jobs go to the fast disk

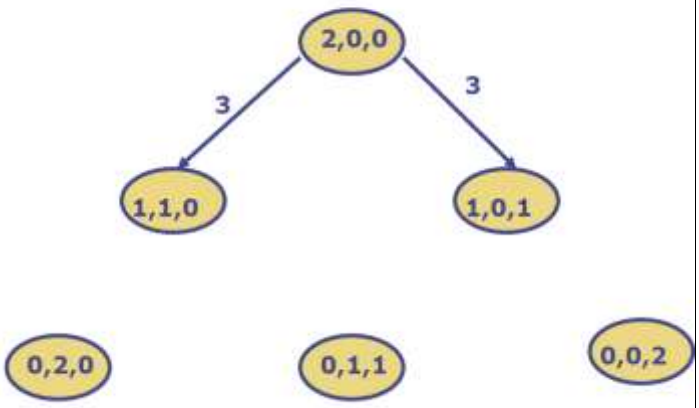
Transition rate from (2,0,0) ➔ (1,1,0) = 3 transactions/minute

Similarly, transition rate from (2,0,0) ➔ (1,0,1) = 3 transactions/minute



Transition rate from (2,0,0) ➔ (1,1,0) = 3 transactions/minute

Transition rate from (2,0,0) ➔ (1,0,1) = 3 transactions/minute

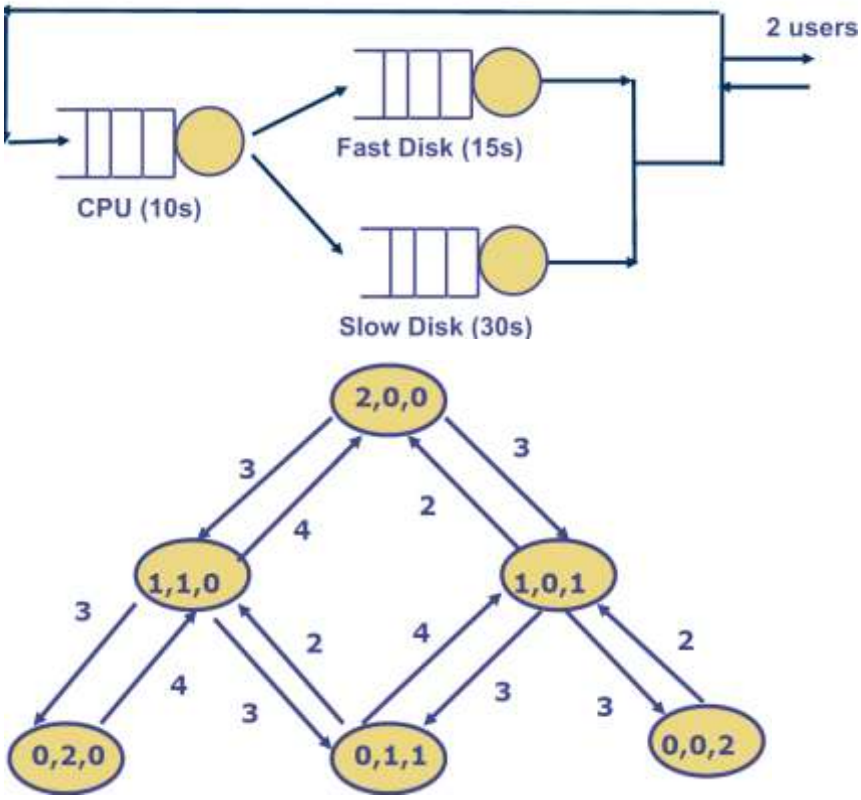


Question: What is the transition rate from (2,0,0) ➔ (0,1,1)?

From (1,1,0) there are 3 possible transitions

- Fast disk user goes back to CPU (2,0,0)
- CPU user goes to the fast disk (0,2,0), or
- CPU user goes to the slow disk (0,1,1)

Question: What are the transition rates in number of transactions per minute?



Balance Equations

Define

$P_{(2,0,0)}$ = Probability in state (2,0,0)

$P_{(1,1,0)}$ = Probability in state (1,1,0) etc.

Exercise: Write down the balance equation for state (2,0,0)

- You can write one flow balance equation for each state:

$$6P_{(2,0,0)} - 4P_{(1,1,0)} - 2P_{(1,0,1)} + 0P_{(0,2,0)} + 0P_{(0,1,1)} + 0P_{(0,0,2)} = 0$$

$$-3P_{(2,0,0)} + 10P_{(1,1,0)} + 0P_{(1,0,1)} - 4P_{(0,2,0)} - 2P_{(0,1,1)} + 0P_{(0,0,2)} = 0$$

$$-3P_{(2,0,0)} + 0P_{(1,1,0)} + 8P_{(1,0,1)} + 0P_{(0,2,0)} - 4P_{(0,1,1)} - 2P_{(0,0,2)} = 0$$

$$0P_{(2,0,0)} - 3P_{(1,1,0)} + 0P_{(1,0,1)} + 4P_{(0,2,0)} + 0P_{(0,1,1)} + 0P_{(0,0,2)} = 0$$

$$0P_{(2,0,0)} - 3P_{(1,1,0)} - 3P_{(1,0,1)} + 0P_{(0,2,0)} + 6P_{(0,1,1)} + 0P_{(0,0,2)} = 0$$

$$0P_{(2,0,0)} + 0P_{(1,1,0)} - 3P_{(1,0,1)} + 0P_{(0,2,0)} + 0P_{(0,1,1)} + 2P_{(0,0,2)} = 0$$

- However, there are only 5 linearly independent equations.
- Need one more equation:

$$P_{(2,0,0)} + P_{(1,1,0)} + P_{(1,0,1)} + P_{(0,2,0)} + P_{(0,1,1)} + P_{(0,0,2)} = 1$$

You can find the steady state probabilities from 6 equations

- It's easier to solve the equations by a software packages, e.g
  - Matlab, Octave, Python etc.

The solutions are:

- $P_{(2,0,0)} = 0.1391$
- $P_{(1,1,0)} = 0.1043$
- $P_{(1,0,1)} = 0.2087$
- $P_{(0,2,0)} = 0.0783$
- $P_{(0,1,1)} = 0.1565$
- $P_{(0,0,2)} = 0.3131$

I used Matlab to solve these equations

- The file is "data_server.m" (can be downloaded from the course web site)

How can we use these results for capacity planning?

Model interpretation

Response time of each transaction

- Use Little's Law R = N/X with N = 2
  - For this system:
    - System throughput = CPU Throughput
  - Throughput = Utilisation x Service rate
    - Recall Utilisation = Throughput x Service time (From Lecture 1B)
  - CPU utilisation (using states where there is a job at CPU): $P_{(2,0,0)} + P_{(1,1,0)} + P_{(1,0,1)} = 0.452$
  - Throughput = 0.452 x 6 = 2.7130 transactions / minute
  - Response time (with 2 users) = 2 /2.7126 = 0.7372 minutes per transaction

Sample capacity planning problem

What is the response time if the system has up to 4 users instead of 2 users only?

- You can't use the previous Markov chain
- You need to develop a new Markov chain
  - The states are again (#users at CPU, #users at fast disk, #users at slow disk)
  - States are (4,0,0), (3,1,0), (1,2,1) etc.
  - There are 15 states
  - Determine the transition rates
  - Write down the balance equations and solve them.
  - Use the steady state probabilities and Little's Law to determine the new response time
  - You can do this as an exercise
  - Throughput = 3.4768 (up 28%), response time = 60.03 seconds (up 56%)

Computation aspect of Markov chain

This example shows that when there are a large number of users, the burden to build a Markov chain model is large

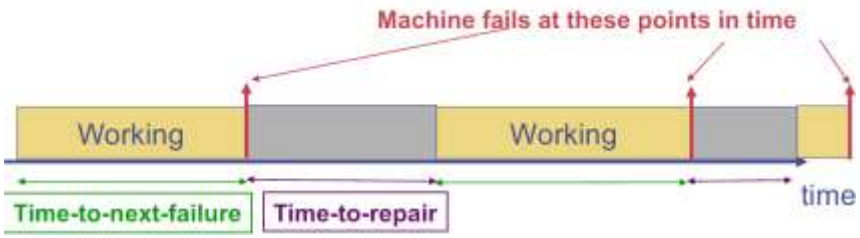
- 15 states
- Many transitions
- Need to solve 15 equations in 15 unknowns

Is there a faster way to do this?

- Yes, we will look at Mean Value Analysis in a few weeks and it can obtain the response time much more quickly

Machine working-repair cycle

- A data centre consists of a number of machines
- Machines can fail and have to be repaired
- Terminology:
  - Time-to-next failure: From the time a machine has been fixed to the time it next fails
  - Time-to-repair = time waiting in the repair queue + service time to repair the machine
    - Time-to-repair is a response time



Data centre reliability problem

Example: A data centre has 10 machines

- Each machine may go down
  - Time-to-next-failure is exponentially distributed with mean 90 days
- Service time to repair is exponentially distributed with mean 6 hours

Capacity planning question:

- Can I make sure that at least 8 machines are available 99.9999% of the time?
- What is the probability that at least *6* machines are available?
- How many repair staff are required to guarantee that at least *k* machines are available with a given probability?
- What is the mean-time-to-repair (MTTR) a machine?
  - Note: Mean-time-to-repair includes waiting time at the repair queue.

Data centre has

- *M* machines
- *N* staff maintain and repair machine
- Assumption: $M > N$

Automatic diagnostic system

- Check "heartbeat" by "ping" (Failure detection)
- Staff are informed if failure is detected

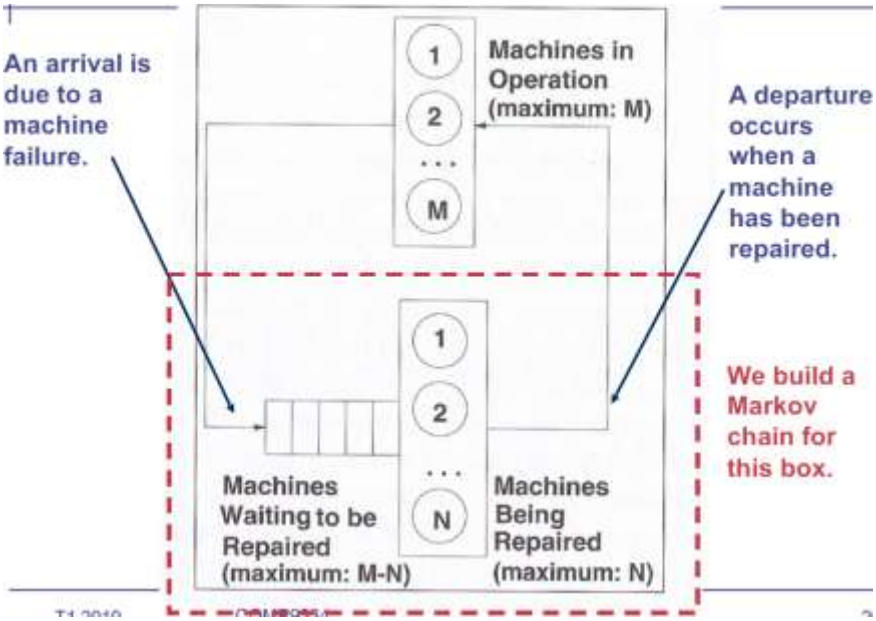
Repair work

- If a machine fails, any one of the idle repair staff (if there is one) will attend to it.
- If all repair staff are busy, a failed machine will need to wait until a repair staff has finished its work

This is a queueing problem solvable by Markov chain!!!

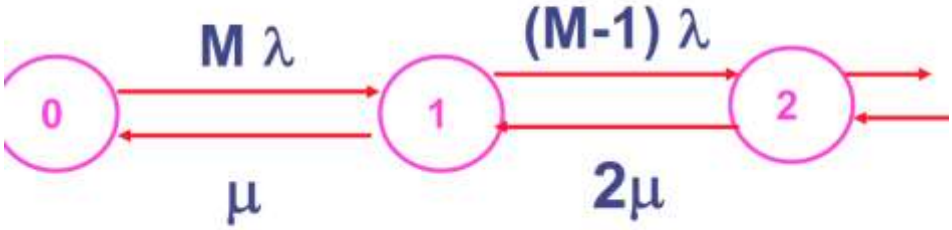
Let us denote

- λ = ***1 / Mean-time-to-failure***
- μ = **1/ Mean service time to repair a machine**



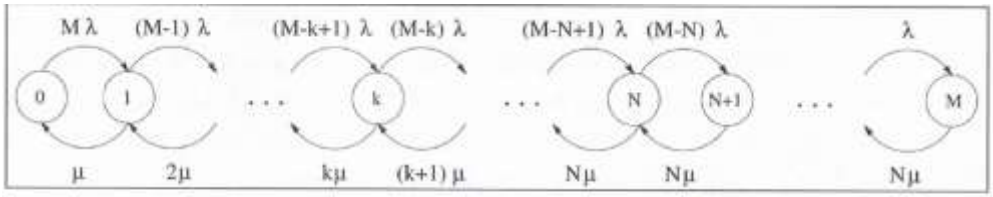
Markov model for the repair queue

- State *k* represents *k* machines have failed
- Part of the state transition diagram is showed below



The rate of failure for one machine is λ. In State 0, there are M working machine, the failure rate is Mλ.

The same argument holds for other state transition probability.



Note: There are (M+1) states.

Why is it Nμ?

Why not (N+1)μ?

- We can solve for P(0), P(1), …, P(M)

$$P(k) = \begin{cases} P(0)(\frac{\lambda}{\mu})^k C_k^m & k = 1, ..., N \\ P(0)(\frac{\lambda}{\mu})^k C_k^m \frac{N^{N-k}k!}{N!} & k = N+1, ...M \end{cases}$$

Where

$$P(0) = \left[\sum_{k=0}^{N} (\frac{\lambda}{\mu})^k C_k^m + \sum_{k=N+1}^{M} (\frac{\lambda}{\mu})^k C_k^m \frac{N^{N-k}k!}{N!} \right]^{-1}$$

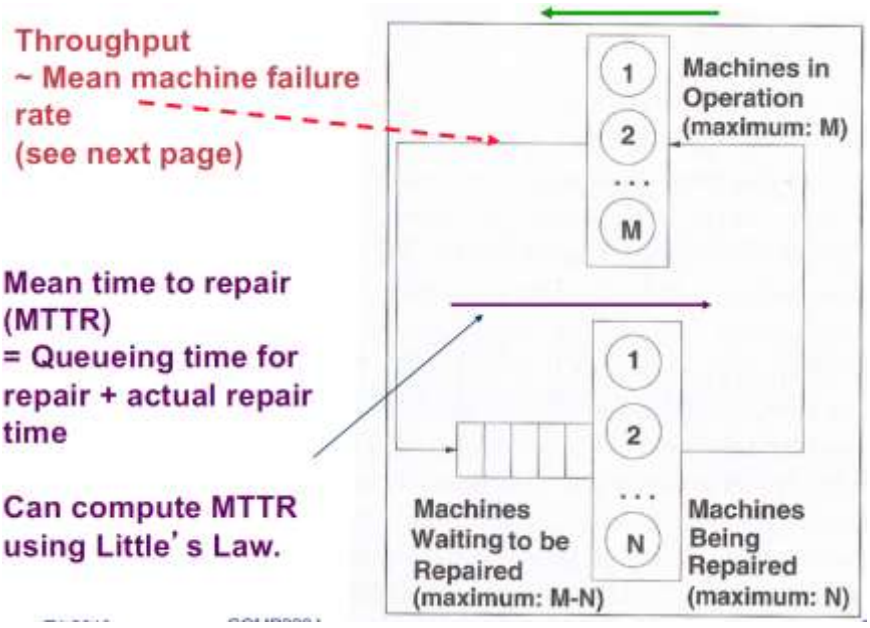
Using the model

Probability that exactly *k* machines are available = P(M-k)
Probability that at least *k* machines are available
= P(0) + P(1) … + P(M-k)
But expression for P(*k*)'s are complicated, need numerical software

Example:

- M = 120
- Mean-time-to-failure = 500 minutes
- Mean service time to repair = 20 minutes
- N = 2, 5 or 10
- The results are showed in the graphs in the next 2 pages
  - I used the file "data_centre.m" to do the computation, the file is available on the course web site.

Think time ~ Mean-time-to-failure (MTTF) = 1 / λ



Mean machine failure rate

State	Probability	Failure rate
0	P(0)	$M\lambda$
1	P(1)	$(M-1)\lambda$
2	P(2)	$(M-2)\lambda$
…	…	
k	P(k)	$(M-k)\lambda$
…	…	
M	P(M)	0

$$\bar{X}_f = \sum_{k=0}^{M-1} (M-k)\lambda P(k)$$

Continuous-time Markov chain

Useful for analysing queues when the inter-arrival or service time distribution is exponential

The procedure is fairly standard for obtaining the steady state probability distribution

- Identify the state
- Find the state transition rates
- Set up the balance equations
- Solve the steady state probability

We can use the steady state probability to obtain other performance metrics: throughput, response time etc.

- May need Little's Law etc.

Continuous-time Markov chain is only applicable when the underlying probability distribution is exponential but the operations laws (e.g. Little's Law) are applicable no matter what the underlying probability distributions are.

Markov chain

Markov chain is big field in itself. We have touched on only continuous-time Markov chain

- There are also discrete time Markov chains
- Markov chain has discrete state, a related concept is Markov process whose states are continuous

Markov chain / processes have many applications

- Page rank algorithm from Google can be explained in terms of discrete-time Markov chain
- Graphical Models (from machine learning)
- Transport engineering
- Mathematical finance

Personally, I use Markov chains to understand how living cells process information

Week 3A: Revision problems

Question 1

You have a computer system with a single CPU.

- Both inter-arrival and service times are exponentially distributed.
- The request only requires services at the CPU.
- Each request only visits the CPU once.
- A finished request will leave the system.
- Mean arrival rate is 9 request/s
- Mean service time required by a request at the CPU is 0.1s.

What is the utilisation of the CPU?

What is the mean response time?

The utilisation is pretty high and you want to change the system. You can think of 3 alternatives.

The assumptions imply that the queue is an M/M/1 queue with arrival rate $\lambda = 9$ and mean service time $(= 1/\mu) = 0.1$. This gives an utilisation $\rho = \lambda / \mu = 9 * 0.1 = 0.9$. Plugging these numbers into the mean response time T for M/M/1 = $1/(\mu (1- \rho))$, we have T = 1s.

Question 1 - Alternative 1

Replace the existing CPU by one that is 2 times faster

You may assume that the service time is inversely proportional to CPU speed.

We upgrade the CPU to one twice as fast and since the service time is inversely proportional to the speed, the new mean service time = 0.05. The system is an M/M/1 queue with $\lambda = 9$ and $1/\mu = 0.05$. The utilisation $\rho = \lambda / \mu = 9 * 0.05 = 0.45$. Using the M/M/1 mean response time formula gives T = 0.0909s.

Question 1 - Alternative 2

Buy a system which is identical to the current one

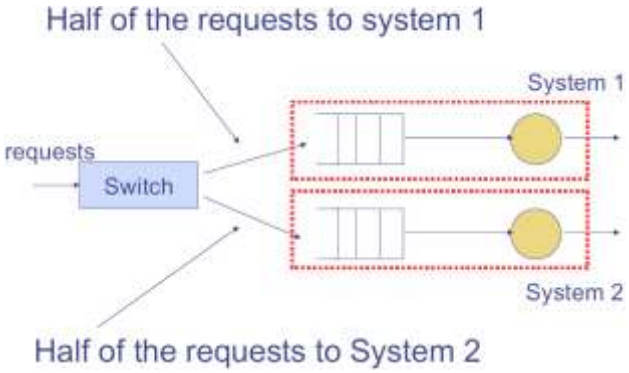
Put the two systems in parallel

Add a switch in front of the system

When a request arrives, the switch will randomly assign the request to one of the systems. On average, half of the request goes to each system

(Pictorial representation on the next slide)

Assume the switch requires negligible time



Question 1 - Alternative 3

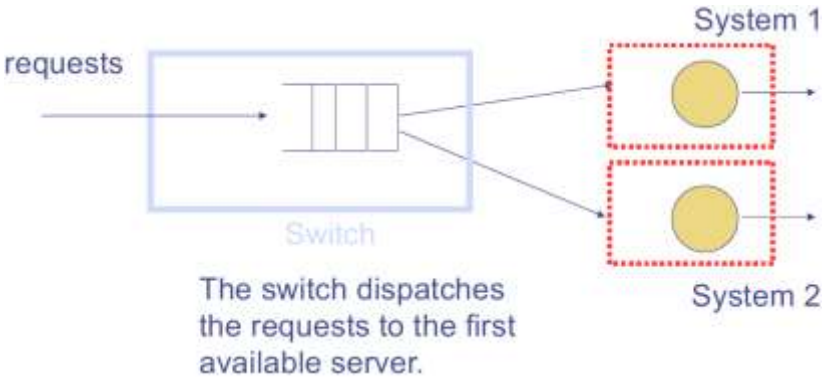
Similar to Alternative 2, we buy a system which is identical to the current one and we also buy a switch

However, we only maintain a queue at the switch

If both systems are busy, the request waits at the switch; otherwise, the switch dispatches the request to any of the available systems

Assuming that it takes negligible time for the switch to find out whether a system is idle

(Pictorial representation on the next page)



This alternative is effectively an M/M/2 queue with $\lambda = 9$ and $1/\mu = 0.1$. The utilisation $\rho = \lambda / \mu \ / 2 = 9 * 0.1 / 2 = 0.45$. Using the M/M/2 mean response time formula gives = 0.1254s.

Question 1

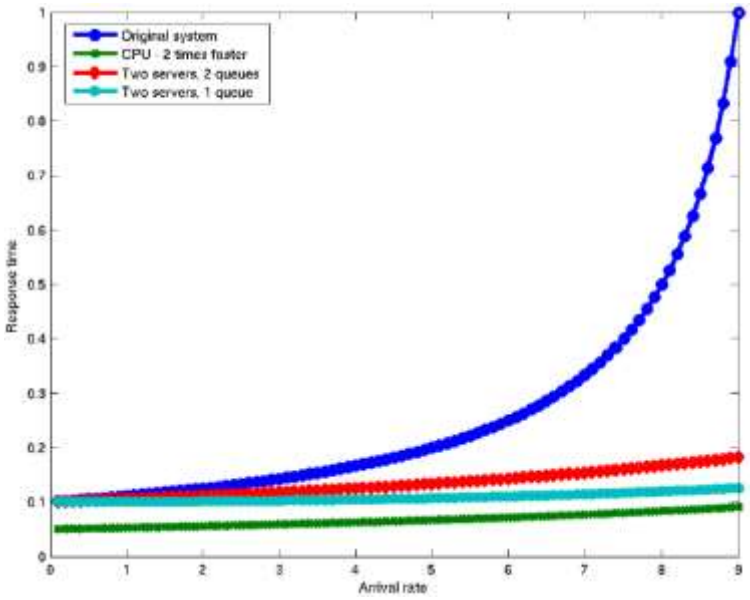
Part (a): Calculate the resulting mean response time for each for the three alternatives

Part (b): Repeat part (a) for a number of different mean arrival rates. Plot a graph of arrival rates against the mean response time.

Part (c): What observations can you make from these calculations?

Part (d): What is the best way to upgrade the system in terms of performance? However, the best way to upgrade in terms of performance may not be the best way to upgrade in terms of cost, why?

We use a number of different arrival rates and we plot a curve of how the response time changes with the arrival rate. See the graph below:



Observations:

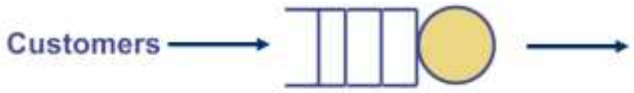
Note all the three alternatives have the same utilisation but have different response times. All three alternatives improve the response time of the current system but to different extend.

Alternative 1 gives the best response time but it means throwing away the existing hardware and buy an entirely new system. This can be costly.

Both Alternatives 2 and 3 make use of the existing hardware but Alternative 3 gives a better response time. It may be cheaper to scale up the system by using Alternatives 2 and 3 in the long term because it allows incremental expansion of the system.

Note: I used Matlab to create the above graph. I've posted the files on the course web site.

Question 2



Consider a single server queue as shown above

Part (a): Consider the situation

- The inter-arrival time is a constant and is given by 1 second.
- The service time required by each customer is always 0.5 second.

- What is the mean waiting time per customer?

Part (b): Consider the situation

- The inter-arrival time is exponentially distributed with mean 1 second
- The service time required by each customer is exponentially distributed with mean 0.5s

- What is the mean waiting time per customer?

Compare the answers of Parts (a) and (b). What conclusions can you draw?

Part (a)

There is no queueing at all in this case. The mean waiting time is zero second.

Part (b)

The mean arrival rate is 1 customer / s. The mean service rate is 2 customers/s. By the M/M/1 formula, we know the mean response time is $1/(\mu(1-\rho)) = 1$s. The mean waiting time = mean response time – mean service time = 1 – 0.5 = 0.5s.

You may draw a number of conclusions here:

- Queueing is a result of the randomness in arrival and service patterns. In Part (b), the mean arrival rate is 1 customer/s, which means that the instantaneous arrival rate can sometimes be greater than 1 customer/s, the high instantaneous arrival rate can create a backlog in the system and create queueing.
- Waiting time is not just a function of the mean service time and mean arrival rate. It depends on the arrival and service time distributions too.

Question 3

An Internet Service Provider has 4 dial-up ports. Connection requests obey Poisson distribution with a mean arrival rate of 3 requests per hour. The session duration of each connection request is exponentially distributed with a mean of 1.5 hours. What is the probability that a connection request will be rejected?

This is effectively a M/M/4/4 system with arrival rate $\lambda = 3$ and mean service rate $\mu = 1/1.5$. The probability that a connection request is blocked is the same as the probability that there are 4 requests in the system. This is given by the formula

$$P_m = \frac{\frac{\rho^m}{m!}}{\sum_{k=0}^{m} \frac{\rho^k}{k!}} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

With m = 4.

Week 3B: Revision problems

Question 1

Consider a hypothetical call centre with 1 receptionist and 2 technical staff. Customers make calls to this call centre to receive technical support. Calls arrive at this call centre with a mean inter-arrival time of 10 minutes, exponentially distributed.

An incoming call is first directed to the receptionist. If a call arrives at the call centre when the receptionist is idle, the call will be answered; otherwise the call is dropped.

After a call has been processed by the receptionist, it will be sent to a technical staff for further processing. The rules are:

- If both technical staff are busy, the call is dropped
- If a technical staff is available, the call will be directed to this staff.
- If both technical staff are available, the receptionist picks one of the staff with equal probability.

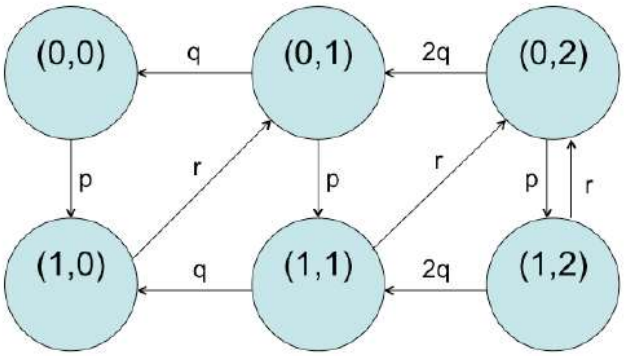
A customer will only be satisfied if their call is processed by both the receptionist and a technical staff, otherwise the customer is unsatisfied.

Assuming that the mean processing time required by the receptionist is 3 minutes, and that of each technical staff is 15 minutes; all distributions are exponentially distributed.

Answer the following questions:

- Draw the appropriate Markov model. Label all arcs.
- In steady state, what are the probabilities of being in each state?
- Find the probability that a technical staff is busy
- What is the probability that an arriving call is dropped by the receptionist?
- What is the "good" throughput of the call centre (i.e. the rate of which satisfied customers leave the call centre) ?

The states are 2-tuple *(x,y)* where x is the number of calls at the receptionist and y is the number of calls at the technical staff. We have $x = 0$ or 1; and, $y = 0, 1, 2$. There are altogether 6 states (0,0), (0,1), (0,2), (1,0), (1,1) and (1,2). The state space diagram is as follows:



Where p = 1/10, q = 1/15, r = 1/3

Note that in the state transition diagram, the transition rate from (0,2) to (0,1) should be 2q. The reason is that each call departs with a rate q and if any one of them has finished, then the state will move from (0,2) to (0,1). Based on the above state space diagram, the balance equations are

P(0,0) p = P(0,1) q
P(0,1) (p+q) = P(1,0) r + P(0,2) 2q
P(0,2) (p+2q) = P(1,1) r + P(1,2) r
P(1,0) r = P(0,0) p + P(1,1) q
P(1,1) (q+r) = P(0,1) p + P(1,2) 2q
P(1,2) (2q+r) = P(0,2) p

Where P(0,0) = Probability in State (0,0) etc.

You can choose any five of these equations together with the conservation of probability relation

P(0,0) + P(0,1) + P(0,2) + P(1,0) + P(1,1) + P(1,2) = 1

to solve for the probabilities.

To find the probability that a technical staff is busy: This is the probability in the states (0,1), (0,2), (1,1) and (1,2) where there is at least one call being answered by a technical staff. The required probability is the sum P(0,1)+P(0,2)+P(1,1)+P(1,2)

To find the probability that an arriving call is dropped by the receptionist: This is the probability that the receptionist is busy. The receptionist is busy in the states (1,0), (1,1) and (1,2). The required probability is the sum P(1,0) + P(1,1) + P(1,2)

Finally, we are asked to calculate the "good" throughput of the call centre (i.e. the rate at which satisfied customers leave the call centre, having their calls answered by both the receptionist and the technical staff). This is effectively the total departure rate of the calls answered by the technical staff. The answer is therefore

P(0,1) q + P(0,2) 2q + P(1,1) q + P(1,2) 2q

An alternative viewpoint is the rate at which the receptionist transfers the calls to the technical staff. The "good" throughput is also give by:

P(1,0) r + P(1,1) r

You can use the flow balance equations to prove that these two answers are the same.

Question 2

On pages 6-20 of the lecture notes, we discuss an example of analysing a database server with one CPU and 2 disks using a Markov chain. (Note: The material is based on Chapter 10 of Menasce et al "Performance Analysis".) In the example, we find that with the current workload and hardware specifications of the system, the response time is 0.7372 minutes per transaction. The engineer in charge of the system would like to improve the response time of the system by using a faster CPU. Assuming:

- The workload remains the same as before.
- There are always 2 users in the system.
- The service time for the disks remains as before.
- The service time for the CPU is inversely proportional to the speed of the CPU.

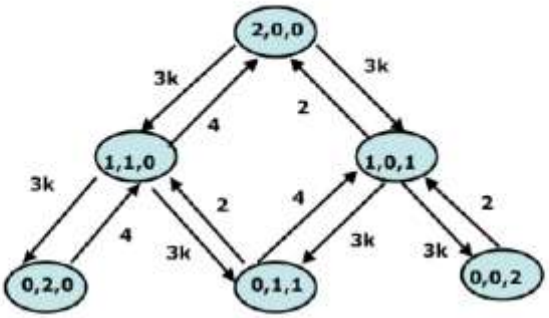
If the engineer would like to achieve a mean response time of 0.65 minutes per transactions, by how many times must the engineer speed up the CPU?

Is speeding up the CPU a good choice? Explain.

Hint: Assuming that you speed the CPU up by k times. You will need to build a Markov chain in terms of k. Once you have done it, you can then write a computer program whose input is the numerical value of k and whose output is the response time for the given speed up factor k. The computer program will need to solve the Markov chain. Once you have this program, you can plot a graph on how response time depends on k. You can then determine the value of k from this graph.

If we speed up the CPU by k times, then the mean processing rate of the CPU will be $ók$ transactions per minute. Since the workload remains the same, we can use the same Markov chain that was used in the lecture except that we need to multiply the transition rates from the CPU by k times.

Thus, if we speed up the CPU by k times, the Markov chain should be:



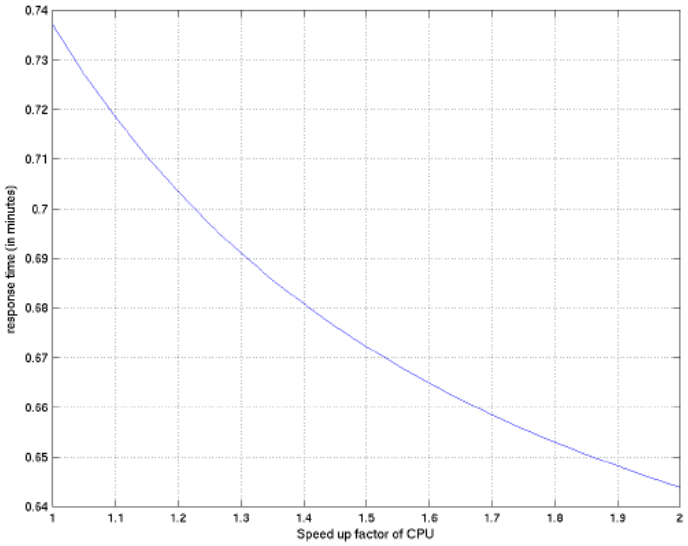
For any given value of k, you can solve this Markov chain to obtain the response time via the following steps:

- Form the balance equations
- Solve the balance equations together with conservation of probability relation to obtain the steady state probabilities.
- Compute the throughput from the steady state probabilities
- Use Little's Law to obtain the mean response time.

(This is exactly the same procedure that we used in the lecture)

What this means is that for every value of k, you will be able to find the response time. You can write a computer program which for a given value of k, determine what the response time is for that value of k. An example of such a program in Matlab is "cpu_speedup.m".

By using the results from this program, you can plot a graph to see how the speed up factor k influences the response time. (You can modify the program cpu_speedup.m slightly so that it will calculate the response time for a number of values of k and plot the graph, see the program cpu_speedup_graph.m)



From this graph, you can see that if you want the response time to reduce to 0.65 minutes per transaction, you need to speed up the CPU by 1.86 times.

Speeding up the CPU may not be the best idea, because the bottleneck of the system is the slow disk. A more effective solution is to upgrade the slow disk.

Week 4A: Non-Markovian queues

General single-server queues

Need to specify the

- Inter-arrival time probability distribution
- Service time probability distribution

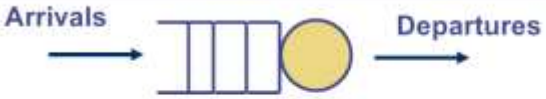
Independence assumptions

- All inter-arrival times are independent
- All service times are independent
  - The amount of service of customer A needs is independent of the amount of time customer B needs
- The inter-arrival time and service time are independent of each other

Under the independence assumption, we can analyse a number of types of single server queues

- Without the independence assumption, queueing problems are very difficult to solve!

Classification of single-server queues



Recall Kendall's notation: "M/M/1" means

- "M" in the 1st place means inter-arrival time is exponentially distributed
- "M" in the 2nd place means service time probability is exponentially distributed
- "1" in 3rd position means 1 server

We use a "G" to denote a general probability distribution

- Meaning any probability distribution

Classification of single-server queues:

		Service time Distribution:	
		Exponential	General
Inter-arrival time distribution:	Exponential	M/M/1	M/G/1
	General	G/M/1	G/G/1

Example M/G/1 queue problem

Consider an e-mailer server

E-mails arrive at the mail server with a Poisson distribution with mean arrival rate of 1.2 messages/s

The service time distribution of the emails are:

- 30% of messages processed in 0.1 s, 50% in 0.3 s, 20% in 2 s

What is

- Average waiting time for a message?
- Average response time for a message?
- Average number of messages in the mail system?

This is an M/G/1 queue problem

- Arrival is Poisson
- Service time is not exponential

In order to solve an M/G/1 queue, we need to understand what the moment of a probability distribution is.

Moment of a probability distribution

First moment

- Consider a discrete probability distribution
  - There are *n* possible outcomes: $x_1, x_2, \ldots, x_n$
  - The probability that x_i occurs is p_i
- Example: For a fair die
  - The possible outcomes are 1,2,..., 6
  - The probability that each outcome occurs is 1/6

- The first moment (also known as the mean or expected value) is

$$E[X] = \sum_{i=1}^{n} x_i p_i$$

- For a fair die, the first moment is

$= 1 * 1/6 + 2 * 1/6 + \ldots + 6 * 1/6 = 3.5$

Second moment:

- The second moment of a discrete probability distribution is

$$E[X^2] = \sum_{i=1}^{n} x_i^2 p_i$$

- For a fair die, the second moment is

$= 1^2 * 1/6 + 2^2 * 1/6 + \ldots + 6^2 * 1/6$

- You can prove that
  - Second moment of X = $(E[X])^2$ + Variance of X

- Note: The above definitions are for discrete probability distribution. We will look at continuous probability distribution a moment later

Solution to M/G/1 queue:

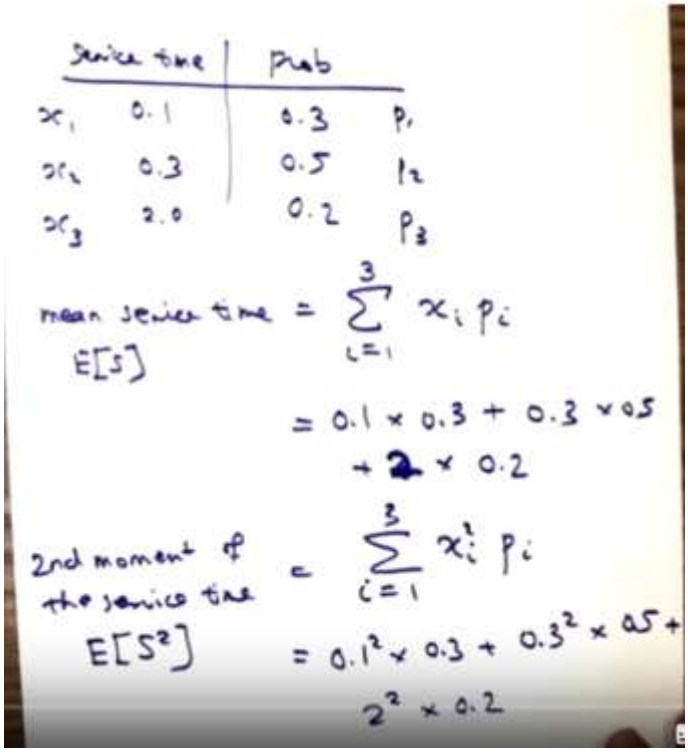
- M/G/1 analysis is still tractable
- M/G/1 is no longer a Markov chain
- For a M/G/1 queue with the characteristics
  - Arrival is Poisson with rate λ
  - Service time S has
    - Mean = $1/\mu$ = E[S] = First moment
    - Second moment = $E[S^2]$
- The mean waiting time W of a M/G/1 queue is given by the Pollaczek-Khinchin (P-K) formula:

$$W = \frac{\lambda E[S^2]}{2(1-\rho)} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

Queueing problem

- Consider an e-mailer server
- E-mails arrive at the mail server with a Poisson distribution with mean arrival rate of 1.2 messages/s
- The service time distribution of the emails are:
  - 30% of messages processed in 0.1 s, 50% in 0.3 s, 20% in 2 s

- *Exercise*: In order to find the mean waiting time using the P-K formula, we need to know
  - Mean arrival rate,
  - Mean service time, and,
  - Second moment of service time.
- Can you find them?



- Since
  - Mean arrival rate λ = 1.2 messages/s
  - Mean service time (E[S] or 1 / μ) = 0.58s
  - Second moment of mean service time $E[S^2]$ = 0.848 s^2
- Utilisation $\rho = \lambda / \mu = \lambda E[S]$ = 1.2 * 0.58 = 0.696
- Substituting these values in the P-K formula

$$W = \frac{\lambda E[S^2]}{2(1-\rho)}$$ W = 1.673s.

- How about:
  - Average response time for a message
  - Average number of messages in the mail system

Since the mean waiting time W = 1.673s.

The mean response time T is
T = W + E[S] = 1.673 + 0.58 = 2.253

By Little's Law,
Average # messages in the system
= Throughput x mean response time
= λ T = 1.2 * 2.253 = 2.704 messages

Exercise: Can you use mean waiting time and Little's Law to determine the mean number of messages in the queue?

Understanding the P-K formula

Since the Second moment of S = $E[S]^2$ + Variance of S
We can write the P-K formula as

- Meaning waiting time =

$$W = \frac{\lambda(E[S]^2 + \sigma_S^2)}{2(1-\rho)}$$

Smaller variance in service time ➔ smaller waiting time
M/D/1 is a special case of M/G/1

- "D" stands for deterministic: Constant service time E[S] and Variance of S = 0
- For the same value of ρ and E[S], deterministic has the smallest mean response time

$$\sigma^2 = \frac{\sum (X-\mu)^2}{N}$$

σ^2：为总体方差，X：为变量，μ：为总体均值，N：为总体例数

Moments for continuous probability density

- Exponential function is a continuous probability density
- If a random variable X has continuous probability density function f(x), then its
  - first moment (= mean, expected value) E[X] and
  - second moment $E[X^2]$

  are given by

$$E[X] = \int x f(x) dx$$

$$E[X^2] = \int x^2 f(x) dx$$

- If the service time S is exponential with rate μ, then
  - $E[S] = 1/\mu$
  - $E[S^2] = 2/\mu^2$

这种相当于是持续得分布而不是离散得，不能从简单几个已知得服务时间来计算平均服务时间，所以要使用积分去求解（因为指数分布本来就不是连续得）

由上可以得知，在连续得概率分布模型中与离散分布模型中是不一样得，然后我们就有（也就是说 **M/M/1 as a special case of M/G/1**）：

M/M/1 as a special case of M/G/1

Let us apply the result of the M/G/1 queue to exponential service time

- Let us put $E[S] = 1/\mu$ and $E[S^2] = 2/\mu^2$ in the P-K formula:

$$W = \frac{\lambda E[S^2]}{2(1-\rho)}$$

We get

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

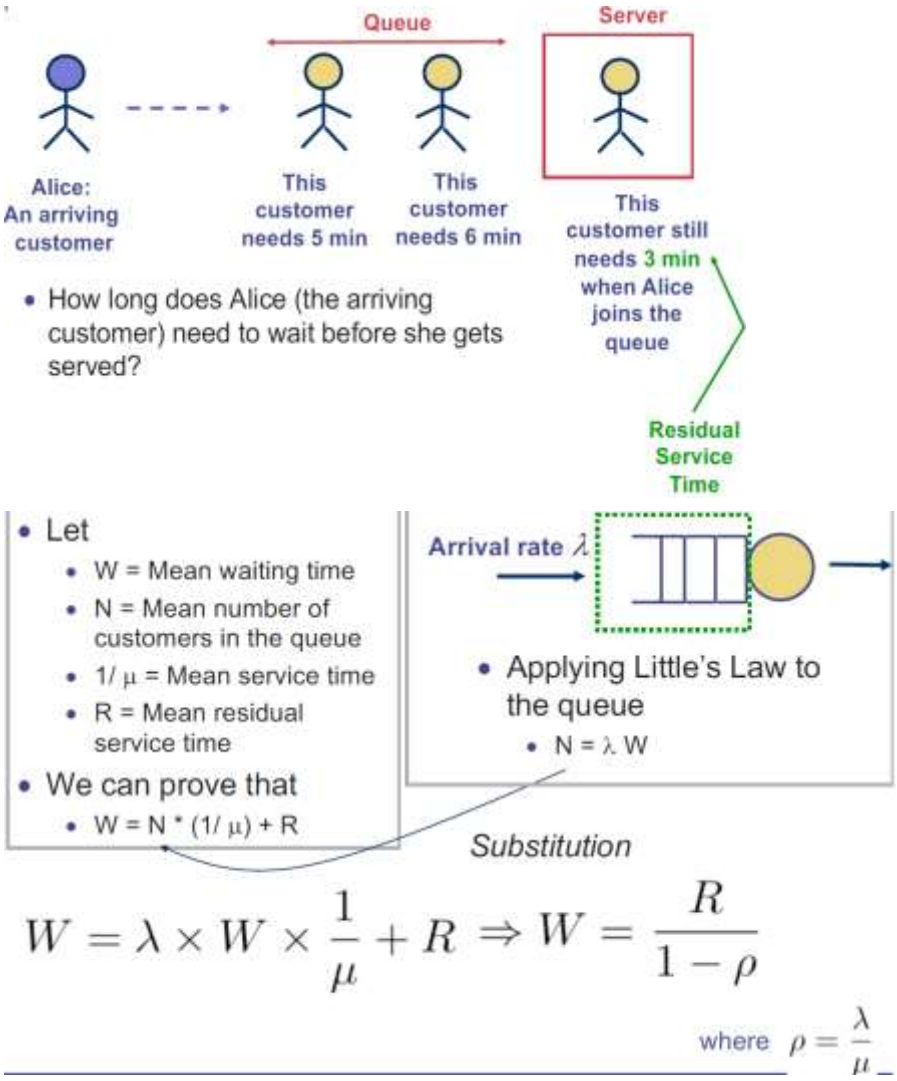
- Which is the same as the M/M/1 queue waiting time formula that we derive in Week 3A

也可以得到关于 waiting time：

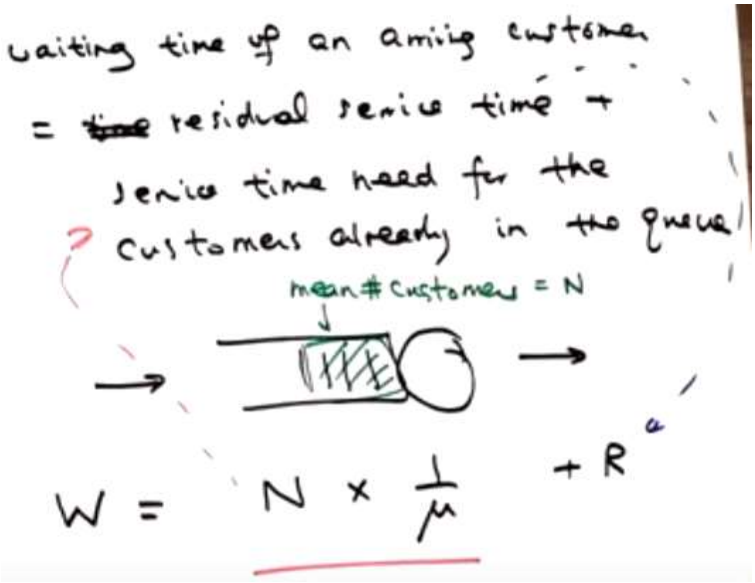
$$W = \frac{\lambda E[S^2]}{2(1-\rho)}$$

- $\rho \to 1, W \to \infty$

Deriving the P-K formula



上面那个可以被证明得等式，可以从下面得到，其实就是相当于:
平均等待时间=已经在服务器中任务还需要得平均时间（R = Mean residual）+ 在队列中得平均客户数目（N） * 平均每个客户得服务时间（1/μ）



由此，可以推的剩余服务时间得表达式：

- We have just showed that the mean waiting time in a M/G/1 queue is

$$W = \frac{R}{1-\rho}$$

- The P-K formula says

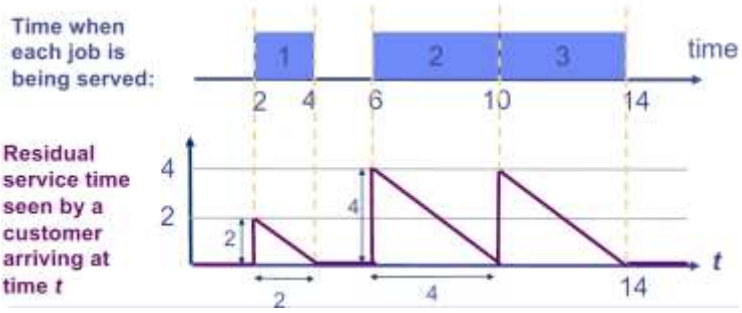
$$W = \frac{\lambda E[S^2]}{2(1-\rho)}$$

- We can prove the P-K formula if we can show that the mean residual time R is

$$R = \frac{1}{2}\lambda E[S^2]$$

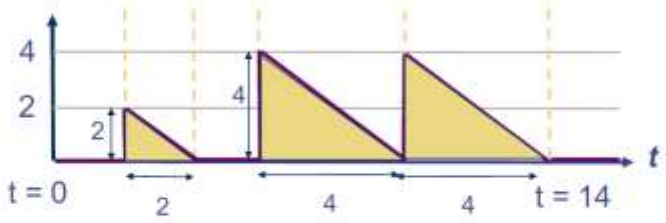
Prove the residual service time

Job index	Arrival time	Processing time required
1	2	2
2	6	4
3	8	4



Mean residual time

Residual service time seen by a customer arriving at time *t*



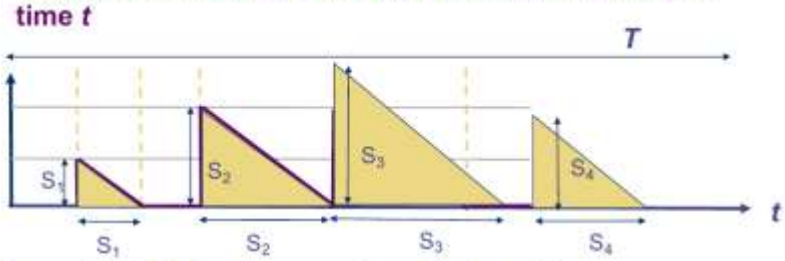
Mean residual time seen by an arriving customer over time [0,14]

$$= \frac{\text{Area under the curve over } [0,14]}{14}$$

Service time!

$$= \frac{\frac{1}{2} \times 2^2 + \frac{1}{2} \times 4^2 + \frac{1}{2} \times 4^2}{14}$$

Residual service time seen by a customer arriving at time *t*



Assuming M jobs are completed in time T
Mean residual time

$$= \frac{\sum_{i=1}^{M} \frac{1}{2} S_i^2}{T} = \frac{1}{2} \frac{\sum_{i=1}^{M} S_i^2}{M} \frac{M}{T} = \frac{1}{2} E[S^2] \lambda$$

The P-K formula:

- Thus, the mean residual time R is

$$R = \frac{1}{2} \lambda E[S^2]$$

- By substituting this into $W = \frac{R}{1-\rho}$
- We get the P-K formula

This derivation also shows that the waiting time is proportional to the residual service time

The residual service time is proportional to the 2nd moment of service time

G/G/1 queue

G/G/1 queue are harder to analyse

Generally, we cannot find an explicit formula for the the waiting time or response time for a G/G/1 queue

Results on G/G/1 queue include

- Approximation results
- Bounds on waiting time

Approximate G/G/1 waiting time

- There are many different methods to find the approximate waiting time for a G/G/1 queue
- Most of the approximation works well when the traffic is heavy, i.e. when the utilisation ρ is high
- Let
  - Mean arrival rate = λ
  - Variance of inter-arrival time = σ_a^2
  - Service time S has mean $1/\mu = E[S]$
  - Variance of service time = σ_s^2
- The approximate waiting time for a G/G/1 queue is

$$W \approx \frac{\lambda^2(\sigma_a^2 + \sigma_s^2)}{1 + \lambda^2\sigma_s^2} \frac{\lambda(E[S]^2 + \sigma_s^2)}{2(1-\rho)} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}$$

- Note: $\rho \to 1$, $W \to \infty$
- Large variance means large waiting time

Bounds for G/G/1 waiting time

Let

- Mean arrival rate = λ
- Variance of inter-arrival time = σ_a^2
- Service time S has mean $1/\mu = E[S]$
- Variance of service time = σ_s^2

A bound for the waiting time for a G/G/1 queue is

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\rho)}$$

Note that the bound suggests that large variance means large waiting time

Approximation for G/G/m queue

- Only approximate waiting time available for G/G/m
- The waiting time is

$$W_{G/G/m} = W_{M/M/m} \frac{C_a^2 + C_s^2}{2}$$

where $W_{M/M/m} = \text{Waiting time of M/M/m queue}$

$C_a = \text{Coeff of variation of inter-arrival time}$

$C_b = \text{Coeff of variation of service time}$

- Coefficient of variation of a random variable *X*

= Standard deviation of *X* / mean of *X*

Note: Variance in arrival or service time increases queueing

Coefficient = 系数，标准差公式如下:

$$\sigma(r) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - r)^2}$$

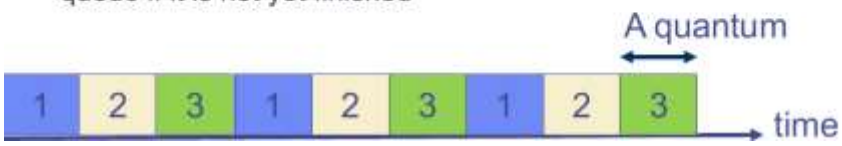
Processor sharing (PS)

We have so far assumed that the processor performs work on a first-come-first-serve basis

However, this is not how CPUs perform tasks

Consider an example: a CPU has a job queue with three tasks called Tasks 1, 2 and 3 in it

- CPU works on Task 1 for a certain amount of time (called a quantum) and then returns the task to the job queue if it is not yet finished
- CPU works on Task 2 for a quantum and returns the task to the job queue if it is not yet finished
- CPU works on Task 3 for a quantum and returns the task to the job queue if it is not yet finished



Modelling processor sharing

We assume the context switching time is negligible

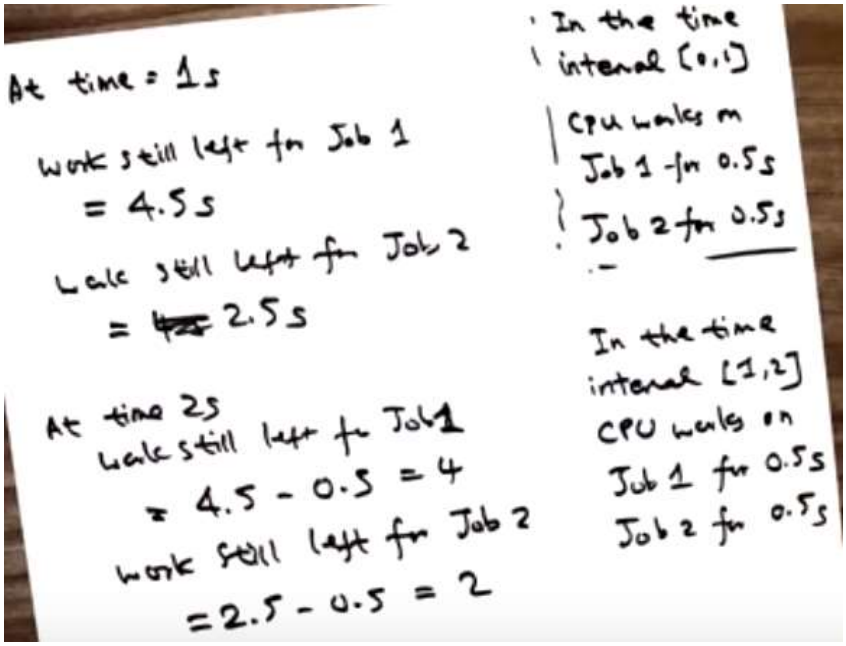
We assume the quantum is small compared with the length of the task, we can think about continuous processing instead of discrete processing

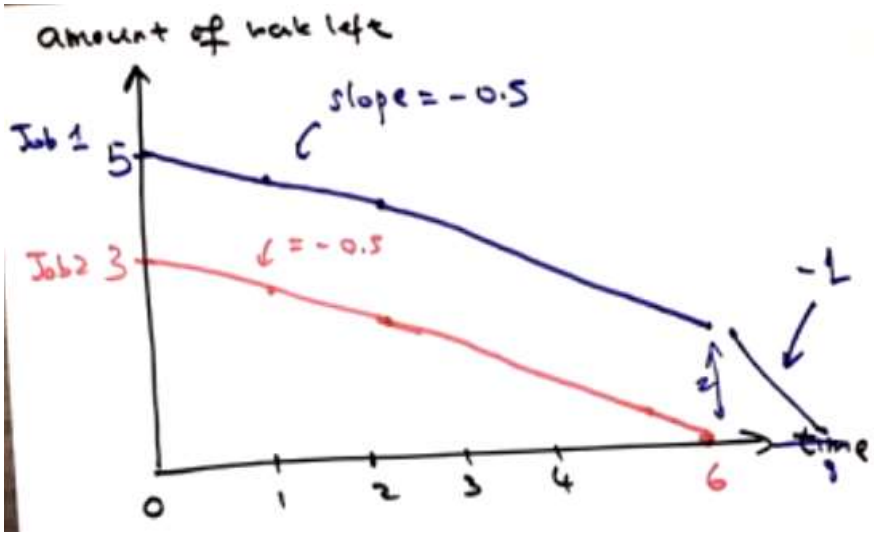
In a duration of time when there are n jobs in the job queue, each job receives 1/n of the service

PS: Example 1

- Example 1:
  - At time 0, there are 2 jobs in the job queue
  - Job 1 still needs 5 seconds of service
  - Job 2 still needs 3 seconds of service

  - Assuming no more jobs will arrive, determine the time at which the jobs will be completed





Example 2:

- At time 0, there are 2 jobs in the job queue
  - Job 1 still needs 5 seconds of service
  - Job 2 still needs 3 seconds of service
- Job 3 arrives at time = 1 second and requires 4 seconds of service
- Job 4 arrives at time = 2 second and requires 1 second of service
- No more jobs will arrive after Job 4

- Questions:
  - Without computing the finished times for Jobs 1 and 3, are you able to tell which of these two jobs will finish first?
  - Determine the time at which the jobs will be completed

Job 3 会比 Job 1 先完成， 因为 Job 1 剩下得工作要比 Job 3 多

At time 1s

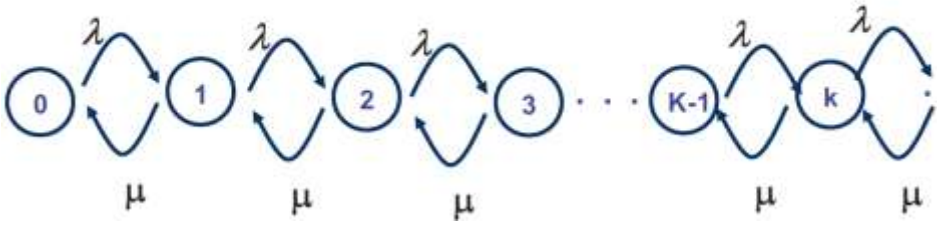
work left for Job 1 = 9 4.5s

—11— Job 3 = 4 s

M/M/1/PS queues

- Jobs arrive according to Poisson distribution
- Exponential service time
- One processor using processor sharing

- State *n* = there are *n* jobs in the job queue
- State diagram: same as M/M/1 queue and there is a reason for that



Summary

- We have studied a few types of non-Markovian queues
  - M/G/1, G/G/1, G/G/m
- Key method to derive the M/G/1 waiting time is via the *residual service time*
- Processor sharing (PS)

Week 4A: Revision problems

1.顾客根据泊松过程到达杂货店的收银台，每分钟收费 1 次。每个客户携带的物品数量均匀分布在 1 到 40 之间。这家商店有两个收银台，每个收银台能以每分钟 15 个的速度处理商品。为减少顾客排队等候时间，店长考虑将两个柜台中的一个专供 X 件以下顾客使用，另一个专供 X 件以上顾客使用。编写一个小的计算机程序，找出 x 的值，从而将平均客户等待时间最小化。

First note that one counter is not sufficient to serve all the customers. If we consider all the customers together, each customer carries on average 20.5 items, which takes $\frac{20.5}{15}$ to complete. Since the customer arrival rate is 1, the utilisation will be above 1 if only one counter is used.

Let us refer to the two counters as Counter 1 and Counter 2. Let us assume that Counter 1 serves customers with x items or less and Counter 2 serves customers with more than x items, where $1 \leq x \leq 39$.

Let $\lambda(=1)$ denote the overall arrival rate and $\mu = 15$ be the service rate of each counter.

Let us consider Counter 1 first. Since only customers with x items or less go to Counter 1, the arrival rate at Counter 1 is $\lambda\frac{x}{40}$. The customers arriving at Counter 1 bring with them 1, 2, ..., x items uniformly distributed. Let S_1 denote the service time at Counter 1. We have

$$E[S_1] = \sum_{i=1}^{x} \frac{i}{\mu}\frac{1}{x} \tag{1}$$

$$E[S_1^2] = \sum_{i=1}^{x} \left(\frac{i}{\mu}\right)^2 \frac{1}{x} \tag{2}$$

Let $\rho_1 = \lambda\frac{x}{40}E[S_1]$, by the P-K formula, the mean waiting time at Counter 1 is

$$W_1 = \begin{cases} \lambda\frac{x}{40}\frac{E[S_1^2]}{2(1-\rho_1)} & \text{if } \rho_1 < 1 \\ \infty & \text{if } \rho_1 \geq 1 \end{cases} \tag{3}$$

Similarly, the arrival rate to Counter 2 is $\lambda_2 = \lambda\frac{40-x}{40}$. Let S_2 denote the service time at Counter 2, then

$$E[S_2] = \sum_{i=x+1}^{40} \frac{i}{\mu}\frac{1}{(40-x)} \tag{4}$$

$$E[S_2^2] = \sum_{i=x+1}^{40} \left(\frac{i}{\mu}\right)^2 \frac{1}{(40-x)} \tag{5}$$

Let $\rho_2 = \lambda\frac{40-x}{40}E[S_2]$, by the P-K formula, the mean waiting time at Counter 2 is

$$W_2 = \begin{cases} \lambda\frac{40-x}{40}\frac{E[S_2^2]}{2(1-\rho_2)} & \text{if } \rho_2 < 1 \\ \infty & \text{if } \rho_2 \geq 1 \end{cases} \tag{6}$$

The mean waiting time of the customers is

$$W = \frac{x}{40}W_1 + \frac{40-x}{40}W_2 \tag{7}$$

Note that W is a function of x. We write a computer program (Matlab file *week04A_q1.m*) to calculate how W varies with x. Figure 1 shows how W varies with x. It can be seem that the minimum value of W is achieved at $x = 28$.

2.能够以 50 kbits/s 的速率传输的通信线路将用于容纳 10 个会话，每个会话以 150 包/分钟的速率生成泊松流量。分组长度分布如下：10%的分组长度为 100 位，其余为 1500 位。找出平均队列长度和数据包在开始在通信线上传输之前必须等待的时间。您可以假定有足够的

缓冲区空间来存储排队的数据包。
The system behaves as an M/G/1 queueing system.

Since there are 10 sessions each generating Poisson traffic at a rate of 150 packets/minute, the packet arrival rate to the communication line is 1500 packets/minute or 25 packets/s ($= \lambda$).

With a transmission rate of 50 kbits/s, a 100-bit packet requires a transmission time (= service time in queueing theory terminology) 0.002s and a 1500-bit packet requires a transmission time of 0.03s. (Recall that transmission time is packet size divided by transmission rate.)

Given that 10% of the packets are 100 bits long and the rest are 1500 bits long, the mean service time $E[S]$ (where S denotes the service time random variable)

$$E[S] = 0.1 * 0.002 + 0.9 * 0.03 = 0.0272s \tag{8}$$

and the second moment of the service time is

$$E[S^2] = 0.1 * 0.002^2 + 0.9 * 0.03^2 = 8.1040 \times 10^{-4} s^2 \tag{9}$$

The mean waiting time W, according to the P-K formula, which applies to M/G/1 queueing system, is

$$W = \frac{\lambda E[S^2]}{2(1 - \lambda E[S])} = 31.7ms \tag{10}$$

By Little's Law, the mean queue length is given by the product of the throughout of the queue and the mean waiting time,

$$\lambda * W = 0.7914 \ \ packets \tag{11}$$

Week 4B: Queueing disciplines and Generating random numbers

Queuing disciplines

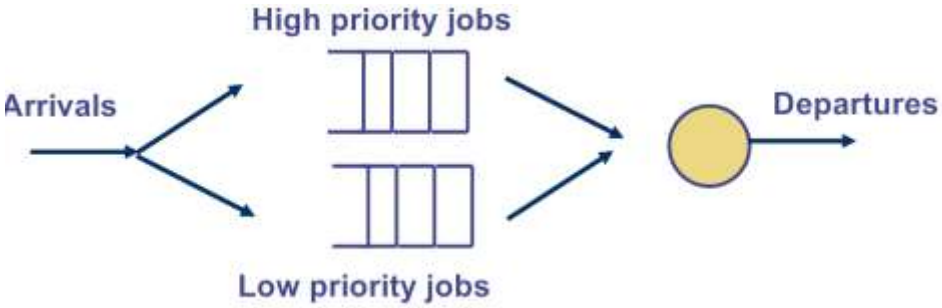
We have focused on *first-come first-serve* (FCFS) queues so far

However, sometimes you may want to give some jobs a higher priority than others

Priority queues can be classified as

- Non-preemptive
- Preemptive resume

Priority queueing



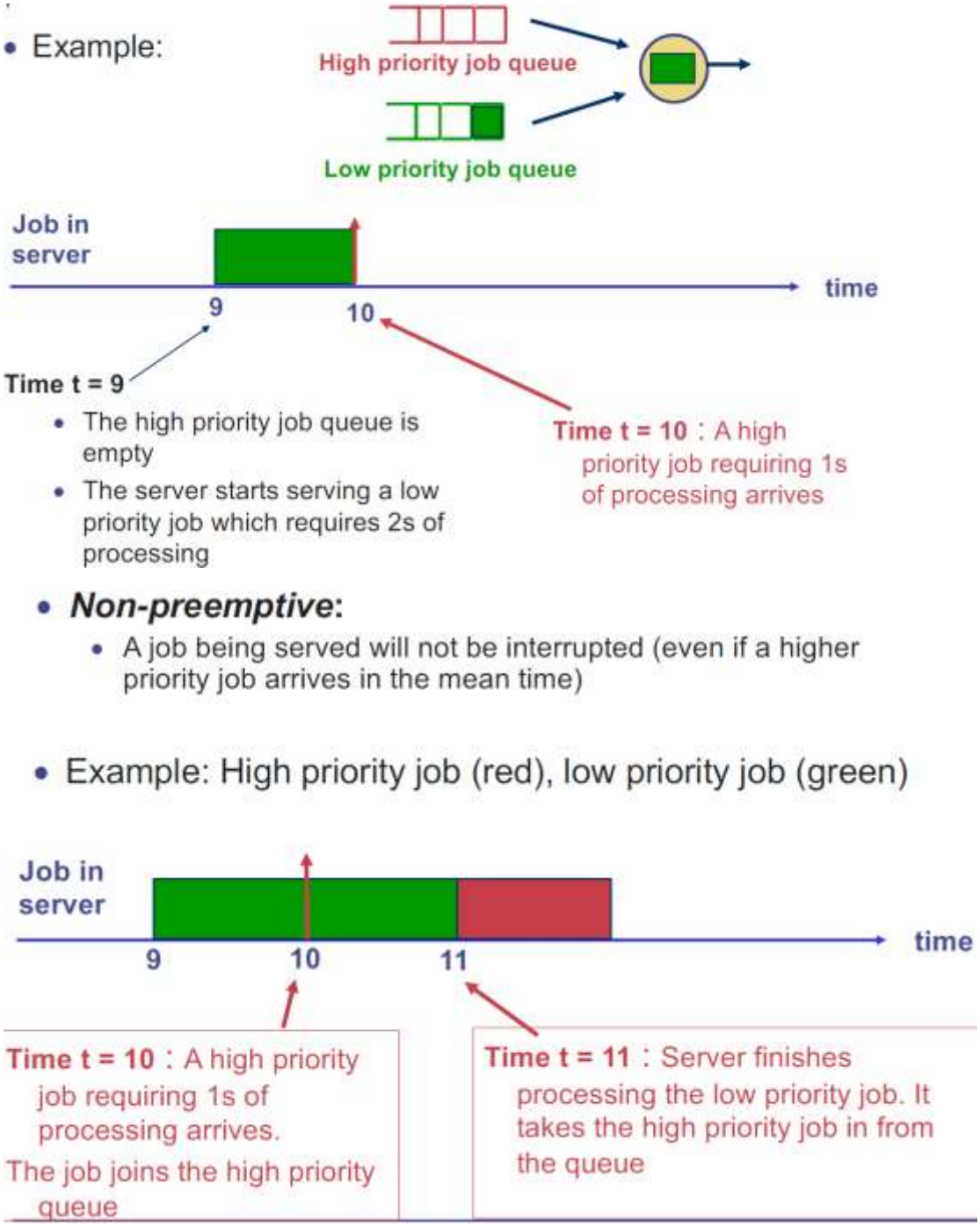
A job with low priority will only get served if the high priority queue is empty

Each priority queue is a FCFS queue

Exercise: If the server has finished a job and finds 1 job in the high priority queue and 3 jobs in the low priority queue, which job will the server start to work on?

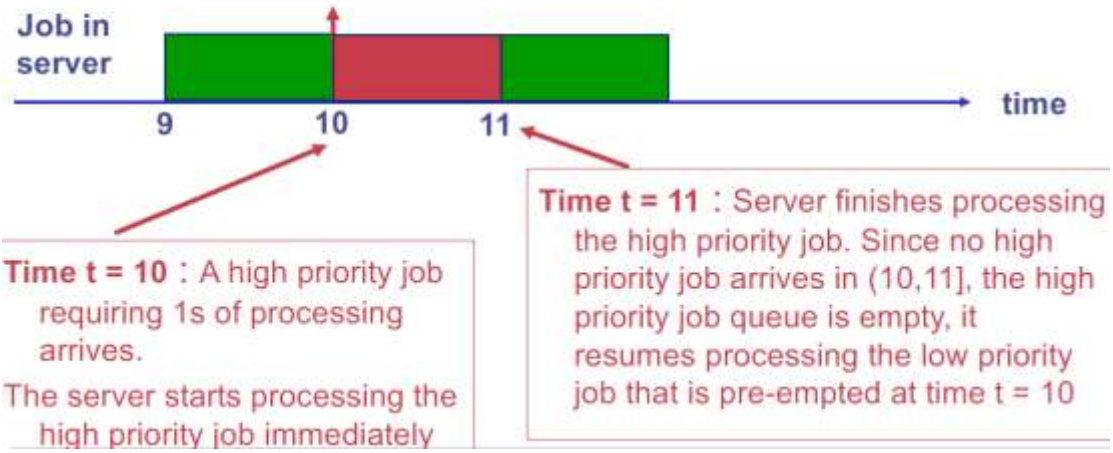
- Repeat the exercise when the high priority queue is empty and there are 3 jobs in the low priority queue.

Preemptive and non-preemptive priority

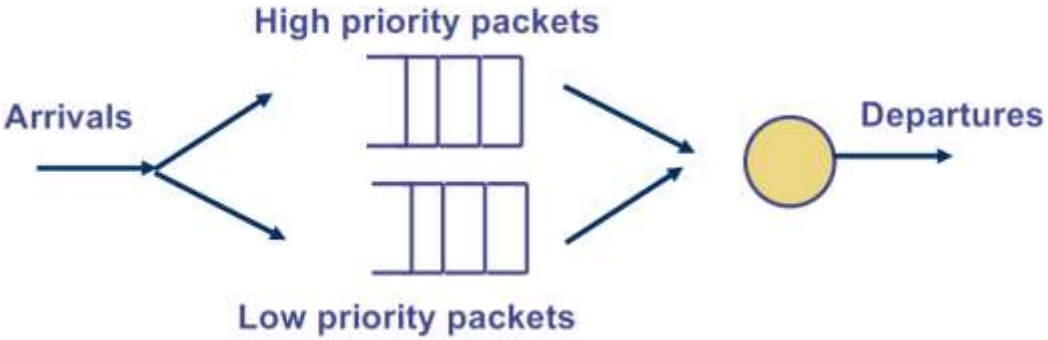


- ***Preemptive resume*:**
  - Higher priority job will interrupt a lower priority job under service. Once all higher priorities served, an interrupted lower priority job is resumed.

- Example: High priority job (red), low priority job (green)



Example of non-preemptive priority queueing



Example: In the output port of a router, you want to give some packets a higher priority

- In Differentiated Service
  - Real-time voice and video packets are given higher priority because they need a lower end-to-end delay
  - Other packets are given lower priority

You cannot preempt a packet transmission and resume its transmission later

- A truncated packet will have a wrong checksum and packet length etc.

Example of preemptive resume priority queueing

E.g. Modelling multi-tasking of processors

Can interrupt a job but you need to do context switching (i.e. save the registers for the current job so that it can be resumed later)

M/G/1 with priorities

Separate queue for each priority (see picture next page)

- Classified into P priorities before entering a queue
- Priorities numbered 1 to P, Queue 1 being the highest priority

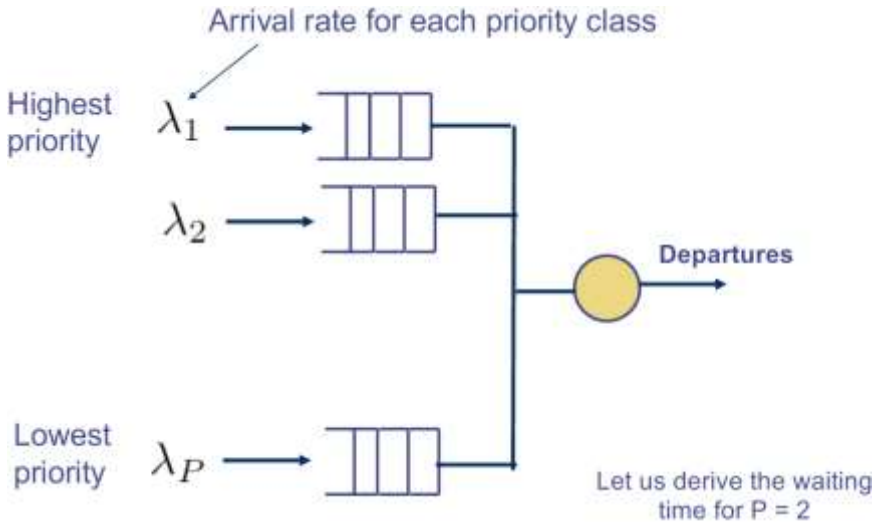
Arrival rate of priority class p is

$$\lambda_p \text{ where } p = 1,\cdots P$$

Average service time and second moment of class p requests is given by

$$E[S_p] \text{ and } E[S_p^2]$$

Priority queue

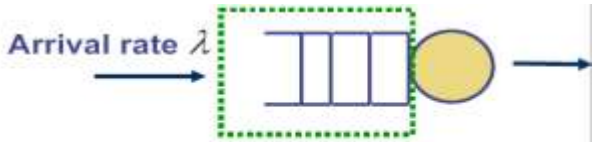


Let

- W = Mean waiting time
- N = Mean number of customers in the queue
- 1/ μ = Mean service time
- R = Mean residual service time

We can prove that

- W = N * (1/ μ) + R



- Applying Little's Law to the queue
  - N = λ W

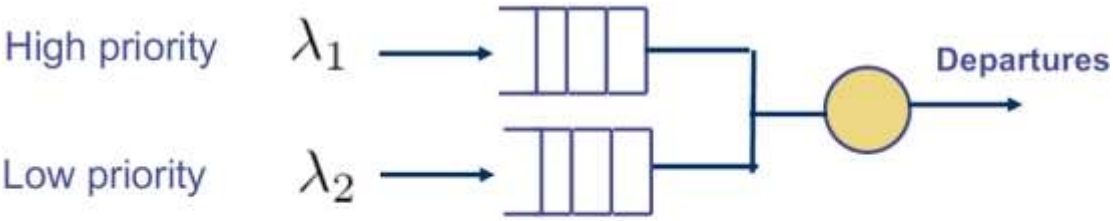
Substitution

$$W = \lambda \times W \times \frac{1}{\mu} + R$$

Mean residual time R

$$R = \frac{1}{2}\lambda E[S^2]$$

Deriving the non-preemptive queue result



- S_1 - service time for Class 1 with mean $E[S_1]$
- W_1 = mean waiting time for Class 1 customers
- N_1 = number of Class 1 customers in the queue
- R = mean residual service time when a customer arrives
- We have for Class 1: $W_1 = N_1 E[S_1] + R$
- Little's Law: $N_1 = \lambda_1 W_1$

$$W_1 = \frac{R}{1-\rho_1} \quad \text{where} \quad \rho_1 = \lambda_1 E[S_1]$$

To find the residual service time R, note that the customer in the server can be a high or low priority customer, we have

$$R = \frac{1}{2}E[S_1^2]\lambda_1 + \frac{1}{2}E[S_2^2]\lambda_2$$

Low priority:

S_2 - service time for Class 2 with mean $E[S_2]$
W_2 = mean waiting time for Class 2 customers
N_2 = number of Class 2 customers in the queue
R = mean residual service time when a customer arrives

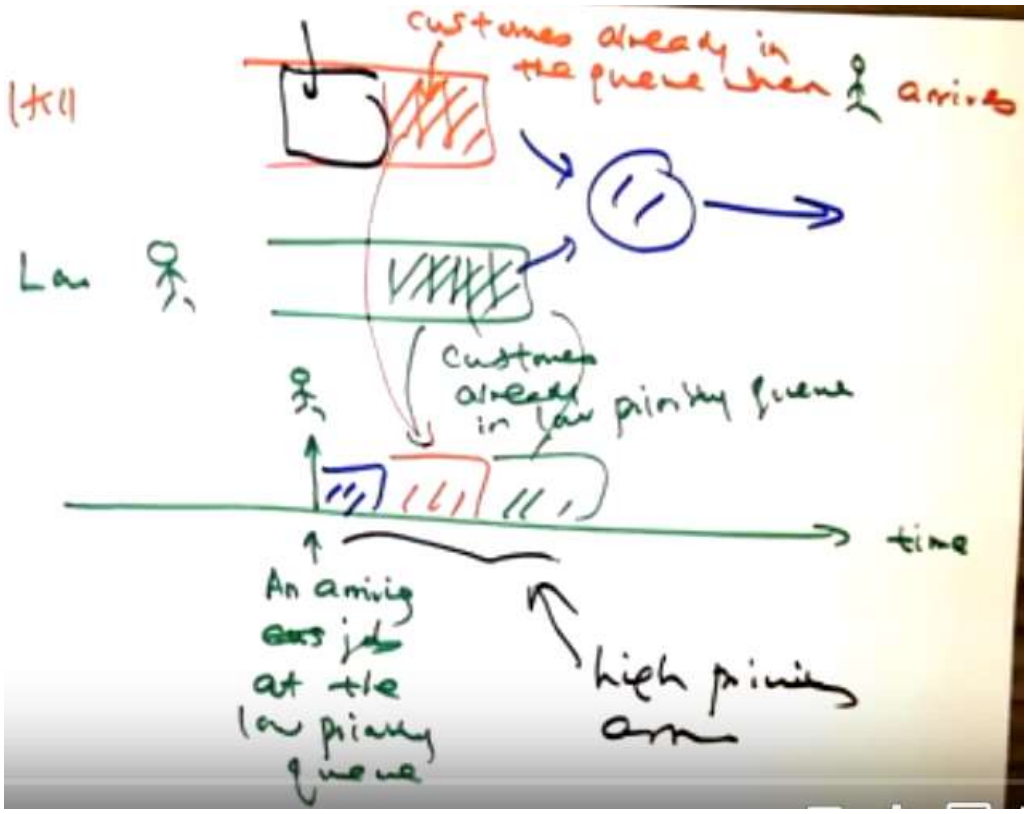
- For Class 2 customers:

$$W_2 = R + N_2E[S_2] + N_1E[S_1] + \lambda_1 W_2 E[S_1]$$

Average number of customers already in Queues 1 and 2 when a Class 2 customer arrives

Average number of customers that arrive in Queue 1 after a low priority customer arrives

一共有四个部分的时间需要等待，一个是已经在服务器中的，一个是在高优先级队列中的，一个是在低优先级队列中的，还有一个是在为低优先级队列服务的时候，插入进来的高优先级的队列。如下图所示。最后一个部分的推理可以理解为，用 W2 的等待时间乘以高优先级队列进入的速率，即可以得到，W2 的等待时间中，有多少个高优先级队列进入，再用它乘以高优先级队列的服务时间即可（Little's law）。



$$W_2 = R + N_2 E[S_2] + N_1 E[S_1] + \lambda_1 W_2 E[S_1]$$

- Little's Law to Queue 1:

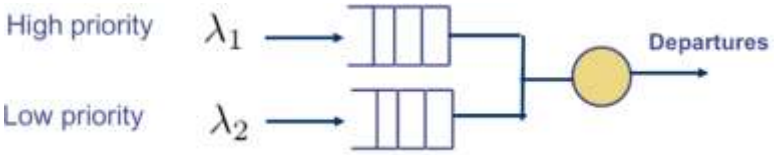
$$N_1 = \lambda_1 W_1$$

- Little's Law to Queue 2:

$$N_2 = \lambda_2 W_2$$

- Combining all of the above

$$W_2 = \frac{R + \rho_1 W_1}{1 - \rho_1 - \rho_2}$$ Where $\rho_2 = \lambda_2 E[S_2]$, $\rho_1 = \lambda_1 E[S_1]$



$$W_2 = \frac{R}{(1-\rho_1)(1-\rho_1-\rho_2)}$$

$$W_1 = \frac{R}{1-\rho_1}$$ where $\rho_1 = \lambda_1 E[S_1]$, $\rho_2 = \lambda_2 E[S_2]$, $R = \frac{1}{2}E[S_1^2]\lambda_1 + \frac{1}{2}E[S_2^2]\lambda_2$

Non-preemptive Priority with P classes

Waiting time of priority class k

$$W_k = \frac{R}{(1-\rho_1-\ldots-\rho_{k-1})(1-\rho_1-\ldots-\rho_k)}$$

where $$R = \frac{1}{2}\sum_{i=1}^{P} E[S_i^2]\lambda_i$$

$$\rho_i = \lambda_i E[S_i] \text{ for } i = 1, \ldots, P$$

$$W_2 = W_1 \cdot \frac{1}{1-\rho_1-\rho_2}$$

意味着 W2 比 W1 大，因为 1-p1-p2 肯定是小于 1 的，低优先级队列等待时间更长。

Example

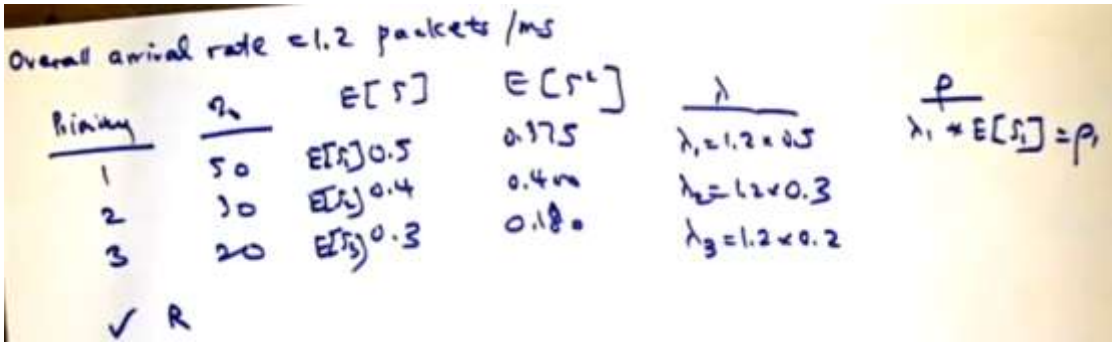
Router receives packet at 1.2 packets/ms (Poisson), only one outgoing link

Assume 50% packet of priority1, 30% of priority 2 and 20% of priority 3. Mean and second moment given in the table below.

What is the average waiting time per class?

Solution to be discussed in class.

Priority	Mean (ms)	2nd Moment (ms^2)
1	0.5	0.375
2	0.4	0.400
3	0.3	0.180



$$W_1 = \frac{R}{1-\rho_1}$$

$$W_2 = \frac{R}{(1-\rho_1)(1-\rho_1-\rho_2)}$$

$$W_3 = \frac{R}{(1-\rho_1-\rho_2)(1-\rho_1-\rho_2-\rho_3)}$$

Pre-emptive resume priority

- Can be derived using a similar method to that used for non-preemptive priority
- The key issue to note is that a job with priority k can be interrupted by a job of higher priority even when it is in the server
- For *k = 1* (highest priority), the response time T_1 is:

$$T_1 = E[S_1] + \frac{R_1}{(1-\rho_1)} \quad \text{where} \quad R_1 = \frac{1}{2}E[S_1^2]\lambda_1, \quad \rho_1 = E[S_1]\lambda_1$$

A highest priority job only has to wait for the highest priority jobs in front of it.

- For k ≥ 2, we have response time for a job in Class k:

$$T_k = E[S_k] + \frac{R_k}{1-\rho_1-\ldots-\rho_k} + (\sum_{i=1}^{k-1}\rho_i)T_k$$

An arriving job in Priority Class k needs to wait for all the jobs in Priority Classes 1 to k, that are already in the system when it arrives, to complete.

An arriving job of priority *k* has to wait for all the jobs of higher priorities that arrive during the time that this job is waiting in the queue and in the server.

$$R_k = \frac{1}{2}\sum_{i=1}^{k}E[S_i^2]\lambda_i$$

Note that R_k contains only terms of priority *k* or higher since a job with priority *k* cannot be interrupted by jobs with a lower priority. In other words, a job with priority *k* does not see the residual service time of lower priority classes.

- Solving these equations, we have the response time of Class k jobs is: $T_k = T_{k,1} + T_{k,2}$

where

$$T_{k,1} = \frac{E[S_k]}{(1 - \rho_1 - ... - \rho_{k-1})}$$

$$T_{k,2} = \frac{R_k}{(1 - \rho_1 - ... - \rho_{k-1})(1 - \rho_1 - ... - \rho_k)}$$

$$R_k = \frac{1}{2} \sum_{i=1}^{k} E[S_i^2] \lambda_i$$

Other queuing disciplines

- There are many other queueing disciplines, examples include
  - Shortest processing time first
  - Shortest remaining processing time first
  - Shortest expected processing time first

Week 4B_2: Generating random numbers

We have so far used mathematical methods to determine the performance of queues or queueing networks

Unfortunately many queues are not analytically tractable

- You can get upper bound of mean response time of G/G/1 but what if you want to estimate it?

Another method to study queue performance is to use discrete event simulation which you will study in Week 5

In order to do discrete event simulation, you need to know how to generate random numbers of any probability distribution

Random number generator in C

- In C, the function *rand()* generates random integers between 0 and RAND_MAX
- E.g. The following program generates 10 random integers:

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  int i;

   for (i = 0; i < 10; i++)
    printf("%d\n",rand());

  return;

}
```

Let us generate 10,000 random integers using rand() and see how they are distributed

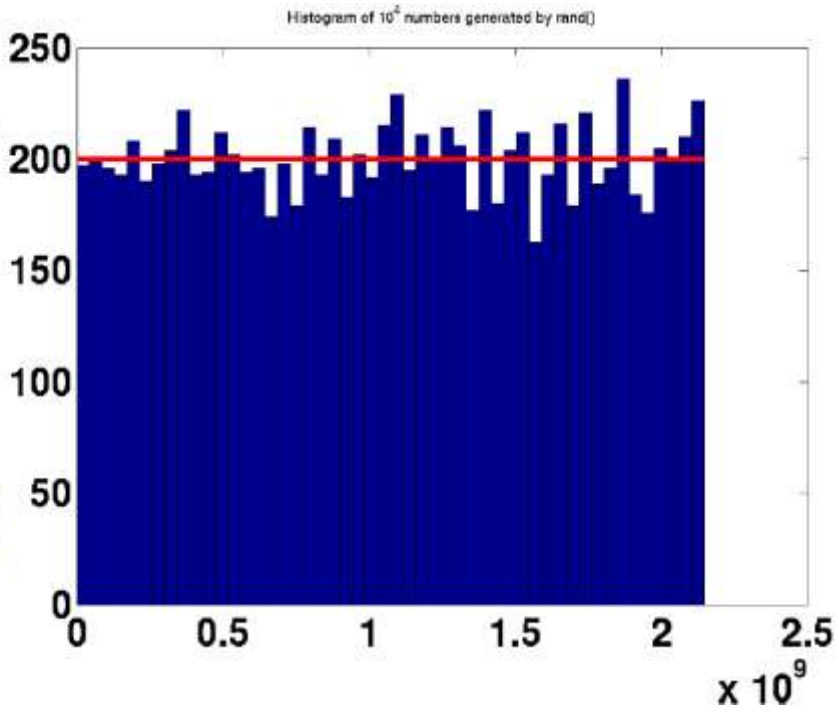
This C file "genrand1.c" is available from the course web site.

Distribution of 10000 entries from rand()

Sort into 50 bins

If the numbers are really uniformly distributed, we expect 200 numbers in each bin.

The numbers are almost uniformly distributed



LCG

- The random number generator in C is a Linear Congruential Generator (LCG)
- LCG generates a sequence of integers $\{Z_1, Z_2, Z_3, \ldots\}$ according to the recursion

  $Z_k = a\, Z_{k-1} + c \pmod m$

  where *a, c* and *m* are integers
- By choosing a, c, m, Z_1 appropriately, we can obtain a sequence of seemingly random integers
- If a = 3, c = 0, m = 5, Z_1 = 1, LCG generates the sequence 1, 3, 4, 2, 1, 3, 4, 2, …
- *Fact:* The sequence generated by LCG has a cycle of *m-1*
- We must choose *m* to be a large integer
  - For C, m = 2^{31}
- The proper name for the numbers generated is *pseudo-random numbers*

Seed

- LCG generates a sequence of integers $\{Z_1, Z_2, Z_3, \ldots\}$ according to the recursion

  $Z_k = a\, Z_{k-1} + c \pmod m$

  where *a, c* and *m* are integers
- The term Z_1 is call a seed
- By default, C also uses 1 as the seed and it will generate the same random sequence
- However, sometimes you need to generate different random sequences and you can change the seed by calling the function *srand()* before using *rand()*
  - Demo genrand1.c, genrand2.c and genrand3.m
  - *genrand1.c – uses the default seed*
  - *genrand2.c – sets the seed using command line argument*
  - *genrand3.c – sets the seed using current time*

Uniformly distributed random numbers between (0,1)

With rand() in C, you can generate uniformly distributed random numbers in between 1 and 2^{31}-1(= RAND_MAX)

- By dividing the numbers by RAND_MAX, you get randomly distributed numbers in (0,1)

In Matlab, *rand(n,1)* generates a sequence of *n* uniformly distributed random numbers in (0,1)

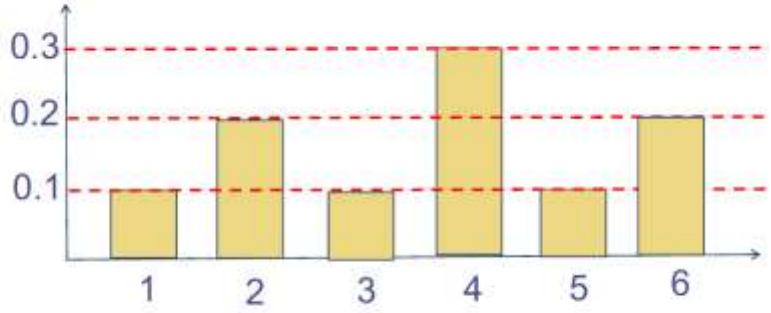
- Matlab uses the Mersenne Twister random number generator with a period of $2^{19937} - 1$
  - The Python random module uses the same generator
- If you use 10^9 random number in a second, the sequence will only repeat after 10^{5985} years

Why are uniformly distributed random numbers important?

- If you can generate uniformly distributed random numbers between (0,1), you can generate random numbers for any probability distribution

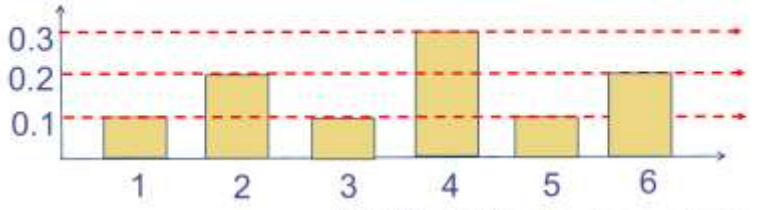
A loaded die

- You want to create a loaded die with probability mass function



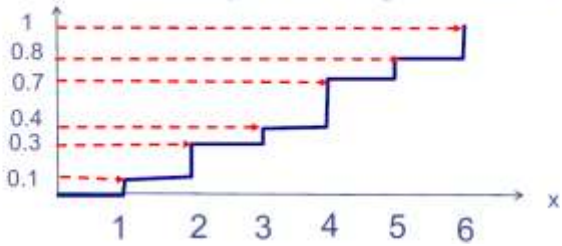
- The algorithm is:
  - Generate a random number u
  - If u < 0.1, output 1
  - If 0.1 ≤ u < 0.3, output 2
  - If 0.3 ≤ u < 0.4, output 3
  - If 0.4 ≤ u < 0.7, output 4
  - If 0.7 ≤ u < 0.8, output 5
  - If 0.8 ≤ u , output 6

Cumulative probability distribution



Probability that the dice gives a value ≤ x

Ex: Can you work out what these levels should be



Comparing algorithm with cumulative distribution

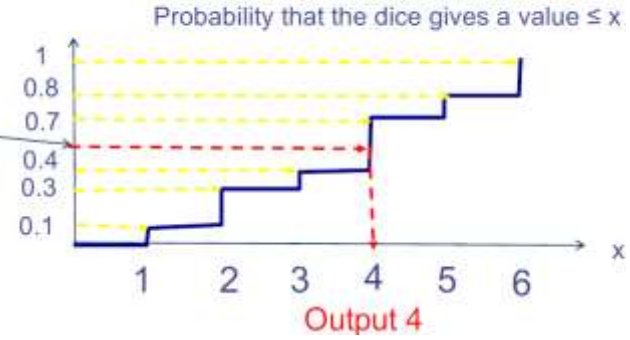
- The algorithm is:
  - Generate a random number u
  - If u < 0.1, output 1
  - If 0.1 ≤ u < 0.3, output 2
  - If 0.3 ≤ u < 0.4, output 3
  - If 0.4 ≤ u < 0.7, output 4
  - If 0.7 ≤ u < 0.8, output 5
  - If 0.8 ≤ u , output 6

Ex: What do you notice about the intervals in the algorithm and the cumulative distribution?

Graphical interpretation of the algorithm

- The algorithm is:
  - Generate a random number u
  - If u < 0.1, output 1
  - If 0.1 ≤ u < 0.3, output 2
  - If 0.3 ≤ u < 0.4, output 3
  - If 0.4 ≤ u < 0.7, output 4
  - If 0.7 ≤ u < 0.8, output 5
  - If 0.8 ≤ u , output 6

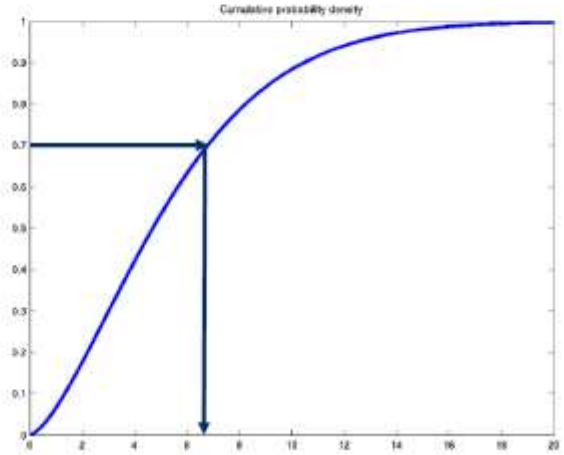
Ex: Let us assume u = 0.5126, what should the algorithm output?



Graphical representation of inverse transform method

- Consider the cumulative density function (CDF) y = *F(x)*, showed in the figure below

For this particular F(x), if u = 0.7 is generated then $F^{-1}(0.7)$ is 6.8



Inverse transform method

- A method to generate random number from a particular distribution is the *inverse transform method*

- In general, if you want to generate random numbers with cumulative density function (CDF) $F(x) = Prob[X \leq x]$, you can use the following procedure:
  - Generate a number u which is uniformly distributed in (0,1)
  - Compute the number $F^{-1}(u)$

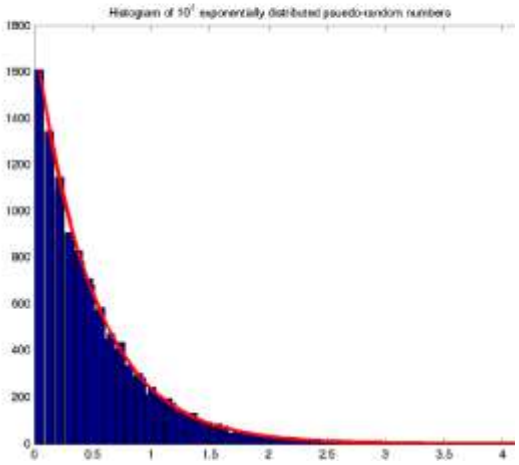
- Example: Let us apply the inverse transform method to the exponential distribution
  - *CDF is 1 - exp(- λx)*

Generating exponential distribution

- Given a sequence $\{U_1, U_2, U_3, \ldots\}$ which is uniformly distributed in (0,1)
- The sequence - log(1 - U_k)/ λ is exponentially distributed with rate λ

- (Matlab file hist_expon.m)

1. Generate 10,000 uniformly distributed numbers in (0,1)
2. Compute -log(1-u_k)/2 where u_k are the numbers generated in Step 1
3. The plot shows
   1. The histogram of the numbers generated in Step 2 in 50 bins
   2. The red line show the expected number of exponential distributed numbers in each bin



Week 4B: Revision problems

A computer system receives requests from a Poisson process at a rate of 10 requests/s. Assume that 30% of the requests are of type *a* and the remaining are of type *b*.

For request type *a*, its average service time is 0.1 seconds and the coefficient of variation of its service time is 1.5.

For request type *b*, its average service time is 0.08 seconds and the coefficient of variation of its service time is 1.2.

Compute the average response time for each type of request under the following scenarios:

1. Requests of types *a* and *b* have equal priorities
2. Requests of type *b* have non-preemptive priority over requests of type *a*
3. Requests of type *b* have preemptive priority over requests of type *a*

计算机系统以 10 个请求/秒的速率接收来自泊松进程的请求。

假设 30%的请求是类型 A，其余的是类型 B。

对于请求类型 A，其平均服务时间为 0.1 秒，其服务时间的变化系数为 1.5。

对于请求类型 B，其平均服务时间为 0.08 秒，其服务时间的变化系数为 1.2。

在以下情况下计算每种请求类型的平均响应时间：

1。A 型和 B 型请求具有同等优先级

2。B 类请求比 A 类请求具有非优先权

3。B 类请求优先于 A 类请求

Let us first compute the second moment of each customer type. Let us use C_a, σ_a, $E[S_a]$ and $E[S_a^2]$ to denote, respectively, the coefficient of variation, standard deviation, mean and second moment of the service time of customer of type a. Recall that the coefficient of variation of a random variable is its standard deviation divided by mean, i.e $C_a = \frac{\sigma_a}{E[S_a]}$. By using the relation:

$$E[S_a^2] = E[S_a]^2 + \sigma_a^2, \tag{1}$$

it can be showed that

$$E[S_a^2] = E[S_a]^2(1 + C_a^2) \tag{2}$$

Since we know $E[S_a] = 0.1$ and $C_a = 1.5$, we can compute $E[S_a^2]$ using the above equation.

Similarly, let C_b, σ_b, $E[S_b]$ and $E[S_b^2]$ to denote, respectively, the coefficient of variation, standard deviation, mean and second moment of the service time of customer of type b. We have

$$E[S_b^2] = E[S_b]^2(1 + C_b^2) \tag{3}$$

With $E[S_b] = 0.08$ and $C_b = 1.2$, we can compute $E[S_b^2]$ using the above equation.

Requests of type a and b have equal priorities

This is an M/G/1 queue without priority. The arrival rate is 10 requests per second ($= \lambda$) . Since 30% of the request are type a and the remaining are type b, we have the mean service time $E[S]$ and second moment $E[S^2]$ of the aggregate are, respectively,

$$E[S] = 0.3E[S_a] + 0.7E[S_b] \tag{4}$$
$$E[S^2] = 0.3E[S_a^2] + 0.7E[S_b^2] \tag{5}$$

The mean response time is therefore $E[S] + \frac{\lambda E[S^2]}{2(1-\rho)}$ where $\rho = \lambda E[S]$.

Requests of type b have non-preemptive priority over type a

Let

$$R = \frac{1}{2}(0.3\lambda E[S_a^2] + 0.7\lambda E[S_b^2]) \tag{6}$$

$$\rho_a = 0.3\lambda E[S_a] \tag{7}$$
$$\rho_b = 0.7\lambda E[S_b] \tag{8}$$

Response time of type b is

$$E[S_b] + \frac{R}{1 - \rho_b} \tag{9}$$

Response time of type a is

$$E[S_a] + \frac{R}{(1 - \rho_b)(1 - \rho_a - \rho_b)} \tag{10}$$

Requests of type b have preemptive priority over type a

Let

$$R_b = \frac{1}{2}(0.7\lambda E[S_b^2]) \tag{11}$$

$$R_a = \frac{1}{2}(0.3\lambda E[S_a^2] + 0.7\lambda E[S_b^2]) \tag{12}$$

$$\rho_a = 0.3\lambda E[S_a] \tag{13}$$
$$\rho_b = 0.7\lambda E[S_b] \tag{14}$$

Response time of type b is

$$E[S_b] + \frac{R_b}{1-\rho_b} \tag{15}$$

Response time of type a is

$$E[S_a]\frac{1}{1-\rho_b} + \frac{R_a}{(1-\rho_b)(1-\rho_a-\rho_b)} \tag{16}$$

The numerical answers are summarised in the following table.

Mean response time	Part 1	Part 2	Part 3
type a	0.8246	1.7787	1.9059
type b	0.8246	0.3150	0.2042

Observe that the response time for type b customers have become better because it has a higher priority, this is of course at the expense of type a customers which have a lower priority.

对于第一小问，用$W_a = N_a E[S_a] + N_b E[S_b] + R$ 也是可以得到相同的答案的，只是第一小问其实就是相当于解决最常见的 M/G/1 的问题，所以可以直接用 P-K formula

The Weibull distribution with parameters α and β has a cumulative probability function $F(x) = 1 - \exp(-\alpha x^{\beta})$. Write a computer program to generate random numbers that have a Weibull distribution with $\alpha = 1.5$ and $\beta = 6$. Verify by using a histogram that the numbers that you have generated do have a Weibull distribution.

We first work out the formula that we should use to generate random numbers with the Weibull distribution using the inverse transform method. Assuming that y and u are related by $y = F^{-1}(u)$, we have

$$\begin{aligned} y &= F^{-1}(u) && (1)\\ \Leftrightarrow u &= F(y) && (2)\\ \Leftrightarrow u &= 1 - \exp(-\alpha y^{\beta}) && (3)\\ \Leftrightarrow y &= \left(-\frac{\log(1-u)}{\alpha}\right)^{\frac{1}{\beta}} && (4) \end{aligned}$$

Therefore we can use the relation

$$\left(-\frac{\log(1-u)}{\alpha}\right)^{\frac{1}{\beta}} \tag{5}$$

where u is uniformly distributed over $[0, 1]$ to generate the Weibull distribution.

By using the above formula, 10,000 (= n) numbers that are Weibull distributed are generated. Figure 1 shows a histogram of the 10,000 numbers sorted into 50 equally spaced bin.

We have also derived the expected number of random numbers in each bin and plotted it as the red curve in Figure 1. Note that the histogram consists of 50 equally spaced bin spanning $[0, y_{\max}]$ where $y_{\max}$ is the largest number generated. Define $\delta = \frac{y_{\max}}{50}$, then the k-th bin spans the range $[(k-1)\delta, k\delta]$.

The probability that a number with Weibull distribution falls within the range $[(k-1)\delta, k\delta]$ is $(F(k\delta) - F((k-1)\delta))$. (Note: You can prove this by using the definition of cumulative density function.) If n random numbers are generated, then the expected number of random numbers in the k-th bin is $n(F(k\delta) - F((k-1)\delta))$

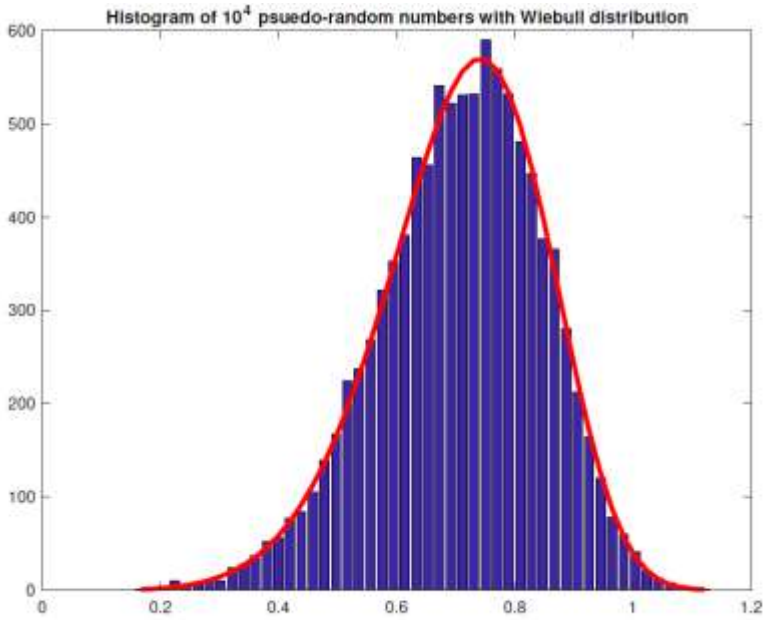


Figure 1: Histogram of 10000 random numbers with Weibull distribution. The red curve shows the expected distribution.

Week 5A: Discrete event simulation

- Queues with general inter-arrival and service time distributions

General Inter-arrivals time distribution
General service time distribution
Arrivals
Departures

- M/G/1 queue
  - Can calculate delay with the P-K formula

$$W = \frac{\lambda E[S^2]}{2(1-\rho)}$$

- G/G/1 queue
  - No explicit formula, get a bound or approximation

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\rho)}$$

- You had learnt how to solve a number of queues analytically (= mathematically) given their
  - Inter-arrival time probability distribution
  - Service time probability distribution
- Queues that you can solve now include M/M/1, M/M/m, M/G/1, M/G/1 with priorities etc.
  - If you know the analytical solution, this is often the most straightforwad way to solve a queueing problem
- Unfortunately, *many queueing problems are still analytically intractable!*
- What can you do if we have an analytically intractable queueing problem?

Lectures 4B, 5A, 5B, 6A: Discrete event simulation

- For a number of lectures, we look at the topic of using *discrete event simulation for queueing problems*
  - *Simulation is an imitation of the operation of real-life system over time.*

- The topics to be covered are
  - (4B) How to generate pseudo-random numbers for simulation?
  - (5A) What are discrete event simulation?
  - (5A) How to structure a discrete event simulation?

  - For 5B and 6A
  - How to choose simulation parameters?
  - How to analyse data?
  - What are the pitfalls that you need to avoid?

Motivating example

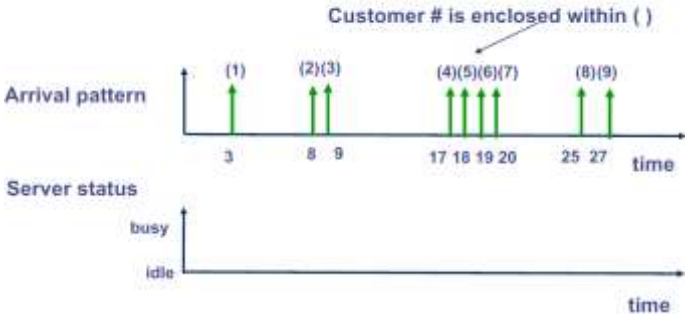
- Consider a single-server queue with only one buffer space (= waiting room)
- If a customer arrives when the buffer is occupied, the customer is rejected.
- Given the arrival times and service times in the table on the right, find
  - The mean response time
  - % of rejected customers

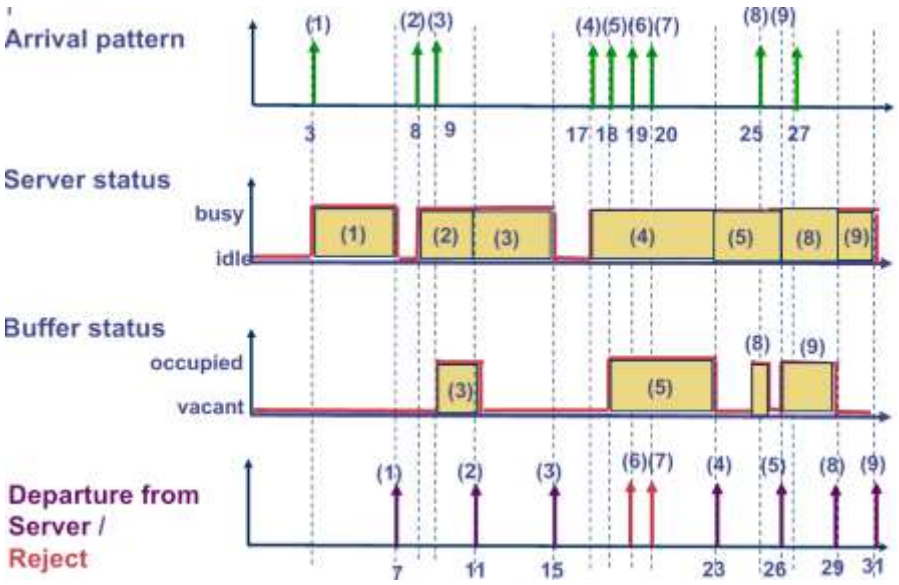
  Assuming an idle server at time = 0.

Customer number	Arrival time	Service time
1	3	4
2	8	3
3	9	4
4	17	6
5	18	3
6	19	2
7	20	2
8	25	3
9	27	2

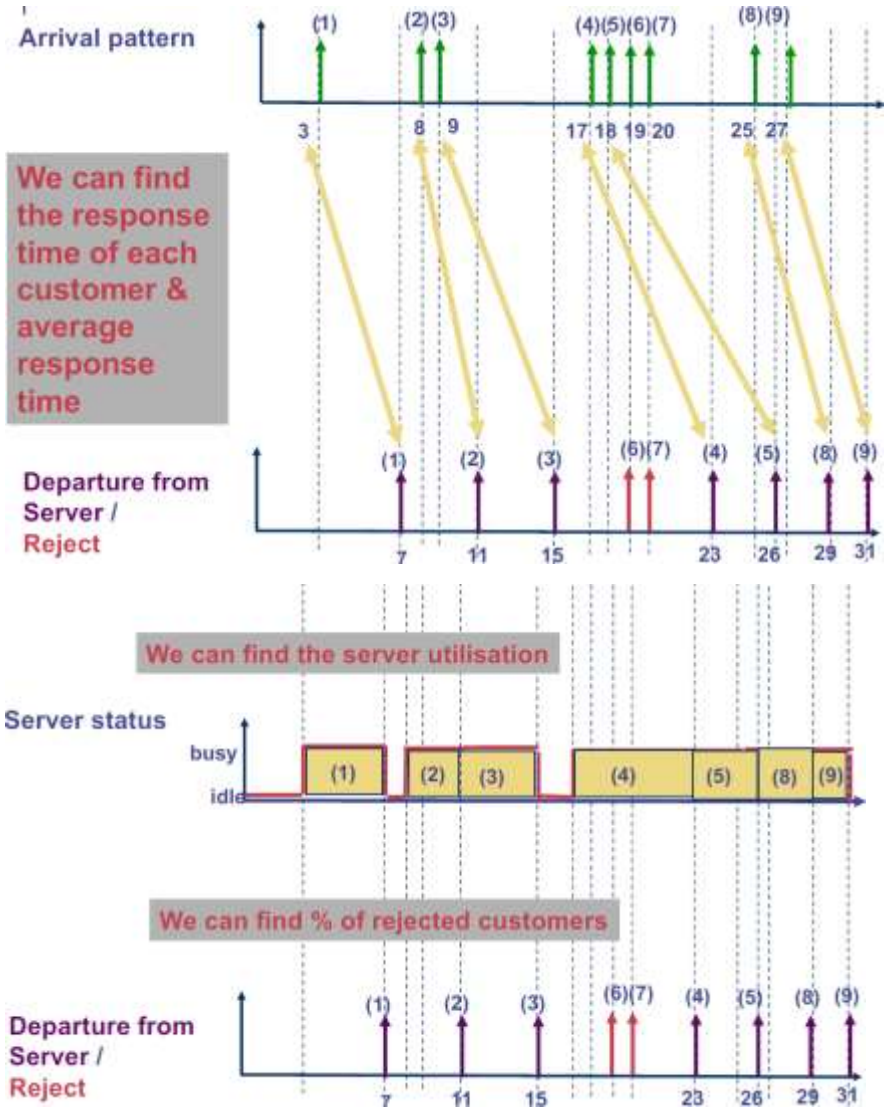
A graphical solution

- In the graphical solution, we will keep track of
  - The status of the server: busy or idle
  - The status of the buffer: occupied or vacant





Using the graphical solution



From graphical solution to computer solution

How can we turn this graphical solution into a computer solution, i.e. a computer program that can solve the problem for us

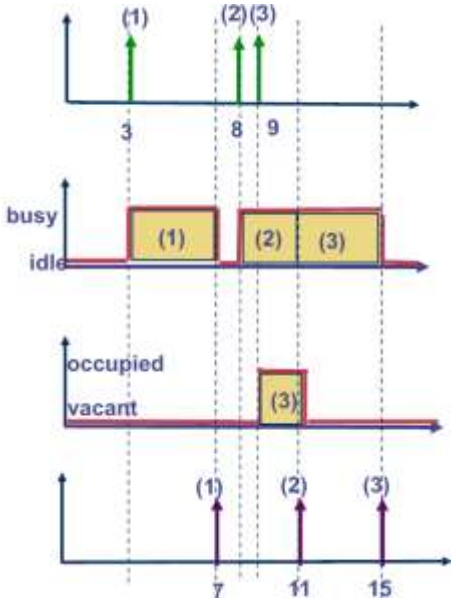
We need to keep track of the status of the server and the status of the buffer,

- This allows us to make decisions
- E.g. If server is BUSY and buffer is OCCUIPIED, an arriving customer is rejected.
- E.g. If server is BUSY and buffer is VACANT, an arriving customer goes to the buffer.
- E.g. If server is IDLE, an arriving customer goes to the sever

What this means: We need to keep track of the status of some variables in our computer solution.

Observation #1:

- An arriving or departing customer causes the server or buffer status to change
- Examples:
  - At time = 3, the arrival of customer #1 causes the server to switch from IDLE to BUSY
  - At time = 7, the departure of customer #1 causes the server to switch from BUSY to IDLE
  - At time = 9, the arrival of customer #3 causes the buffer to switch from VACANT to OCCUPIED
  - Etc.



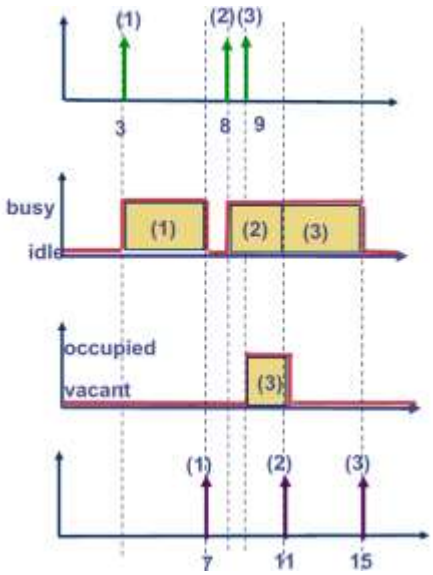
Let us call the arrival of a customer or the departure of a customer an ***event***

Observation #2:

- The status of the server and the status of the buffer remain the same between two consecutive events

What this means:

- We need to keep track of the timing of the events
  - Events can cause status transitions
  - In between events, status remain the same



In our computer solution, we will use a ***master clock*** to keep track of the current time

We will advance the master clock from event to event

In order to see how the computer solution works, let us try it out on paper first

On paper simulation

- In our simulation, we keep track of a number of variables
  - MC = Master clock
  - Status of
    - Server: 1 = BUSY, 0 = IDLE
    - Buffer: 1 = OCCUPIED, 0 = VACANT
  - Event time:
    - Next arrival event and service time of this arrival
    - Next departure event and arrival time of this departure
  - The (arrival time, service time) of the customer in buffer
  - In order to compute the response time, we keep track of
    - The cumulative response time (T)
    - Cumulative number of customers rejected (R)

MC	Next arrival		Next departure		Server status	Buffer status + customer in buffer	T	R
	Arrival time	Service time	Departure time	Arrival time of this departure				
0	3	4	-	-	0	0	0	0
3	8	3	7	3	1	0	0	0
7	8	3	-	-	0	0	4	0

MC	Next arrival		Next departure		Server status	Buffer status + Customer in buffer	T	R
	Arrival time	Service time	Departure time	Arrival time of this departure				
0	3	4	-	-	0	0	0	0
3	8	3	7	3	1	0	0	0
7	8	3	-	-	0	0	4	0
8	9	4	11	8	1	0	4	0
9	17	6	11	8	1	1 (9,4)	4	0
11	17	6	15	9	1	0	7	0
15	17	6	-	-	0	0	13	0

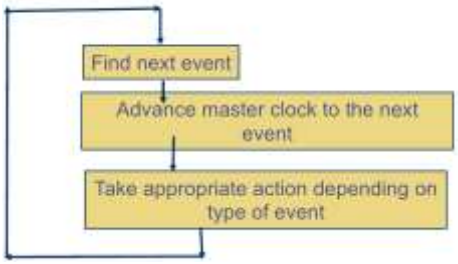
Can you continue?

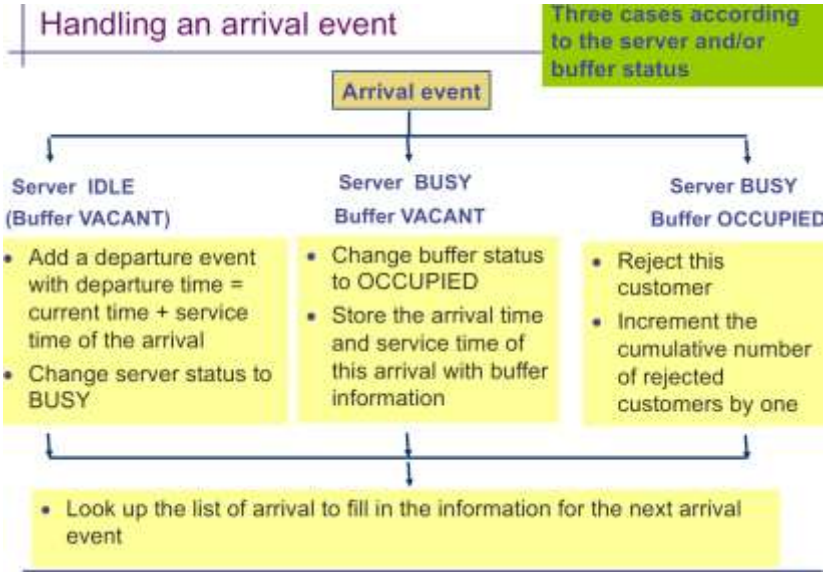
(Arrival time, service time) of the customer in the buffer.

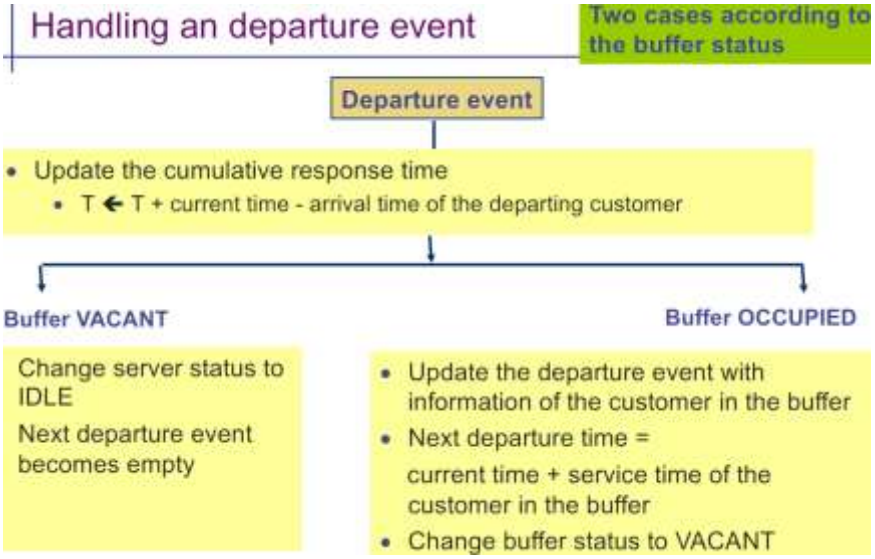
MC	Next arrival		Next departure		Server status	Buffer status + Customer in buffer	T	R
	Arrival time	Service time	Departure time	Arrival time of this departure				
0	3	4	-	-	0	0	0	0
3	8	3	7	3	1	0	0	0
7	8	3	-	-	0	0	4	0
8	9	4	11 8	8	1	0	4	0
9	17	6	11	8	1	(9,4) 1	4	0
11	17	6	15	9	1	0	7	0
15	17	6	-	-	0	0	13	0
17	18	3	23	17	1	0	13	0
18	19	2	23	17	1	(18,3) 1	13	0
19	20	2	23	17	1	(19,2) 1	13	1
20	25	2	23	17	1	(19,2) 1	13	2

Logic of the program

- At each step, we advance to the next event that will take place







Discrete event simulation

The above computer program is an example of a discrete event simulation

It allows you to solve a queueing problem with one server and one buffer space

You can generalise the above procedure to

- Multi-server
- Finite or infinite buffer space
- Different queueing disciplines

Let us generalise it to the case of single-server with infinite buffer

Single server with infinite buffer simulation

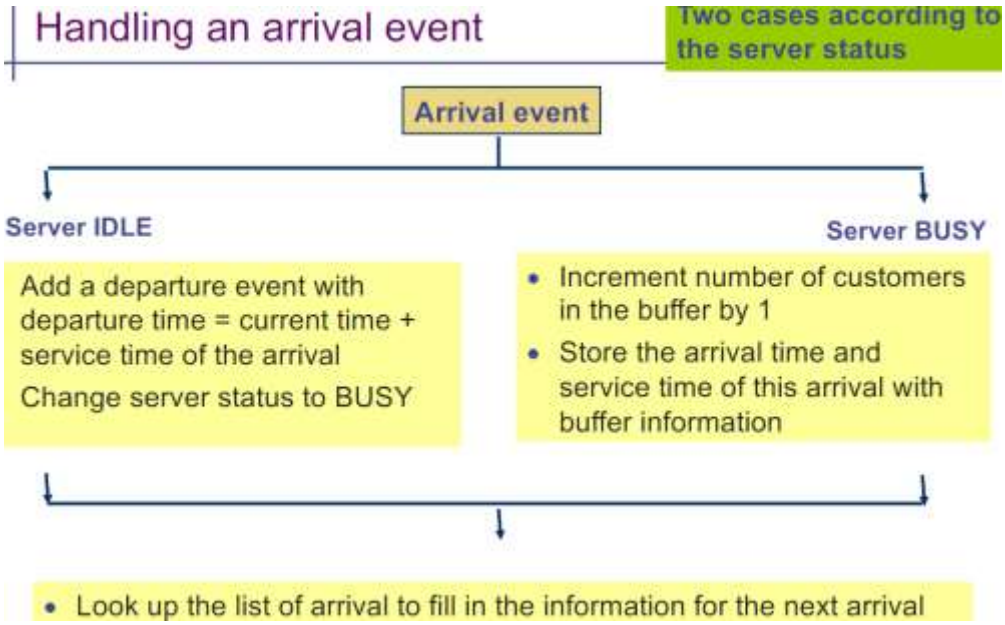
In this case, we will use buffer status to denote the number of customers in buffer

- Buffer status = 0, 1, 2, 3, …

We also need to store all the (arrival time, service time) of all the customers in the buffer

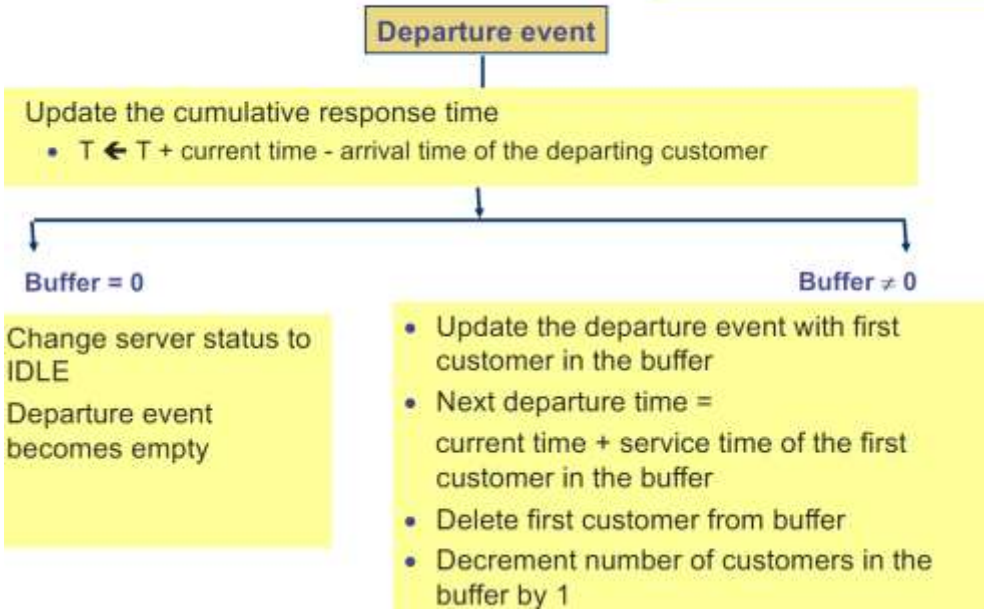
Compare with the single-server single-buffer case, we only need to change the handling of

- An arrival event
- A departing event



Handling an departure event

Two cases according to the buffer status



We know how to write a discrete event simulation program to simulate a single-server queue with infinite buffer

We know how to generate random numbers

- From Lecture 4B

This will allow us to simulate a G/G/1 queue provided that we can generate the probability distribution

In order to test how well our discrete event simulation program works, we will use it to simulate an M/M/1 queue and compare it with the expected result

An M/M/1 simulation program (based on Matlab) is given ir *sim_mm1.m* (available on the course web site)

Reproducible simulation

- We run the simulation sim_mm1.m a few times, we get mean response times of 0.98623, 0.98445, 1.0034, …
- Each simulation run gives a different result because different set of random numbers is used
- In order to realise reproducibility of results, you can save the setting of the random number generator before simulation. If you reuse the setting later, you can reproduce the result

```
% obtain setting and save it in a file
rand_setting = rng;
save saved_rand_setting rand_setting
sim_mm1
```

```
% load the save setting and apply it
load saved_rand_setting
rng(rand_setting)
sim_mm1
```

Trace driven simulation

We considered this example in the beginning of this lecture

We simulated using

- A sequence (or trace) of arrival times
- A sequence of service times

We call this trace driven simulation

Trace driven simulation is useful

- You have a server and you have a log of the arrival time and service time of the job
- You are considering changing to a new server
- You can use the traces that you have and simulation to calculate the response time of the new server

Customer number	Arrival time	Service time
1	3	4
2	8	3
3	9	4
4	17	6
5	18	3
6	19	2
7	20	2
8	25	3
9	27	2

标准差公式:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Week 5A: Revision problems

The Matlab program *sim_mm1_func.m* simulates an M/M/1 queue with arrival rate λ and service rate μ over a time period of T. It returns the average response time for the given λ, μ and T.

We would like to investigate the effect of the length of simulation T on the simulation. Let us fix $\lambda = 0.7$ and $\mu = 1$. For each of the following values of T: 1000, 5000, 10000 and 50000, perform the simulation 20 times (with a different set of random numbers) and record the value of the average mean response time.

Answer the following:

1. What is the mean response time according to the M/M/1 result?
2. For each value of T used, compute the mean and standard deviation over 20 experiments.
3. How does the standard deviation vary with T?

Note on using *sim_mm1_func.m*: If you type at the matlab prompt: *sim_mm1_func(0.7,1,1000)*, it will simulate an M/M/1 queue with $\lambda = 0.7$, $\mu = 1$ and $T = 1000$.

The mean response given by the M/M/1 theory is $\frac{1}{\mu-\lambda}$. For the given values of λ and μ, we expect the mean response time to be 3.3333s.

The Matlab file *week05A_q1.m* contains a script which, for each value of T, performs 20 simulation experiments (or replications). This is done by calling the function *sim_mm1_func.m* 20 times for each value of T.

The results are plotted in Figure 1. The horizontal axis is the value of T and the vertical axis is mean response time. The horizontal line shows the value given by the M/M/1 theory. The circle shows the mean response time given by the simulation experiments. You can see that every simulation run produces a slightly different value of mean response time. However, the values appear to be scattered around the mean response time given by the M/M/1 theory. You can graphically observe that the standard deviation becomes smaller with increasing T.

The mean and standard value from these simulations are:

	T = 1000	T = 5000	T = 10000	T = 50000
mean	3.2232	3.3177	3.3949	3.3321
standard deviation	0.7066	0.3665	0.2212	0.1313

Remark: If you want to reproduce exactly the above results, you need to know the setting (similar to a seed) of the generator that I have used to generate the random numbers. The setting is stored in *week05A_q1_rand_setting.mat*. If you use this given setting (the file *week05A_q1.m* contains comments to tell you how to do this), you will be able to reproduce the results above.

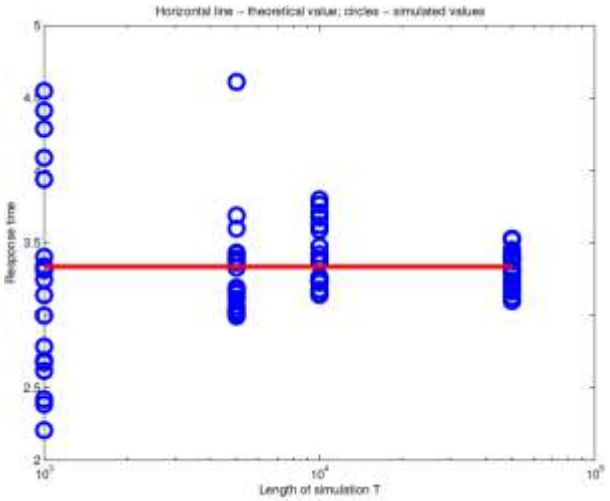


Figure 1: Response time from M/M/1 simulations of different length of time.

Write a simulation program (in whatever language you prefer) to simulate an M/M/2 queue. You should be able to control the arrival rate λ, service rate μ and the length of simulation T.

Use your M/M/2 and M/M/1 simulation program to compare:

1. The mean response time of an M/M/1 queue with $\lambda = 0.9$ and $\mu = 1$.
2. The mean response time of an M/M/2 queue with $\lambda = 0.9$ and for each server, $\mu = 0.5$.

Simulate each of the above configurations 10 times and record the mean response time in the simulation. You may use a simulation time of $T = 1000$.

You have learnt in Week 3 that the the first system should have a smaller mean response time. Did your simulation results also suggest a smaller mean response time for the first system? s

Recall that in simulating a single-server queue, we maintain the following events and variables

- Events
  - Arrival to the system
  - Departure from the server
- Variables
  - The status of the server: either idle or busy
  - The number of customers in the buffer and their attributes

For an m-server queue, you will need to keep track of the departure from each of the m servers. In addition, you will need to keep track of the busy/idle status of all the m servers. These are the two main alternations that you need to make to adapt a single-server simulation program to a multi-server simulation program.

A program called *sim_mmm_func.m* has been written to simulate an M/M/m queue.

Simulation experiments have been carried out to simulate these two queues.

1. The mean response time of an M/M/1 queue with $\lambda = 0.9$ and $\mu = 1$.
2. The mean response time of an M/M/2 queue with $\lambda = 0.9$ and for each server, $\mu = 0.5$.

Each configuration is simulated 10 times. Each simulation is carried out using an independent set of random numbers. In other words, altogether of 20 set of independent random numbers are used in the simulation.

The results of the simulation comparing the two configurations have been plotted in Figure 2. The figure also shows the mean response time predicted by the M/M/1 and M/M/2 queueing theory.

Five experiments suggest that Configuration 1 is better and the five experiments suggest that Configuration 2 is better. It does not seem to be possible to conclude which configuration is better from the simulation results.

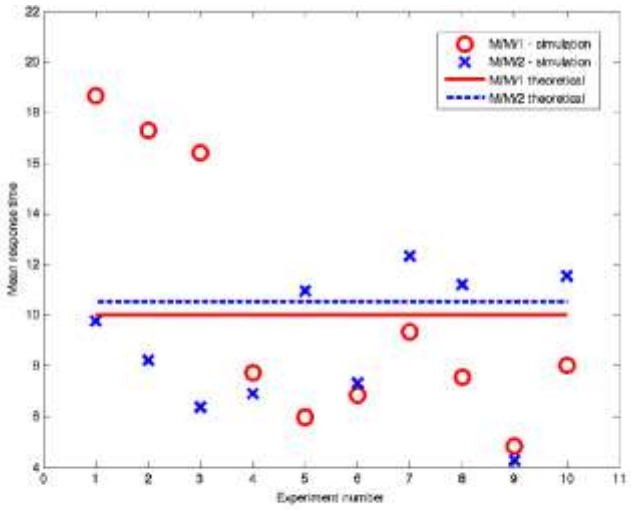


Figure 2: Comparison of an M/M/1 queue with $\lambda = 0.9$ and $\mu = 1$ and M/M/2 queue with $\lambda = 0.9$ and $\mu = 0.5$.

Week 5B: Mean Value Analysis

Methods to *efficiently* analyse a closed queueing network

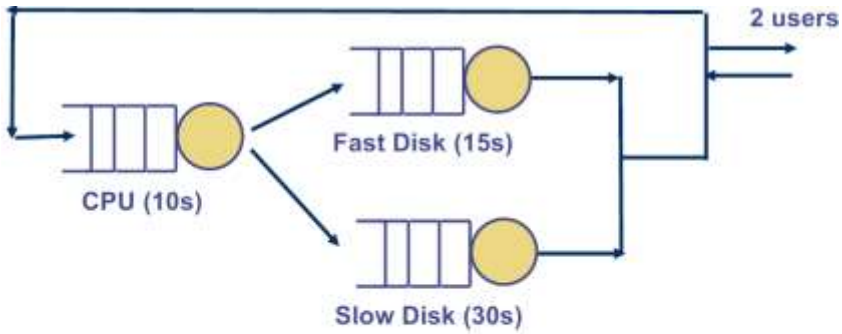
Motivation

- You have learnt how to analyse a closed queueing network in Week 3B using Markov chain
- However, the method can only be used for a small number of users

This week we will study a method that can be used for a large number of users

Let us begin by revisiting the database server example in Week 3B

DB server example



- 1 CPU, 1 fast disk, 1 slow disk.
- Peak demand = 2 users in the system all the time.
- Transactions alternate between CPU and disks.
- The transactions will equally likely find files on either disk
- Service time are exponentially distributed with mean showed in parentheses.

Markov chain solution to the DB server problem

In Week 3B, we used Markov chain to solve this problem

We use a 3-tuple (X,Y,Z) as the state

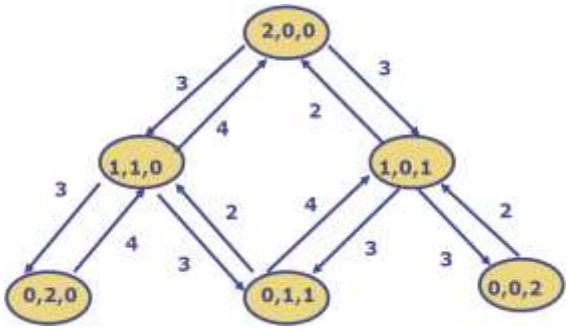
- X is # users at CPU
- Y is # users at fast disk
- Z is # users at slow disk

Examples

- (2,0,0): both users at CPU
- (1,0,1): one user at CPU and one user at slow disk

Six possible states

- (2,0,0) (1,1,0) (1,0,1) (0,2,0) (0,1,1) (0,0,2)



Solving the model

Solve for the probability in each state P(2,0,0), P(1,1,0) etc.

- There are 6 states so we need 6 equations

After solving for P(2,0,0), P(1,1,0) etc. we can find

- Utilisation
- Throughput,
- Response time,
- Average number of users in each component etc.

What if we have 3 users instead?

What if we have 3 users in the database example instead of only 2 users?

We continue to use (X,Y,Z) as the state

- X is the # users at CPU
- Y is the # users at the fast disk
- Z is the # users at the slow disk

How many states will you need?

We need 10 states:

- (3,0,0),
- (2,1,0),(2,0,1)
- (1,2,0),(1,1,1),(1,0,2)
- (0,3,0),(0,2,1),(0,1,2),(0,0,3)

What if there are n users?

You can show that if there are *n* users in the database server, the number of states *m* required will be

$$\frac{(n+1)(n+2)}{2}$$

For n = 100, m (= #states) ~ 50000

You can automate the computational process but where is the computational bottleneck?

- Solving a system of m linear equations in m unknowns has a complexity of $O(m^3)$

For our database server with n users, the computational complexity is about $O(n^6)$

Weaknesses of Markov model

The Markov model for a practical system will require many states due to

- Large number of users
- Large number of components

Large # states

- More transitions to identify
  - Though this can be automated
- If you've m states, you need to solve a set of m equations. A larger set of equation to solve.
  - The complexity of solving a set of m linear equations in m unknowns is $O(m^3)$

Mean value analysis (MVA)

An iterative method to find the

- Utilisation
- Mean throughput
- Mean response time
- Mean number of users

The complexity is approximately $O(nk)$ where

- n is the number of users
- k is the number of devices

The complexity of MVA makes it a very practical method

MVA – overview

MVA analysis has been derived for

- Closed model
  - Single-class
  - Multi-class
- Open model
- Mixed model with both open and closed queueing

This lecture discusses MVA for single-class closed model

MVA for closed system

Consider a closed queueing network with a single-class of customers

You are given a system with *K* devices

You are given that each customer

- Visits device *j* on average *V(j)* times
- Requires a mean service time of *S(j)* from device *j*
  - *Note: The service time required is assumed to be exponentially distributed*

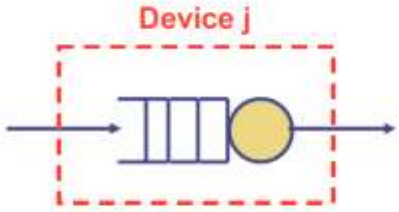
From the information given, we can deduce that the service demand *D(j)* for device *j* is *V(j) S(j)*

How do we obtain *D(j)* for a practical system?

Key idea behind MVA is *iteration*

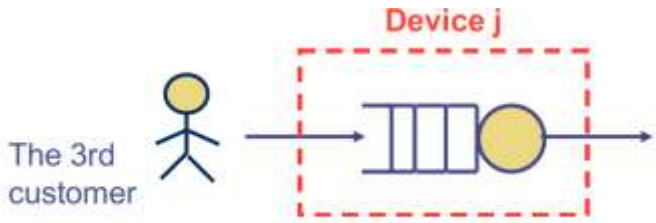
- If you know the solution to the problem when there are *n* customers in the system, you can find the solution when there are *(n+1)* customers

Let us consider a simple example to motivate the iteration in MVA. Consider device j (say) of a queueing network.

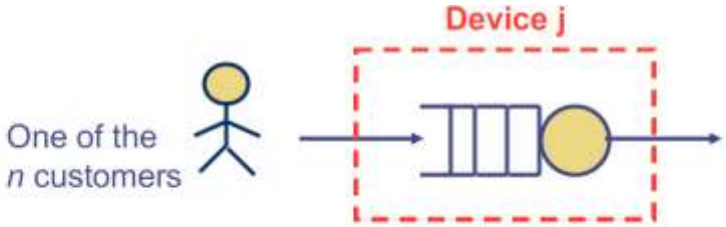


Assume that we know when there are 2 customers in the system, the average number of users in device j is 0.6 (say).

What happens when there are 3 customers?



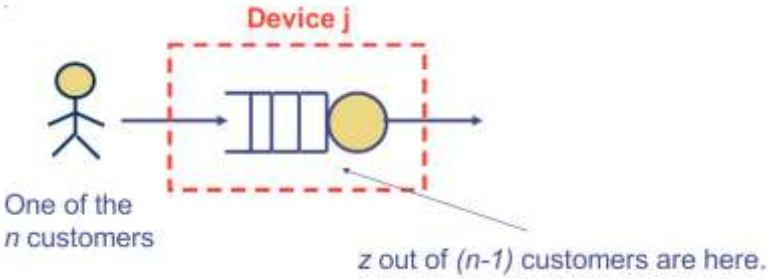
Let us assume the 3rd customer is arriving at device *j*.
Where will the other 2 customers be? We cannot tell exactly but we know that there is on average of 0.6 customers in device *j* when there are 2 customers.
The 3rd customer will see on average 0.6 customers when it arrives at device *j*.



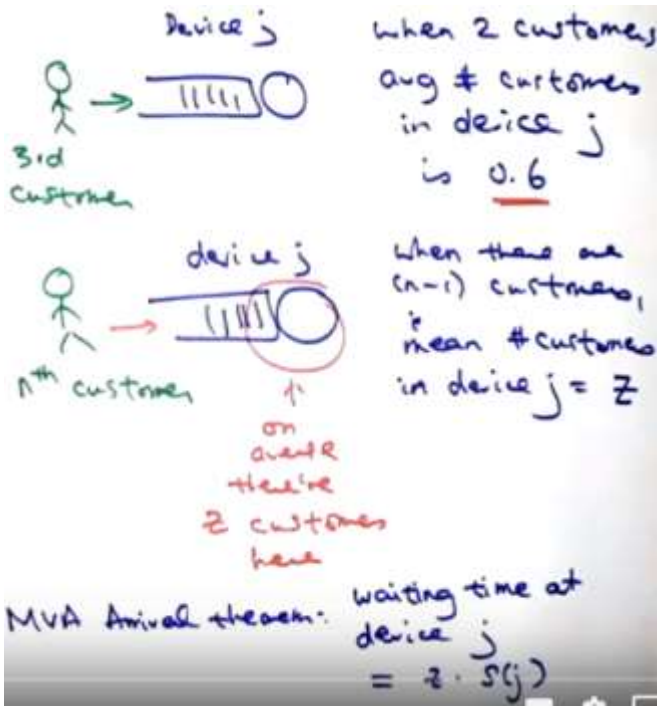
Arrival Theorem

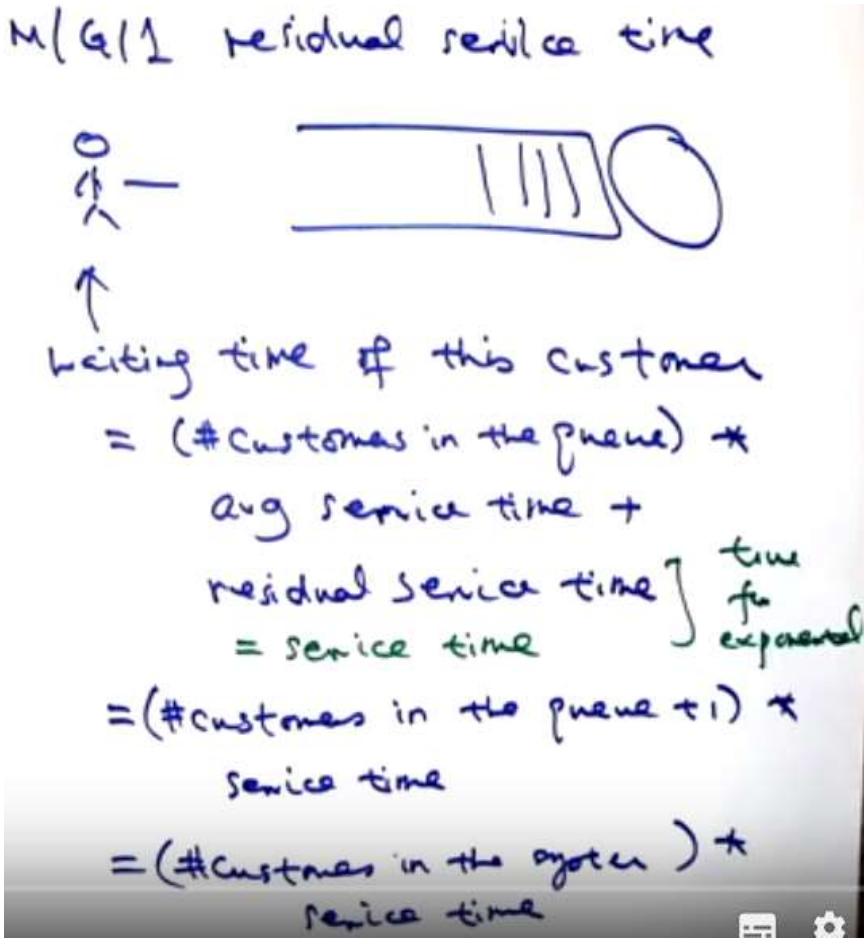
- If there are *(n-1)* customers in the system, the mean number of customers in device *j* is *z* customers,
- Then, when there are *n* customers, each customer arriving at device *j* will see on average *z* customers ahead of itself in device *j*.

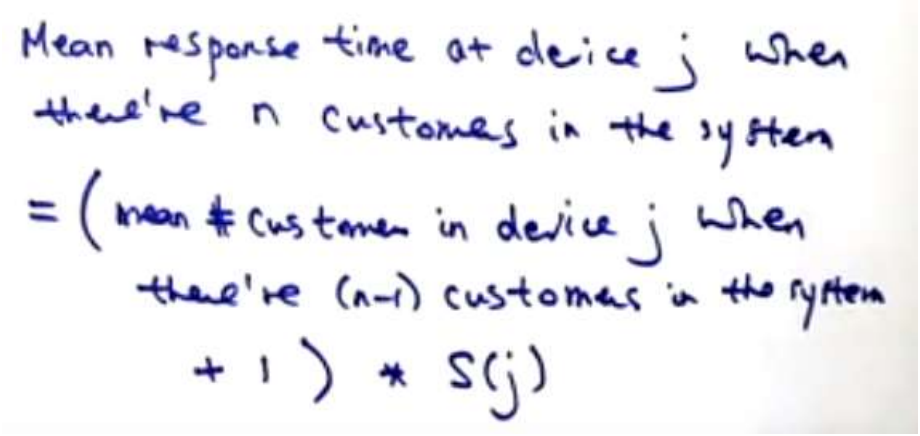
How can Arrival Theorem help?



Let *S(j)* = mean service time at device *j*.
When there are *n* customers,
The mean waiting time at device *j* = *z S(j)*
The mean response time at device *j* = *(z+1) S(j)*

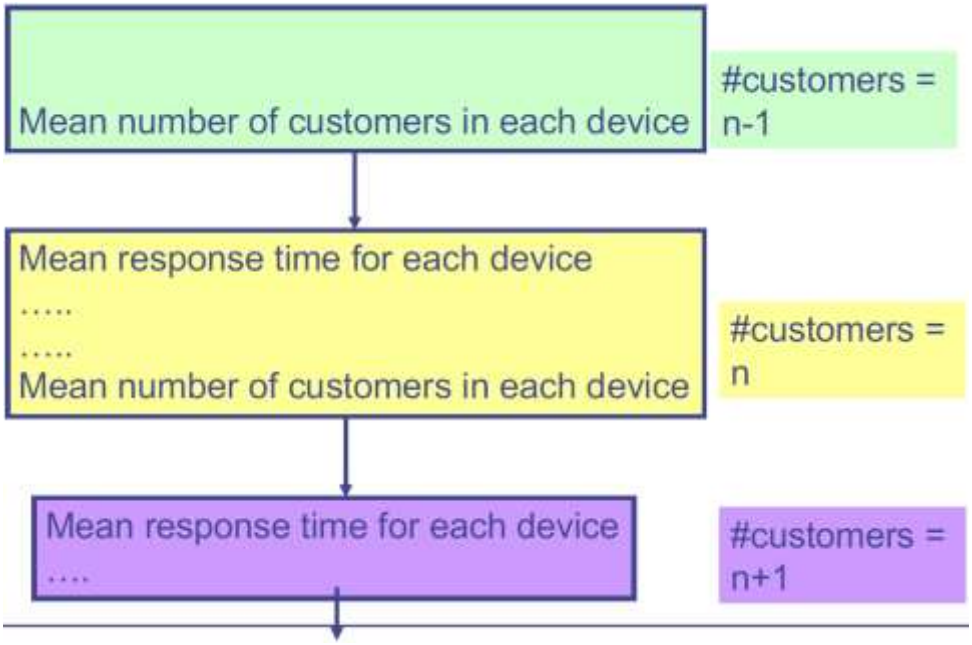

Device j
When 2 customers
avg # customers
in device j
is 0.6
3rd
customer
device j
When there are
(n-1) customers,
mean # customers
in device j = Z
nth customer
on
average
there're
Z customers
here
MVA Arrival theorem:
waiting time at
device j
= Z · S(j)


M/G/1 residual service time
Waiting time of this customer
= (# Customers in the queue) *
avg service time +
residual service time
= service time
time for exponential
= (# customers in the queue + 1) *
service time
= (# Customers in the system) *
service time



通过上面得公式可以通过 n-1 个 users 得出当 users 变成 n 得时候的 mean response time，下面是关于其他有关变量的推导

Iterations of MVA:



Note "(n)" means there are n customers in the system

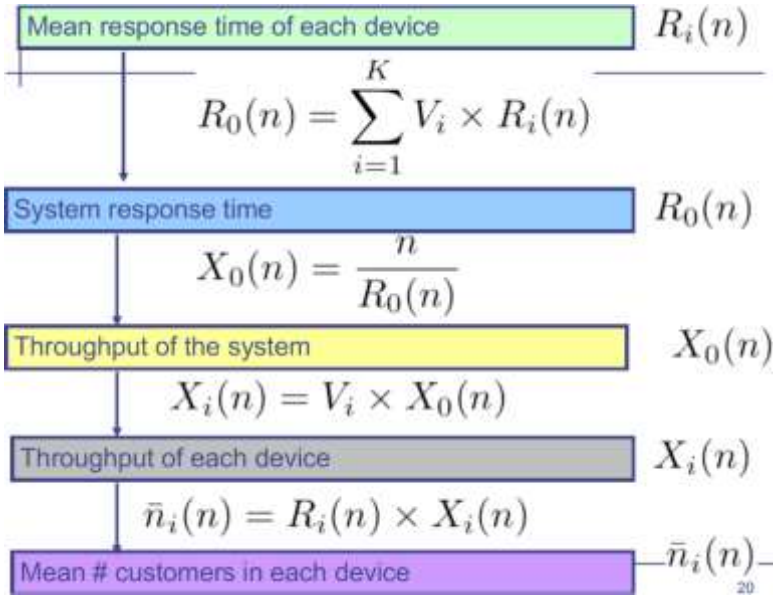
$\bar{n}_i(n) =$ Mean # of customers in device i

$R_i(n) =$ Mean response time in device i

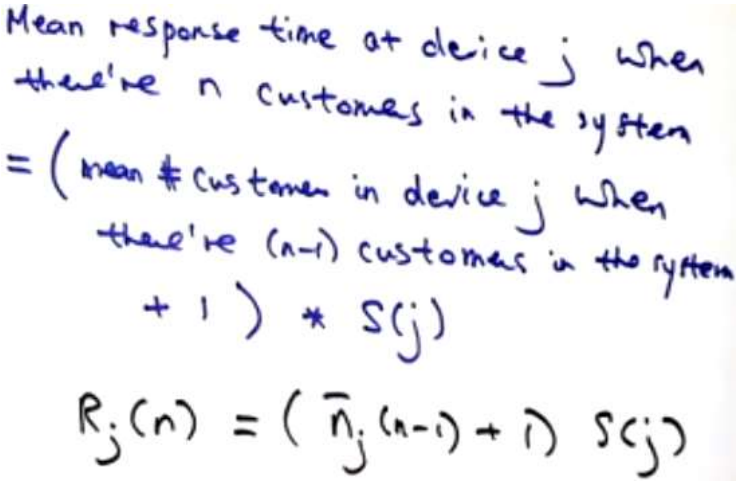
$R_0(n) =$ Mean response time of the system

$X_i(n) =$ Throughput of device i

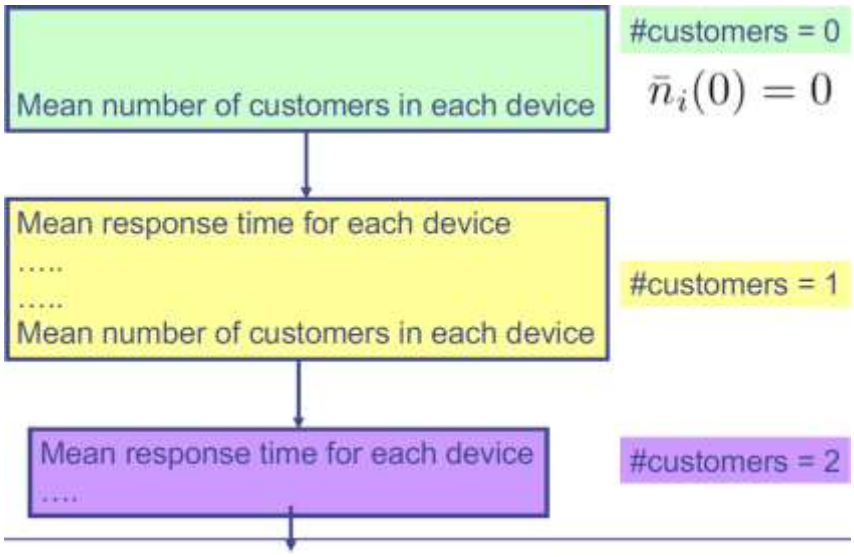
$X_0(n) =$ Throughput of the system



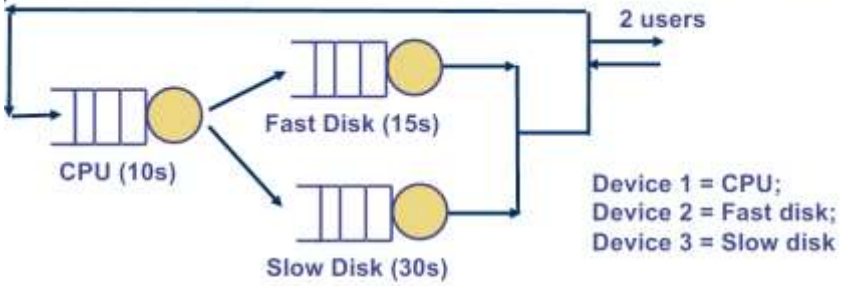
How to use above:



Initialization of MVA:



Let us apply MVA to the database server example



$$S_1 = 10; S_2 = 15; S_3 = 30;$$
$$V_1 = 1; V_2 = \frac{1}{2}; V_3 = \frac{1}{2};$$

- Determine the performance when there are 2 users in the system
- And how about 3 users?

Calculation steps:

$S_1 = 10,\ S_2 = 15,\ S_3 = 30,\ V_1 = 1,\ V_2 = \frac{1}{2} = V_3$

$n = 0 \quad \bar{n}_1(0) = \bar{n}_2(0) = \bar{n}_3(0) = 0$

$n = 1 \quad R_1(1) = (\bar{n}_1(0) + 1) S_1 = 10$

$R_2(1) = S_2 = 15$

$R_3(1) = S_3 = 30$

$R_0(1) = V_1 R_1(1) + V_2 R_2(1) + V_3 R_3(1)$

$= 1 \cdot 10 + \frac{1}{2} \cdot 15 + \frac{1}{2} \cdot 30$

$= 32.5$

$X_0(1) = \frac{1}{R_0(1)} = \frac{1}{32.5}$

$X_1(1) = V_1 X_0(1) = \frac{1}{32.5}$

$X_2(1) = X_3(1) = \frac{1}{2} \cdot \frac{1}{32.5} = \frac{1}{65}$

$\frac{20}{65} =$

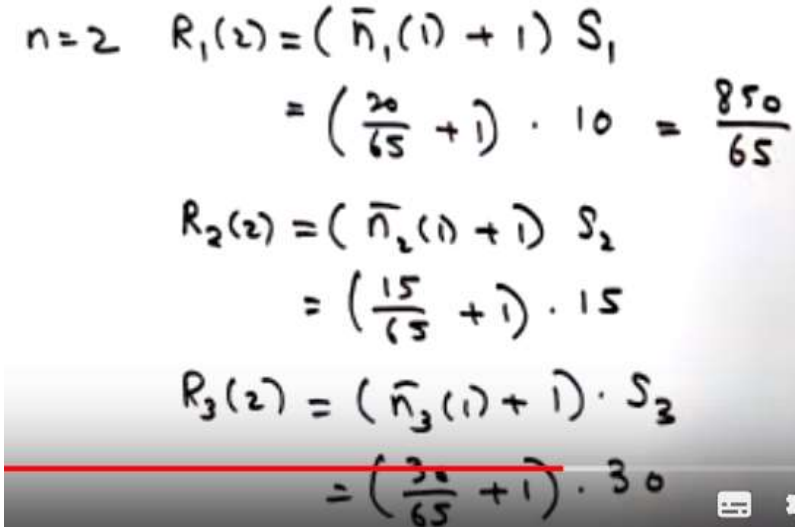
$\bar{n}_1(1) = R_1(1) X_1(1) = 10 \cdot \frac{1}{32.5} = \frac{10}{32.5}$

$\bar{n}_2(1) = R_2(1) X_2(1) = 15 \cdot \frac{1}{65} = \frac{15}{65}$

$\bar{n}_3(1) = R_3(1) X_3(1) = 30 \cdot \frac{1}{65} = \frac{20}{65}$

要想验证 n1, n2, n3 的值是否正确，只需要将其相加，即：$\bar{n}_1 + \bar{n}_2 + \bar{n}_3 = mean\#\ of\ users$即可，这里是判断其是否为一，即是：$\bar{n}_1(1) + \bar{n}_2(1) + \bar{n}_3(1) = 1$

以下为 users 为 2 得步骤：



Limitation of MVA

MVA allows you to find the mean value of throughput, response time etc.

However, if you are interested to find the probability that the system is in a certain state. MVA cannot give you the answer. You will need to resort to Markov model.

Extensions of MVA

Closed queueing networks with multiple classes of customers

- Example: Database servers with 2 classes of customers
  - One class of customers require mean service time of 0.02s, 0.03s and 0.05s from the CPU, fast and slow disk
  - Another class of customers require mean service time of 0.04s, 0.01s and 0.1s from the CPU, fast and slow disk

Open queueing networks

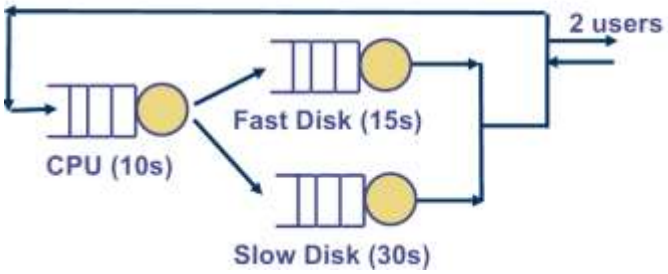
Mixed queueing networks

Assumptions behind MVA

The service time is exponentially distributed

The service time required at each component is independent

- For example, MVA assumes that the service time required at CPU is independent of the service time at the disk



Solution to network of queues

You have seen two possible methods to solve a network of queues

- Analytical solution
- Simulation

For closed queueing networks with exponentially distributed service time

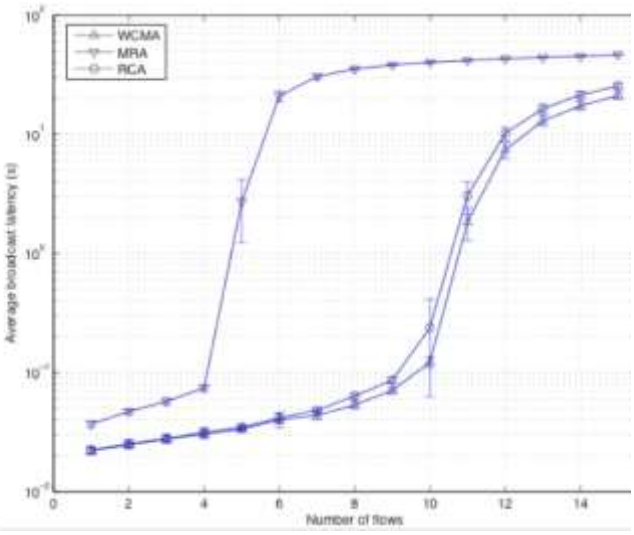
- Markov chain
- MVA

Commercial simulation tools can deal with hundred of nodes

Multicast in wireless mesh networks

In my research on designing multicast protocol for wireless mesh networks, we use simulation package *Qualnet* to investigate which of the multicast protocols that we have designed is better

The network has 400 wireless mesh routers (= 400 queues)



Analytical solution versus simulation

Analytical solution

- Limited to specific cases
  - E.g. Exponential assumptions
- Efficient computation algorithm exists for certain cases
  - MVA for closed queueing networks with exponential service time

Simulation

- Can apply to general settings
  - Difference classes of traffic, protocols etc.
- Can apply to reasonably large networks too

Week 5B: Revision problems

Question 1

Measurements were conducted on an interactive computer system and the following results were obtained:

Length of measurement interval:	1 hour
Number of completed requests:	36,000
Utilisation of CPU:	75%
Utilisation of Disk 1:	50%
Utilisation of Disk 2:	50%
Utilisation of Disk 3:	25%
Think time:	7s

You may assume that the service time is exponentially distributed.

(a) Compute the service demand on the CPU and the disks.

(b) Write a computer program to implement the MVA algorithm. The inputs to your computer program should be

- Service demand of the components
- The think time
- The maximum number of interactive users N

The output of the program should be

- System throughput when there are 1,...,N interactive users
- The system response time when there are 1,...,N interactive users

(c) Use your computer program to determine the system throughput when the number of interactive users varies from 1 to 200.

Use asymptotic analysis to determine the upper bound on system throughput when the number of interactive users from 1 to 200.

Plot both the actual throughout and the asymptotic bound on the same graph. What do you observe?

(d) Assuming that there are 70 interactive users in the system. By what factor must you speed up the CPU so that the system response time is 0.3s.

在交互式计算机系统上进行测量，得到以下结果：

测量间隔时间：1 小时

已完成请求数：36000

CPU 利用率：75%

磁盘 1 利用率:50%

磁盘 2 的利用率:50%

磁盘 3 利用率:25%

思考时间：7S

您可以假设服务时间是指数分布的。

（a）计算 CPU 和磁盘上的服务需求。

（b）编写计算机程序以实现 MVA 算法。计算机程序的输入应该是

1.组件的服务需求

2.思考时间

3.最大交互用户数 n

程序的输出应该是

1.有 1 个，…，n 个交互用户时的系统吞吐量

2.有 1 个，…，n 个交互用户时的系统响应时间

（c）当交互用户数从 1 到 200 时，使用计算机程序确定系统吞吐量。

利用渐近分析确定交互用户数从 1 到 200 时系统吞吐量的上限。

在同一个图上绘制实数和渐近界。你观察到什么？

（d）假设系统中有 70 个交互用户。您必须通过什么因素加速 CPU，使系统响应时间为 0.3 秒。

(a) The system throughput = Number of completed jobs / measurement time = 36,000/3,600 = 10 jobs per second.

Since service demand = utilisation of the device / system throughput.

The service demands for the CPU, Disk 1, Disk 2 and Disk 3 are respectively, 0.0750, 0.0500, 0.0500 and 0.0250 second.

(b) A sample program is given in *mva_sc.m*.

Note that the MVA algorithm given in the lecture assumes that the think time is zero. Think time can be taken into account by adding a pure delay element. if there is not think time, the system throughput is given by the number of users divided by the system response time (which is basically Little's Law). When there is think time, the system throughput should instead be given by the number of users divided by the sum of the system response time and think time (which is again Little's Law).

(c) The results are plotted in figure 1. We observe that the upper bound that we have derived in Week 2 is indeed correct. It is indeed an upper bound. Moreover, the upper bound seems to be quite accurate.

Remark: It is important to note that the upper bound from operational analysis was derived without making any assumption on the probability distribution of the service time. In other words, the bounds hold even if the probability distribution of service time is not exponential. However, for MVA to apply, the service time has to be exponentially distributed.

(d) If we speed up the CPU by f times, then the new service demand for the CPU will be $\frac{0.075}{f}$. We vary f from 1 to 2 and for each value of f, we compute the new response time using MVA. By plotting how the response time varies with f, we can find the required scaling factor.

The plot of response time is in Figure 2. From the figure, we see that the speed up factor required is about 1.65.

(b) 请注意，本课给出的 MVA 算法假定思考时间为零。

通过添加纯延迟元素，可以考虑思考时间。如果没有思考时间，系统吞吐量由用户数除以系统响应时间（基本上是小的定律）得出。当存在思考时间时，系统吞吐量应该由用户数除以系统响应时间和思考时间之和（这也是 Little 定律）给出。即如果在一个 interactive 系统中，有 think time，那么此时得 little's law 应该是：$X(0) = \frac{N}{R+Z}$,其中 z=think time.

(c) 需要注意的是，在不假设服务时间概率分布的情况下，从运行分析中得出上限。换句话说，即使服务时间的概率分布不是指数分布，边界也保持不变。然而，要应用 MVA，服务时间必须呈指数分布。

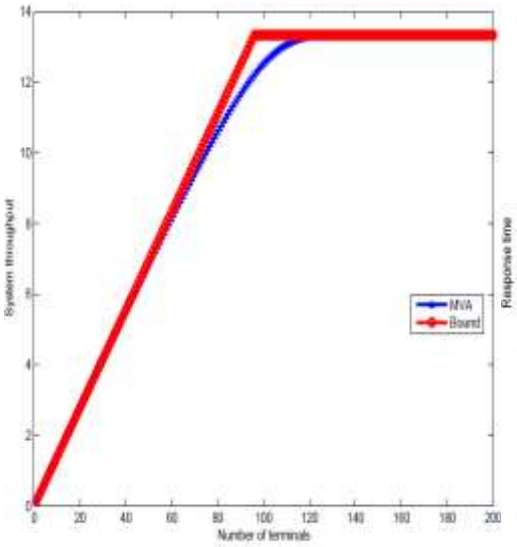


Figure 1: Throughput: MVA versus asymptotic bound.

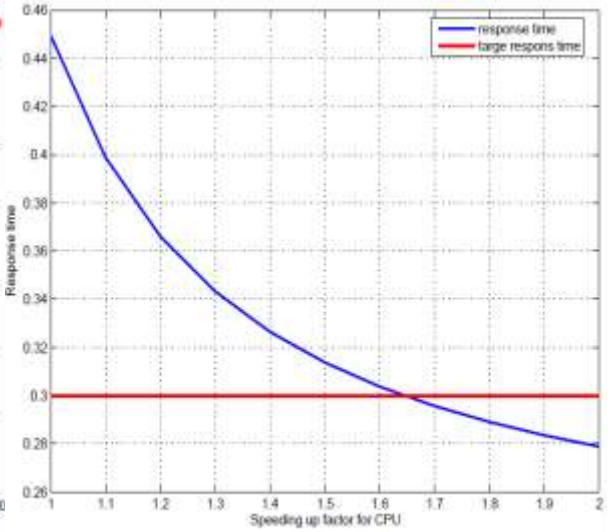


Figure 2: Response time.

Question 2

A web server has one CPU and one disk and was monitored during one hour. The utilisation of the CPU was measured at 30%. During this period, 10,800 HTTP requests were processed. Each request requires, on average, 3 I/Os on the server's disk. The average service time at the disk is 20 ms.

You may assume that the service time is exponentially distributed.

(a) What are the service demands of an HTTP request at the CPU and at the disk.

(b) Find the throughput, $X_0(n)$, of the web server for $n = 0, 1, 2$ and 3, where n is the number of concurrent HTTP requests in execution at the web server.

(c) Assuming that the web server receives requests at a rate of $\lambda = 5$ requests per second Poisson distributed. At most three HTTP requests can be in execution at any point in time. Requests that arrive and find 3 requests being processed will be placed in a processing queue, which is assumed to have an infinite size. Find the average response time of an HTTP request. This time includes the time spent by a request in the processing queue plus the time required to process the request. (Hint: Model the problem as a Markov chain.)

You may find the following formula useful:

$$p + m(p+q) + m^2(p+2q) + m^3(p+3q) + ... = \frac{p}{1-m} + \frac{mq}{(1-m)^2} \quad (1)$$

Web 服务器有一个 CPU 和一个磁盘，在一个小时内被监视。CPU 的利用率被测量为 30%。在此期间，处理了 10800 个 HTTP 请求。每个请求平均需要服务器磁盘上的 3 个 I/O。磁盘的平均服务时间为 20 毫秒。

您可以假设服务时间是指数分布的。

（A）在 CPU 和磁盘上 HTTP 请求的服务需求是什么？

（b）找到 n=0；1；2 和 3 时 Web 服务器的吞吐量 x0（n），其中 n 是在 Web 服务器上执行的并发 HTTP 请求数。

（c）假设 Web 服务器以每秒分布的 λ=5 个请求的速率接收请求。在任何时间点最多可以执行三个 HTTP 请求。到达并找到 3 个正在处理的请求的请求将放置在一个处理队列中，该队列假定具有无限大的大小。查找 HTTP 请求的平均响应时间。这个时间包括请求在处理队列中花费的时间加上处理请求所需的时间。

To find the service demands of an HTTP request at the CPU:

The throughput of the server is

$$= \frac{10800}{3600}$$

$$= 3 \text{ HTTP requests/s}$$

By service demand law

Service demand of CPU

$$= \frac{\text{utilisation of CPU}}{\text{Server throughput}}$$

$$= \frac{0.3}{3}$$

$$= 0.1\,s$$

D(Disk) = U(Disk)/X(0) = X(Disk)*S(Disk)/X(0) = V(Disk)*X(0)*S(Disk)/X(0)

(B)

To find the throughput:

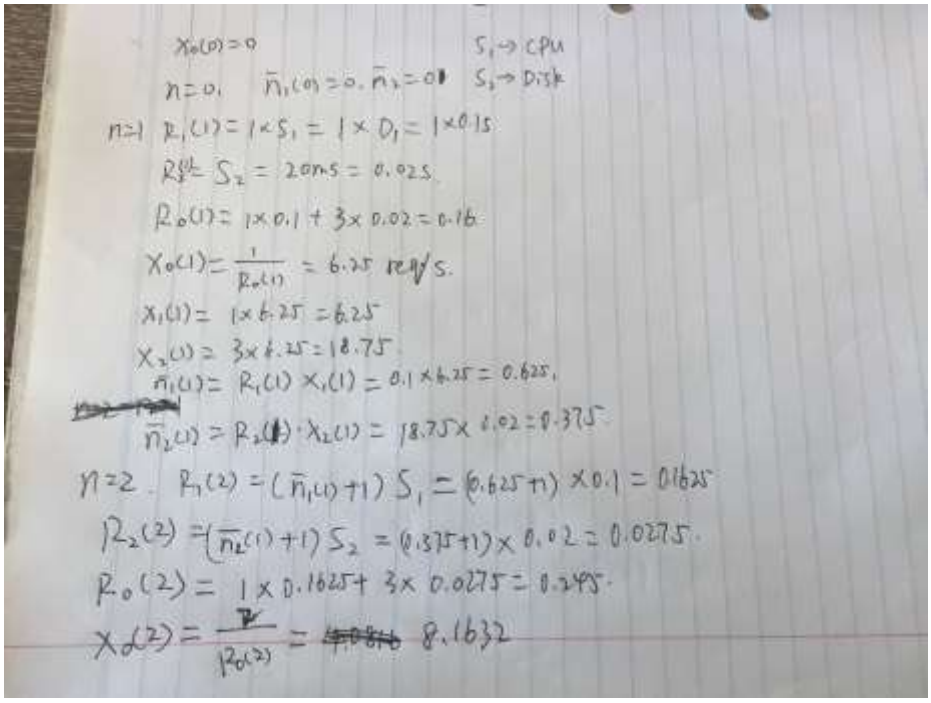
The easiest is to write a computer program and plug the values in.

$X(0) = 0$

$X_0(1) = 6.25$ requests/s

$X_0(2) = 8.16$ requests/s

$X_0(3) = 9.01$ requests/s



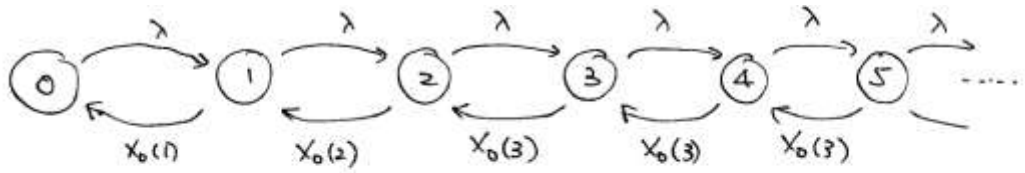
(C)

Part 3 of the quertin: To find the average response time of HTTP requests when $\lambda = 5$ and the Server can only process up to 3 requests at a time.

This can be modelled as a ~~generalised~~ Markov chain ~~birth-death model~~.

Let state k ($k = 0, 1, 2, \ldots$) be the number of requests in the web server. Note that the number of requests in the web server includes those that are being served (up to three) and those that are in the processing queue.

The state space diagram is in the following page:



* The transition rate from State k to State (k+1) (for k=0,1,...) is the arrival rate of the request.
* The transition rate from State (k+1) to State k (for k=0,...) is the rate at which requests are completed.
  - For State 1 to State 0, this is the same as the throughput of the web server when there is only one client. (Note that throughput is effectively the number of requests completed in an unit time.)
  - For State 2 to State 1, the request completion rate is $X_0(2)$.
- For State 3 to State 2, the request completion rate is $X_0(3)$
- For State (k+1) to State k (where $k \geq 3$), the request completion rate is always $X_0(3)$ because only 3 requests are being processed by the server. The others are waiting in the queue.

In order to find the response time, we need to solve the model. Using the trick given in the notes, we know that

$$P(1)\ X_0(1) = \lambda P(0)$$
$$P(2)\ X_0(2) = \lambda P(1)$$
$$P(3)\ X_0(3) = \lambda P(2)$$
$$P(4)\ X_0(3) = \lambda P(3)$$
$$P(5)\ X_0(3) = \lambda P(4)$$
$$\text{- - - -}$$

Expressing $P(1), P(2), \ldots$ in terms of $P(0)$, we have

$$P(1) = \frac{\lambda}{X_0(1)} P(0)$$

$$P(2) = \frac{\lambda}{X_0(2)} \frac{\lambda}{X_0(1)} P(0)$$

$$P(3) = \frac{\lambda}{X_0(3)} \frac{\lambda}{X_0(2)} \frac{\lambda}{X_0(1)} P(0)$$

$$P(4) = \left(\frac{\lambda}{X_0(3)}\right)^2 \frac{\lambda}{X_0(2)} \frac{\lambda}{X_0(1)} P(0)$$

$$P(5) = \left(\frac{\lambda}{X_0(3)}\right)^3 \frac{\lambda}{X_0(2)} \frac{\lambda}{X_0(1)} P(0)$$

observing the pattern, we've

$$P(k) = \left(\frac{\lambda}{X_0(3)}\right)^{k-2} \frac{\lambda}{X_0(2)} \frac{\lambda}{X_0(1)} P(0)$$

for $k \geq 3$

Define
$$\begin{cases} \rho_1 = \dfrac{\lambda}{X_0(1)} \\ \rho_2 = \dfrac{\lambda}{X_0(2)} \\ \rho_3 = \dfrac{\lambda}{X_0(3)} \end{cases}$$

we have

$$P(1) = \rho_1 P(0)$$

$$P(2) = \rho_2 \rho_1 P(0)$$

$$P(k) = \rho_3^{k-2} \rho_2 \rho_1 P(0) \quad \text{for } k \geq 3$$

Since the sum of all probabilities must be 1,

$$P(0) + P(1) + \dots + \dots = 1$$

~~$\rho_1 P(0)$~~

$$P(0) + \rho_1 P(0) + \rho_2 \rho_1 P(0) + \rho_3 \rho_2 \rho_1 P(0) + \rho_3^2 \rho_2 \rho_1 P(0) + \dots = 1$$

↓ (geometric progression) (Note: you can use the formula given in the question)

$$\Leftrightarrow P(0) + \rho_1 P(0) + \frac{\rho_2 \rho_1}{1 - \rho_3} P(0) = 1$$

(Note that $\rho_3 < 1$, so the geometric progression converges)

$$\Rightarrow P(0) = \frac{1}{1 + \rho_1 + \frac{\rho_2 \rho_1}{1 - \rho_3}}$$

In order to calculate the response time, we need to compute the throughput and ~~queue length first~~. the mean # requests in the server.

Throughput

$$= X_0(1)\,P(1) + X_0(2)\,P(2) + X_0(3)\left(P(3)+P(4)+\cdots\right)$$

$$= X_0(1)\cdot\rho_1\,P(0) + X_0(2)\cdot\rho_2\rho_1\,P(0) + X_0(3)\left(\rho_3\rho_2\rho_1 P(0) + \rho_3^2\rho_2\rho_1 P(0) + \cdots\right)$$

$$= X_0(1)\,\rho_1\,P(0) + X_0(2)\,\rho_2\rho_1\,P(0) + X_0(3)\,\rho_3\rho_2\rho_1 P(0)\cdot\frac{1}{1-\rho_3}$$

$$= \left(X_0(1)\,\rho_1 + X_0(2)\,\rho_2\rho_1 + X_0(3)\,\frac{\rho_3\rho_2\rho_1}{1-\rho_3}\right)\cdot\frac{1}{1+\rho_1+\frac{\rho_2\rho_1}{1-\rho_3}}$$

You can plug the values of $\rho_1, \rho_2, \rho_3, X_0(1), X_0(2)$ and $X_0(3)$ into the expression.

The mean # requests in the sewer is

$$0\,P(0) + 1\cdot P(1) + 2P(2) + 3\,P(3) + \cdots$$

$$= \rho_1 P(0) + 2\,\rho_2\rho_1 P(0) +$$

$$3\rho_3\rho_2\rho_1 P(0) + 4\,\rho_3^2\rho_2\rho_1 P(0) + 5\,\rho_3^3\rho_2\rho_1 P(0)$$

$$= \rho_1 P(0) +$$

$$\rho_2\rho_1 P(0)\left[\underbrace{2 + 3\rho_3 + 4\rho_3^2 + 5\rho_3^3 + \cdots}\right]$$

Need to recognise that this is an arithmetic-geometric progression. You can ~~look up~~ use the given formula ~~book~~ or you can derive the result. Let us derive the result:

Let $Z = 2 + 3\rho_3 + 4\rho_3^2 + 5\rho_3^3 + \cdots$ — ①

then $\rho_3 Z = 2\rho_3 + 3\rho_3^2 + 4\rho_3^3 + \cdots$ — ②

Substracting ② from ①, we have

$$(1-\rho_3)Z = 2 + \rho_3 + \rho_3^2 + \cdots$$

$$= 2 + \frac{\rho_3}{1-\rho_3} + \cancel{1}$$ ③

$$\Rightarrow \quad Z = \frac{\rho_3}{(1-\rho_3)^2} + \frac{2}{1-\rho_3}$$

(Alternatively, substitute $p=2$, $m=\rho_3$ & $\beta=1$ in the given formula)

Thus the mean # requests in the web server is

$$= \rho_1 P(0) + \rho_2 \rho_1 P(0) \cdot \left[\frac{\rho_3}{(1-\rho_3)^2} + \frac{2}{(1-\rho_3)}\right]$$

You can now plug the values of ρ_1, ρ_2, ρ_3 and $P(0)$ to find the mean # requests

Finally, to obtain the mean response time, we use Little's law

$$\text{Mean response time} = \frac{\text{mean \# requests in the server}}{\text{mean throughput}}$$

Week 6A: Discrete event simulation (2). Independent replications. Confidence interval.

Two topics

- How to general random numbers of a given probability distribution by using inverse transform method
- How to perform discrete event simulation of queues

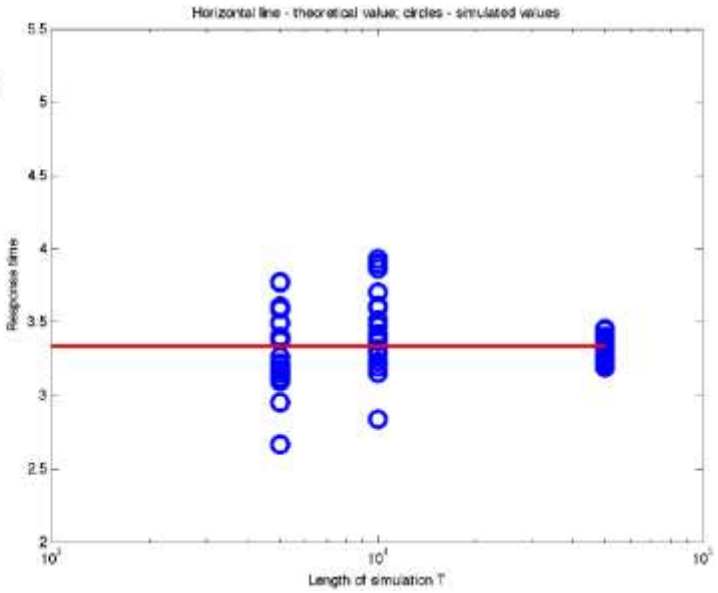
You should be able to simulate

- Many types of queues
  - Single-server or multi-server
  - Different queueing disciplines
- Many inter-arrival time and service time distributions

However, there are a number of problems …

Problem: data interpretation, simulation length

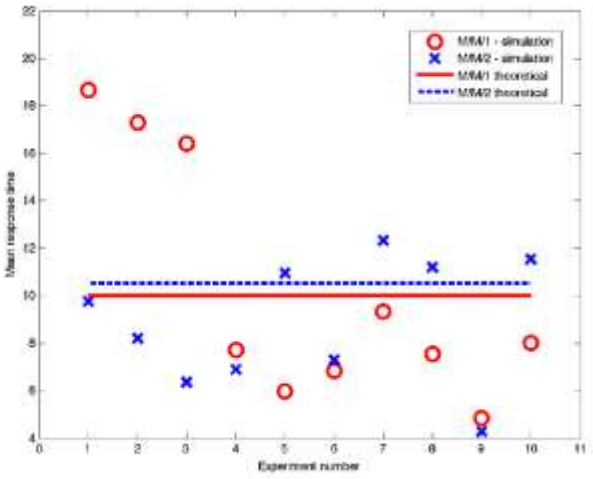
Figure from Week 5A's revision problem #1



Problem: How do we compare 2 alternative choices?

Week 5A's Revision Problem, #2

From Queueing theory, we expect the M/M/1 system to be better but simulation doesn't seem to suggest that?



Analysis of simulation results

A very important topic but it is very often ignored

Simulation is *not* simply about

- Writing a computer program
- Verifying the correctness of the program
- Running the simulation once and present the results

Verifying the correctness of the simulation program is important

It is equally important to do **sound statistical analysis** on the simulation results obtained

This lecture

Analysis of simulation results

How to choose simulation parameters?

- How long should I simulate for?
- How many times should I repeat the simulation?

Confidence interval

Analysis of simulation data

There are many statistical methods to analyse data depending on the situation

We will focus on analysing *steady state mean value* only

For example, we are interested to find the *steady state mean response time* of a queue

Recall that we talked about

- Transient and steady state behaviour of queue in Week 2B
- Steady state of Markov chain in Week 3B

What is steady state?

Let us simulate an M/M/1 queue with

- Arrival rate $\lambda = 0.7$
- Service rate $\mu = 1$
- Simulation ends when master clock is 50000s

In this simulation, we record the response time for each job

- Let X(k) = Response time of k^{th} job
- The next page shows X(k) changes continuously

Let N denote the number of jobs in the simulation

- N = 35000 for our simulation

In Week 5A, we computed the mean response time using

$$\frac{X(1)+X(2)+\ldots+X(N)}{N}$$

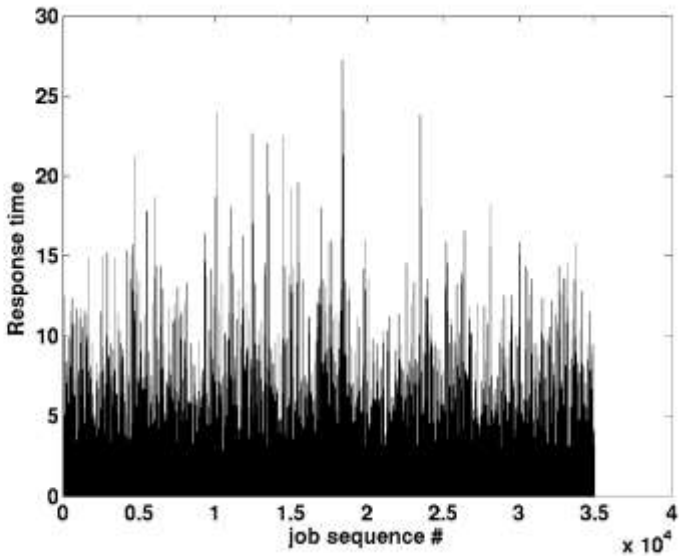
Response time continuously changes over time

This graph shows response time of X(k) of the *k*-th job where k = 1 to 35000

Note response time continuously varies

Response time *does not* settle to a constant value

But *mean* response time *does* settle



- Let us instead compute the running mean M(k) where

$$M(k) = \frac{X(1)+X(2)+...+X(k)}{k}$$

For example, if k = 5, then

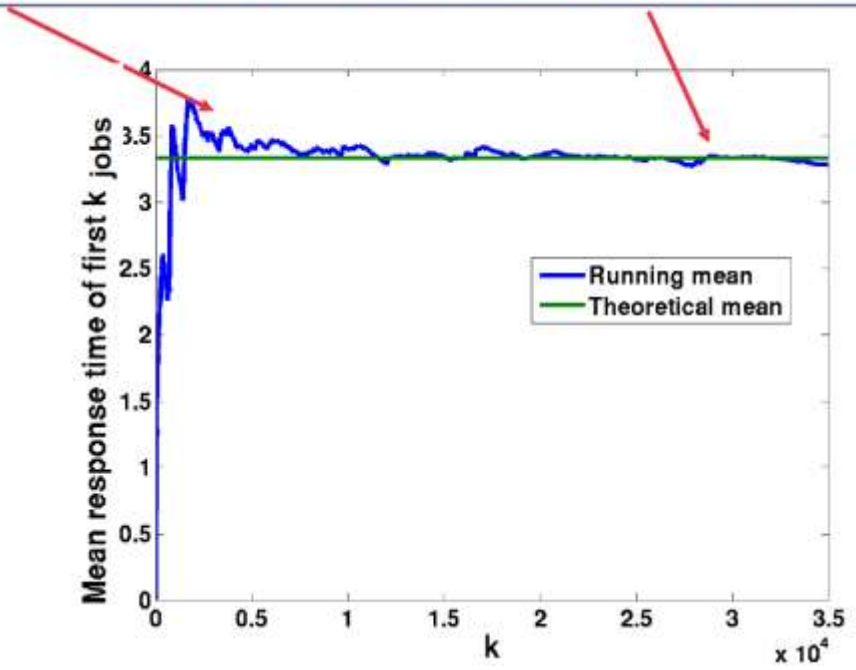
$$M(5) = \frac{X(1)+X(2)+...+X(5)}{5}$$

Thus M(5) is the mean response time of the first 5 jobs

In general, M(k) is the mean response time of the first *k* jobs

Let us plot M(k) - see the next slide

Transient behaviour versus steady state behaviour



Transient removal: Introduction

The early part of the simulation displays transient (= non-steady state behaviour)

The later part of the simulation converges or fluctuates around the steady state value

Since we are interested in the steady state value, we should not use the transient part of the data to compute the steady state value

We should remove the transient part and only use the steady state part to compute the mean

One method to identify the transient part is to use visual inspection

- Note: In the previous slide, we have the theoretical value to guide us but in practice you don't, you will learn a transient removal method based on batch means in Revision Problem 5B

- Let us assume that the first m jobs constitute the transient part and there are N jobs altogether, we should revise the formula to compute the mean to

$$\frac{X(m+1)+X(m+2)+...+X(N)}{N-m}$$

Note: We used too simple a method to compute the mean in Week 5A but I didn't want to complicate things at that time!

Important: You must run the simulation long enough so that you have a good number of data points (or jobs) in the steady state part.

Independent replications

Assume that we carry out simulations to find out what the steady state mean response time of a queueing system is

- Important note: *We cannot get exact answer from simulation*

- We express our simulation results as e.g. there is a probability of 95% that the mean response time is in the interval [3.1,3.3].
- We call the interval [3.1,3.3] the 95% *confidence interval*.

Independent replications: Repeat the simulation a number of times using ***different*** sets of random numbers

Why independent replications?

- Independent replications allow us to use statistical method to estimate a confidence interval of steady state mean response time

Example: Independent replications

We want to use simulation to estimate the mean response time of an M/M/1 queue with

- Arrival rate $\lambda = 0.7$
- Service rate $\mu = 1$
- Simulation ends when master clock is 16000s

We repeat the experiment 30 times using different sets of random numbers

For each independent experiments

- We record the response time of all the jobs
- Remove the transient part
- Compute the mean response time using the steady state section

We obtain 30 different estimates of the mean response time, one from each independent experiment

These independent estimates allow us to find a confidence interval

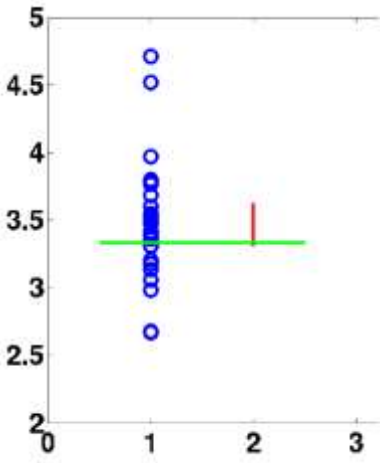
Example (Cont'd)

The blue circles show the estimated mean response time from the 30 independent experiments

The red line is the 95% confidence interval

- There is a 95% probability that the true mean response time that we want to estimate is in the interval [3.30,3.62]

The green line is the theoretical mean response time (which you should not normally know).



Computing the confidence interval

- Assume that you do *n* independent replications
- In each replication, you remove the transient part and compute an estimate of the mean steady state response time
  - Let us call your estimate from the kth replication, T(k)
- Compute the sample mean

$$\hat{T} = \frac{\sum_{i=1}^{n} T(i)}{n}$$

- And the sample standard deviation

$$\hat{S} = \sqrt{\frac{\sum_{i=1}^{n} (\hat{T} - T(i))^2}{n-1}}$$

Note: for sample standard deviation, *(n-1)* is in the denominator, *not n*.

- There is a probability (1-α) that the mean response time that you want to estimate lies in the interval

$$[\hat{T} - t_{n-1,1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}, \hat{T} + t_{n-1,1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}]$$

where $t_{n-1,1-\frac{\alpha}{2}}$ is the upper $(1 - \frac{\alpha}{2})$ critical point for the Student t distribution with $(n-1)$degrees of freedom

Note: $t_{[p;n]} = t_{n,p}$ in the lecture notes

$p = 1 - \frac{\alpha}{2}$

A.4 QUANTILES OF THE t DISTRIBUTION

Table A.4 lists $t_{[p;n]}$. For example, the $t_{[0.95;13]}$ required for a two-sided 90% confidence interval of the mean of a sample of 14 observation is 1.771.

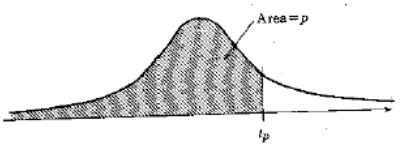


TABLE A.4 Quantiles of the t Distribution

	p							
n	0.6000	0.7000	0.8000	0.9000	0.9500	0.9750	0.9950	0.9995
1	0.325	0.727	1.377	3.078	6.314	12.706	63.657	636.619
2	0.289	0.617	1.061	1.886	2.920	4.303	9.925	31.599
3	0.277	0.584	0.978	1.638	2.353	3.182	5.841	12.924
4	0.271	0.569	0.941	1.533	2.132	2.776	4.604	8.610
5	0.267	0.559	0.920	1.476	2.015	2.571	4.032	6.869
6	0.265	0.553	0.906	1.440	1.943	2.447	3.707	5.959
7	0.263	0.549	0.896	1.415	1.895	2.365	3.499	5.408
8	0.262	0.546	0.889	1.397	1.860	2.306	3.355	5.041
9	0.261	0.543	0.883	1.383	1.833	2.262	3.250	4.781
10	0.260	0.542	0.879	1.372	1.812	2.228	3.169	4.587
11	0.260	0.540	0.876	1.363	1.796	2.201	3.106	4.437
12	0.259	0.539	0.873	1.356	1.782	2.179	3.055	4.318
13	0.259	0.538	0.870	1.350	1.771	2.160	3.012	4.221
14	0.258	0.537	0.868	1.345	1.761	2.145	2.977	4.140
15	0.258	0.536	0.866	1.341	1.753	2.131	2.947	4.073
16	0.258	0.535	0.865	1.337	1.746	2.120	2.921	4.015
17	0.257	0.534	0.863	1.333	1.740	2.110	2.898	3.965
18	0.257	0.534	0.862	1.330	1.734	2.101	2.878	3.922
19	0.257	0.533	0.861	1.328	1.729	2.093	2.861	3.883
20	0.257	0.533	0.860	1.325	1.725	2.086	2.845	3.850
21	0.257	0.532	0.859	1.323	1.721	2.080	2.831	3.819
22	0.256	0.532	0.858	1.321	1.717	2.074	2.819	3.792
23	0.256	0.532	0.858	1.319	1.714	2.069	2.807	3.768
24	0.256	0.531	0.857	1.318	1.711	2.064	2.797	3.745
25	0.256	0.531	0.856	1.316	1.708	2.060	2.787	3.725
26	0.256	0.531	0.856	1.315	1.706	2.056	2.779	3.707
27	0.256	0.531	0.855	1.314	1.703	2.052	2.771	3.690
28	0.256	0.530	0.855	1.313	1.701	2.048	2.763	3.674
29	0.256	0.530	0.854	1.311	1.699	2.045	2.756	3.659
30	0.256	0.530	0.854	1.310	1.697	2.042	2.750	3.646
60	0.254	0.527	0.848	1.296	1.671	2.000	2.660	3.460
90	0.254	0.526	0.846	1.291	1.662	1.987	2.632	3.402
120	0.254	0.526	0.845	1.289	1.658	1.980	2.617	3.373

The value $t_{n-1,1-\frac{\alpha}{2}}$ can be obtained from looking up

the Student t distribution table

- Note: A Student t table has been provided on the web site

There are also programs that compute it

- In Matlab, you can use tinv(1-alpha/2,n-1)

Example: Independent replications (Cont'd)
From the example above:

The sample mean of (n =) 30 replications = 3.47
The sample standard deviation of 30 replications is 0.43
If we want to compute the 95% confidence interval, α = 0.05
Since we did 30 independent experiments and want 95% confidence interval, we use $t_{29,0.975}$

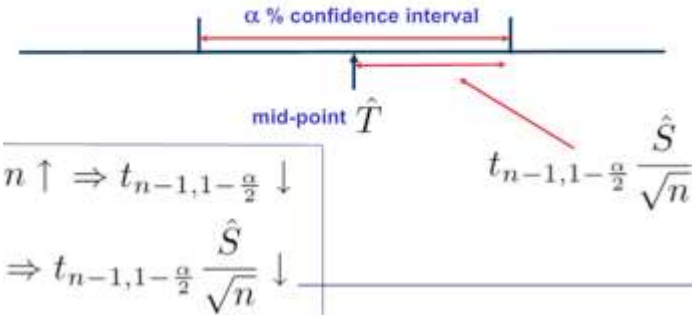
From the t-distribution table, the value of $t_{29,0.975}$ is 2.0452, the 95% confidence interval is

$$[3.47 - 2.0452\frac{0.43}{\sqrt{30}}, 3.47 + 2.0452\frac{0.43}{\sqrt{30}}] = [3.30, 3.62]$$

More on confidence interval

- Confidence interval

$$[\hat{T} - t_{n-1,1-\frac{\alpha}{2}}\frac{\hat{S}}{\sqrt{n}}, \hat{T} + t_{n-1,1-\frac{\alpha}{2}}\frac{\hat{S}}{\sqrt{n}}]$$



What can we get from simulation?

If your queueing problem has a mathematical solution, you will get *one* value for the steady state mean response time

If you simulate a queue to try to estimate the mean response time, you will *not* know the exact value of the steady state mean response time

Simulation can only give you a confidence interval of what you want to estimate

You can reduce the confidence interval by doing many independent replications!

Choice of simulation parameters

Simulation parameters

- Length of simulation
- Number of replications
- Accuracy

Unfortunately, there are *no* hard rules to choose them. You will need to do some trial and error

- If the length of simulation is not long enough, you will need to increase it
- If the number of replications is not enough to give you the desired accuracy, you will need to increase it

Length of simulation

- Must be longer than the transient
- Should have a good number of data point in the steady state part
  - Hard to say what "good" is. Get a few hundred if you can. The more the better but of course your simulation will run longer

Number of replications

- You may want to have 5 replications to start with
- After removing the transient, compute the confidence interval for your estimate.
- Compare the width of your confidence interval with your desired accuracy. If the confidence interval that you have obtained is too wide, you will need to increase the number of replications.
- Progressively (basically by trial-and-error), increase the number of replications until you get the desired level of accuracy

Summary

Simulation is not just a computer programming exercise

You need to make sure that your program is correct

It is also important to analyse your results statistically

Methods discussed include

- Transient removal technique
- Confidence interval
- Determining number of replications

Week 6A: Revision problems

We conducted five independent replications of the discrete event simulation of a queueing system and recorded the response time of the first 20,000 jobs in each replication. You can find these simulation results in 5 separate files `trace*` where * = 1,2, ..., 5.

(a) Program the transient removal procedure in Law and Kelton, Section 9.5.1 and find what the value of w should be. (A scanned copy of Section 9.5.1 can be found on the course web site.)

(b) After removing the transient, compute the steady state mean response time obtained from these independent replications. Calculate also the 90% confidence interval.

(a) A program that implements the transient removal procedure in Section 9.5.1 Law and Kelton can be found in `week06A_q1_a.m` (matlab file). You can vary the value of w. You should adjust w until you get a smoothed curve. There are no hard rules are to how to choose w. This is done by trial-and-error.

We start with $w = 10$. The result is plotted in the Figure 1. You see a lot of oscillation in the graph so we will need to increase the value of w to smooth it out.

Let us try $w = 100$. The result is plotted in Figure 2. The graph is still oscillatory but less.

Let us try $w = 500$. The result is plotted in Figure 3. The graph is a smoother but it still oscillates. It is difficult to get the ideal shape where the graph rises up initially and then settles down to a steady state value. From Figure 3, you can see that the curve oscillates around a value of 3.3. That is probably the mean value. Based on that, the suggestion is to cut away the first 1000 points.

(b) Since we have decided to take the first 1000 data points as the transient. For each replication, we compute the mean over 19000 data points (i.e. from data point 1001 to 20,000). The mean response times given by the 5 replications are: 3.488, 3.3309, 3.2025, 3.3242 and 3.2356.

The sample mean and sample standard deviation are calculated to be, respectively, 3.3163 and 0.1110. Since there are 5 replications, to compute the 90% confidence interval, we need to the value of $t_{4,0.95}$ which is 2.132. The 90% confidence interval is therefore $3.3163 \pm 2.132\frac{0.1110}{\sqrt{5}} = [3.2105, 3.4221]$.

Some of these calculations can be found in the file `week06A_q1_b.m`

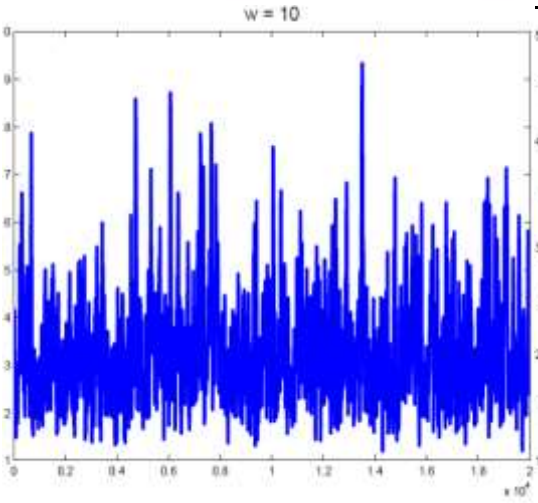


Figure 1: Question 1a. $w = 10$

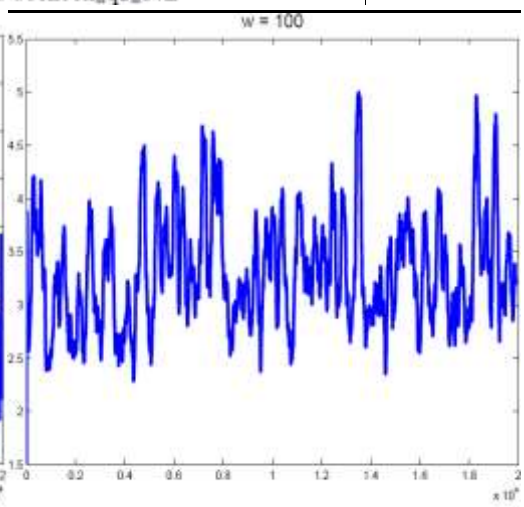


Figure 2: Question 1a. $w = 100$

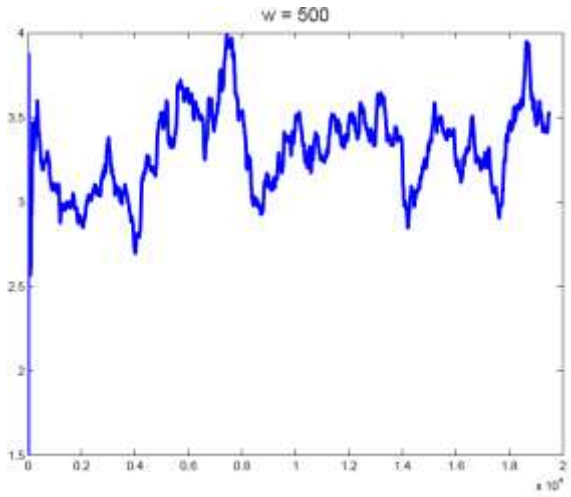


Figure 3: Question 1a. $w = 500$

Question 2

You performed 6 independent replications for a simulation. The mean response times from the 6 replications are: 4.56, 5.23, 5.12, 6.15, 5.57 and 5.34. Answer the following questions.

(a) Determine the 90% confidence interval.

(b) Determine the 95% confidence interval.

(c) Which confidence interval is wider? Why?

您为一个模拟执行了 6 个独立的复制。6 次重复的平均响应时间分别为：4.56、5.23、5.12、6.15、5.57 和 5.34。回答以下问题。

（a）确定 90%的压缩间隔。

（b）确定 95%的压缩间隔。

（c）哪个压缩间隔更宽？为什么？

The computation for Parts (a) and (b) can be found in the Matlab script `week06A_q2.m`.

(a) The 95% confidence interval is $[4.7776, 5.8791]$

(b) The 90% confidence interval is $[4.8966, 5.7601]$

(c) The 95% confidence interval has a wider range. Note that the data that are available for computing both 90% and 95% confidence intervals are the same. With a given amount of information, if you wish to make a prediction which is correct with a higher probability, e.g. 95% confidence rather than 90% confidence, then you need to use an interval which is wider. Let us use an analogy to explain this. Let us say that you want to guess the number that may come out from a throw of a fair die. If you want to get the number correct with probability $\frac{2}{3}$, you may say the number is in the interval $[1, 4]$. If you want to get the number correct with probability $\frac{1}{3}$, you may say the number is in the interval $[1, 2]$.

As a remark, if your goal in simulation is to have a more precise estimate of the mean response time. What you want to do is to make the confidence interval narrow enough. This is achieved by either simulating longer or using a greater number of independent replications.

Week 6B: Discrete event simulation (3). Comparing two systems.

You have learnt:

- How to write simulation program
- You know you cannot get exact mean response time from simulation but you can get a confidence interval
- You can reduce the width of the confidence interval by
  - Simulate for longer
  - Increase the number of independent replications

Today, you will learn how you can compare two systems in a statistically sound way

Before that, we show you that comparing systems can be tricky

Comparing two systems: motivation

An application of simulation is to compare two systems

For example, in Week 5A's revision question, you used simulation to compare the mean response time of

- System 1: M/M/1 queue with $\lambda = 0.9$ and $\mu = 1$
- System 2: M/M/2 queue with $\lambda = 0.9$ and $\mu = 0.5$ for both server

If you use analytical method, you can find the steady state mean response time of both systems exactly and you compare two numbers

If you use simulation, you get a confidence interval for each system instead. How do you compare them?

Let us assume our goal is to use simulation to compare:

- System 1: M/M/1 queue with $\lambda = 0.9$ and $\mu = 1$
- System 2: M/M/2 queue with $\lambda = 0.9$ and $\mu = 0.5$ for both server

For each system we carry out 3 independent replications

- That is, we use 6 sets of independent random numbers together

After removing the transient, the estimated mean response times are:

- System 1: 6.8769, 8.5769, 10.6340
- System 2: 8.8087, 7.4616, 9.1565

In order to compare them, let us pair up these results

- 1st experiment for System 1 with 1st experiment for System 2
- 2nd experiment for System 1 with 2nd experiment for System 2 etc.

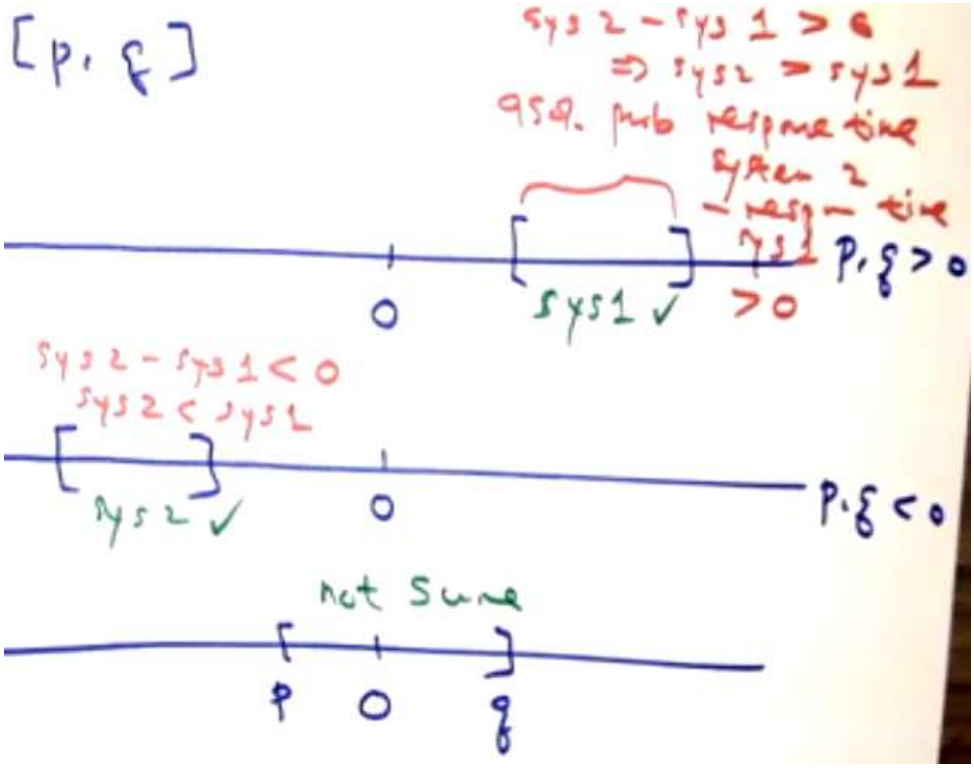
A paired-t confidence interval

- Let us summarise the data in a table
  - EMRT = estimated mean response time

	EMRT System 1	EMRT System 2	EMRT System 2 - EMRT System 1
Rep. 1	6.8769	8.8087	1.9318
Rep. 2	8.5769	7.4616	-1.1154
Rep. 3	10.6340	9.1565	-1.4775

- We compute the 100 $(1-\alpha)$% confidence interval of the difference between 2 systems (= last column)

- Let us denote the computed confidence interval by [p,q]
  - Case 1: $p,q > 0$ ➔ System 1 is better than System 2 with probability $(1-\alpha)$
  - Case 2: $p,q < 0$ ➔ System 2 is better than System 1 with probability $(1-\alpha)$
  - Case 3: $q > 0$ & $p < 0$ ➔ Systems 1 and 2 are not different with probability $(1-\alpha)$



Example: Paired-t confidence interval

We compute the 95% confidence interval of the data showed in the last slide, the confidence interval is:

- [-4.8721,4.4314]

Therefore, with 95% probability that the mean response times of the two systems are not different

Hmmm, we have a problem here, we know from queueing theory that System 1 has a better mean response time than System 2, but our simulation does not seem to be able to distinguish them.

What can we do?

Let us increase the number of replications

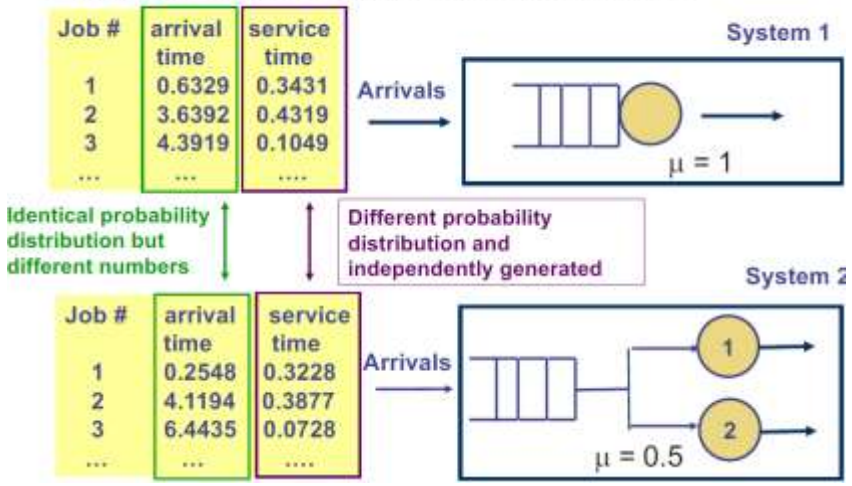
- Since increasing the number of replications can reduce the width of the confidence interval, let us try that.
- Let us try 5, 10, 20, 30 replications

# independent replications	95% Confidence interval of EMRT System 2 - EMRT System 1
5	[-4.9540, 5.0242]
10	[-1.5347, 2.8020]
20	[-1.2724, 1.9870]
30	[-0.6001, 1.8046]

- Increasing the number of replications does reduce the width of the confidence interval
- However, we still cannot conclude which system is better

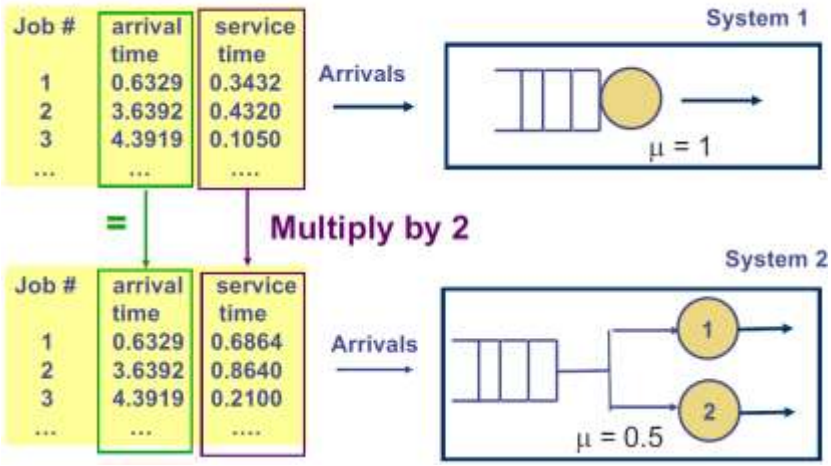
Let us have a look at how we did our experiments …

- We did our experiment with independent random numbers



Common random numbers method

- An alternative is to compare two systems under similar condition



Common random numbers method

A method to reduce the variance when comparing two alternative systems is to subject them to similar experimental condition

In each replication, generate only one arrival time and one service time sequence

- Apply this to both systems
- Note: Service time may need to be adjusted according to service rate

In next replication, generate a new arrival time and a new service time sequence

- Apply this to both systems

This method can reduce the variance if the behaviour of the two systems is positively correlated

Applying common random numbers to our problem

Let us apply the common random numbers method to compare

- System 1: M/M/1 queue with $\lambda = 0.9$ and $\mu = 1$
- System 2: M/M/2 queue with $\lambda = 0.9$ and $\mu = 0.5$ for both server

Let us carry out 5 replications

In each replication, we generate one arrival time sequence and one service time sequence (adjusted by service rate) and apply to both systems

- Let us compare the estimated mean response time (EMRT) from the 5 replications:

	EMRT System 1	EMRT System 2	EMRT System 2 - EMRT System 1
Rep. 1	8.3022	8.8087	0.5065
Rep. 2	6.8809	7.4616	0.5807
Rep. 3	8.5769	9.1565	0.5796
Rep. 4	10.6340	11.3409	0.7069
Rep. 5	16.2648	16.6485	0.3837

- Observation: The EMRT of System 2 is higher than that of System 1 in all 5 replications
- If we compute the 95% confidence interval of the last column, we get [0.4046,0.6983]
- There is a 95% probability that System 1 is better than System 2

Comparing two methods

- Let us compare using common random number (CRN) method or not

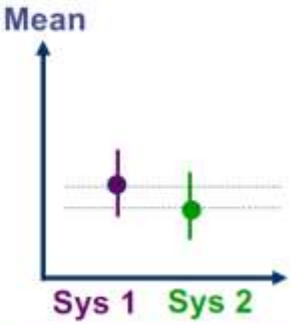
# independent replications	95% Confidence interval of EMRT System 2 - EMRT System 1	
	Not using CNR	Using CNR
5	[-4.9540, 5.0242]	[0.4046, 0.6983]
10	[-1.5347, 2.8020]	[0.4705, 0.6103]
20	[-1.2724, 1.9870]	[0.5127, 0.5942]
30	[-0.6001, 1.8046]	[0.5026, 0.5786]

- Observations
  - By using CRN, all 95% confidence interval does not include 0
  - The width of the confidence interval for CRN method is a lot lower!

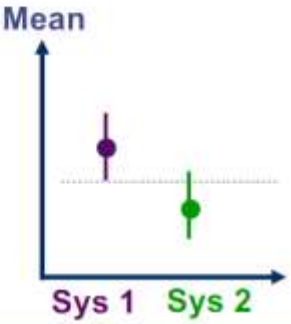
Approximate visual test

- Let us assume that you know the mean response time and its confidence interval (CI) for 2 systems: **System 1** and **System 2**
- Consider the following 3 possibilities:

CIs do not overlap Mean of System 1 > Mean of Sys. 2



CIs overlap and mean of a system is in the CI of the other: System are *not* different



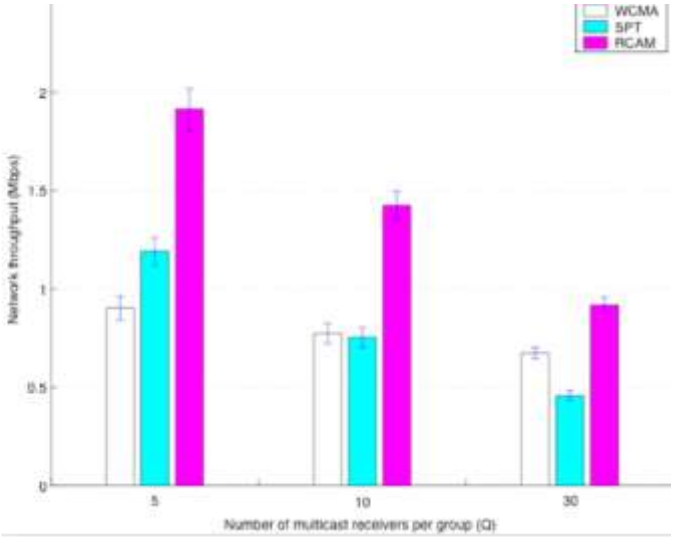
CIs overlap and mean of any one is not in the CI of the other: do *t*-test

Multicast protocol design for wireless mesh networks

Comparing 3 multicast protocols (WCMA, SPT and RCAM) for wireless mesh networks

The thin vertical line shows the confidence interval

What conclusion can you draw?



Simulation tools and some applications

You do not always have to write your own simulation programs from scratch

There are plenty of simulation tools available

- Many with GUI

Simulation tools are used in a lot in computer networking research

- Protocol #1 is the existing protocol, you have designed Protocol #2. You want to see whether Protocol #2 is better or not.

- You have two options (Option #1 and Option #2) to design a network. Which option is better?

Summary

Simulation is not just a computer programming exercise

You need to make sure that your program is correct

It is also important to analyse your results statistically

Unfortunately, a lot of published research papers in computer networking did not do sound statistical analysis

Week 6B: Revision problems

Three systems (Systems 1, 2 and 3) are tested. The mean response time of each system is measured 5 times. The results for the three systems are summarised below:

System 1	System 2	System 3
13.64	12.78	12.21
13.09	13.98	13.64
13.84	13.58	13.09
12.28	14.59	13.84
14.55	12.72	12.28

Can you conclude which system has the best performance with high confidence (say 95%)?

Let us first compare Systems 1 and 2. The mean response time of System 1 minus mean response time of System 2, over 5 replications, are 0.86, -0.89, 0.26, -2.31, 1.83. The sample mean and sample standard deviation are, respectively, -0.05 and 1.6025. The 95% confidence interval is $-0.05 \pm t_{4,0.0975}\frac{1.6025}{\sqrt{5}} = [-2.0397, 1.9397]$. Thus we can not determine whether System 1 is better or worse than System 2. (Note: $t_{4,0.0975} = 2.7764$)

Consider the difference: mean response time of System 1 minus mean response time of System 3, we find the 95% confidence interval for this difference is $[-1.4356, 2.3716]$, thus we cannot conclude whether System 1 is better or worst than System 3.

Consider the difference: mean response time of System 2 minus mean response time of System 3, we find the 95% confidence interval for this difference is $[0.3266, 0.7094]$, thus we can conclude System 3 is better than System 2 with 95% confidence.

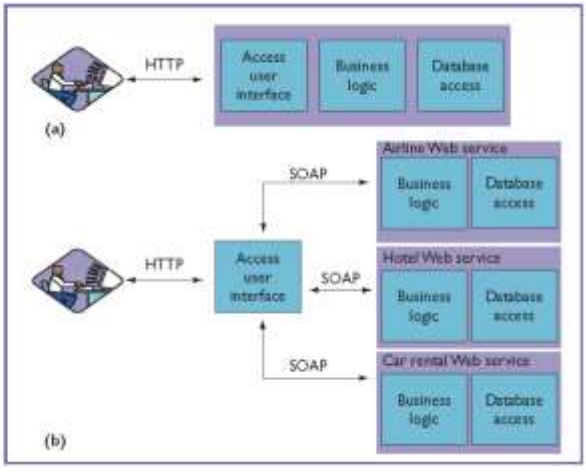
We can only conclude that System 3 is better than System 2 with 95% confidence. However, we cannot say with 95% confidence which system is the best out of the three.

Week 7A: Web services and fork-join queues

- Web services
  - What is it?
  - Performance analysis
- Fork-join queue
  - Markov chain
  - MVA

(a) Web access

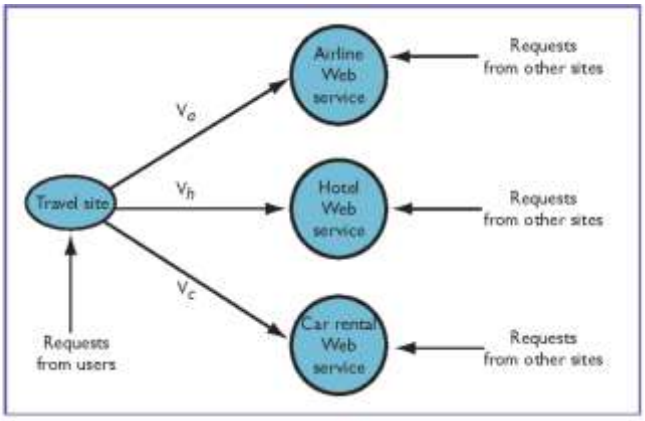
(b) Composite web service for travel



Web service performance issues

- Metrics
  - Response time
  - Throughput
  - Availability
- Performance analysis method
  - Operational analysis
  - Markov chain

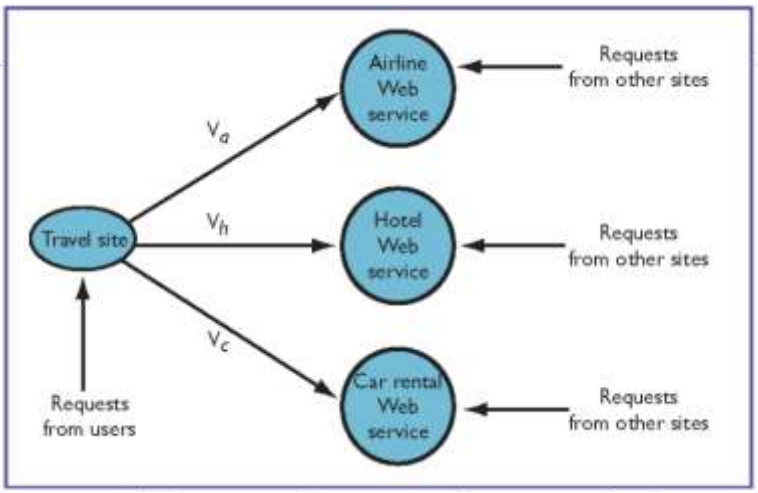
Web service flow graph



V_a, V_h, V_c: Relative Visit Ratio

Figure 2. Web service flow graph. Arrows link the travel site to other Web services. The labels on the links indicate the average number of times a Web service is invoked per request to the travel site.

Every request to the travel site generates on average V_a requests to the Airline web service etc.



$$X_{TA} = \text{Throughput of travel site}$$

$$X_a = \text{Throughput of airline Web service}$$

$$X_a \geq V_a \times X_{TA}$$

$$X_a \geq V_a \times X_{TA}$$
$$X_h \geq V_h \times X_{TA}$$
$$X_c \geq V_c \times X_{TA}$$

Xh = Throughput of hotel web service
Xc = Throughput of car rental web service

Can you find an upper bound on the throughput of the travel site

$$X_{TA} \leq \min\{\frac{X_a}{V_a}, \frac{X_h}{V_h}, \frac{X_c}{V_c}\}$$

Xa = 20 requests/s
Xh = 15 requests/s
Xc = 10 requests/s
Va = 4, Vh = 2, Vc = 1

$$X_{TA} \leq \min\{\frac{20}{4}, \frac{15}{2}, \frac{10}{1}\} = 5 \text{ requests/s}$$

The airline web service is the bottleneck of the travel web site.

More complex web service graphs

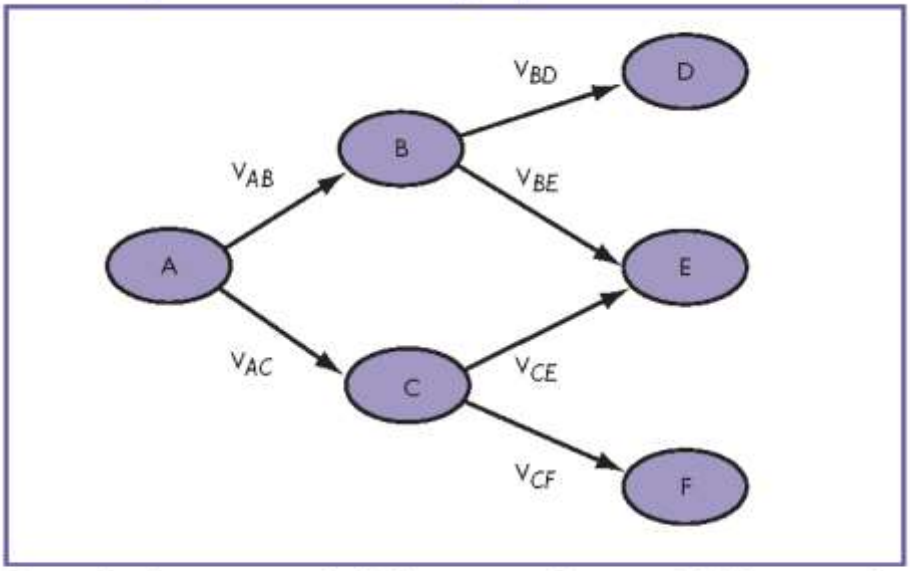
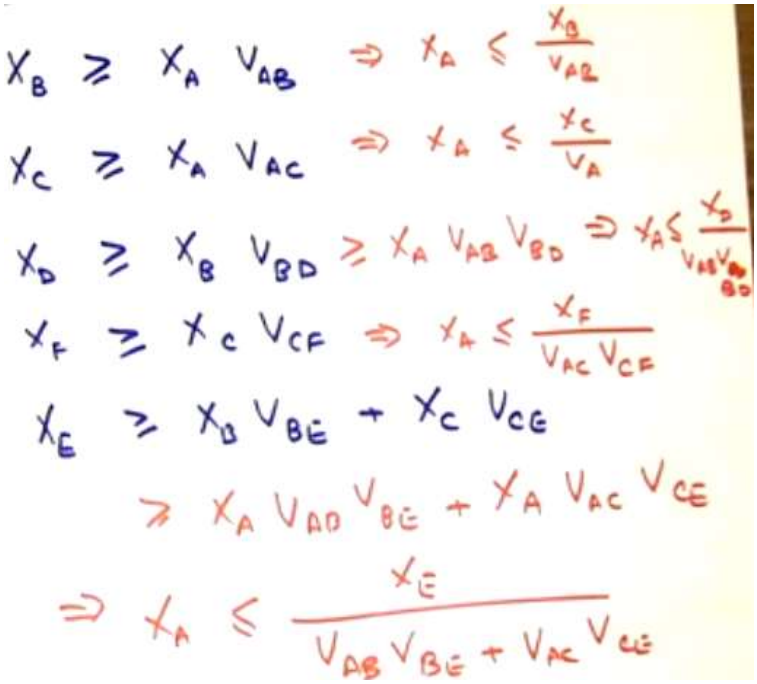


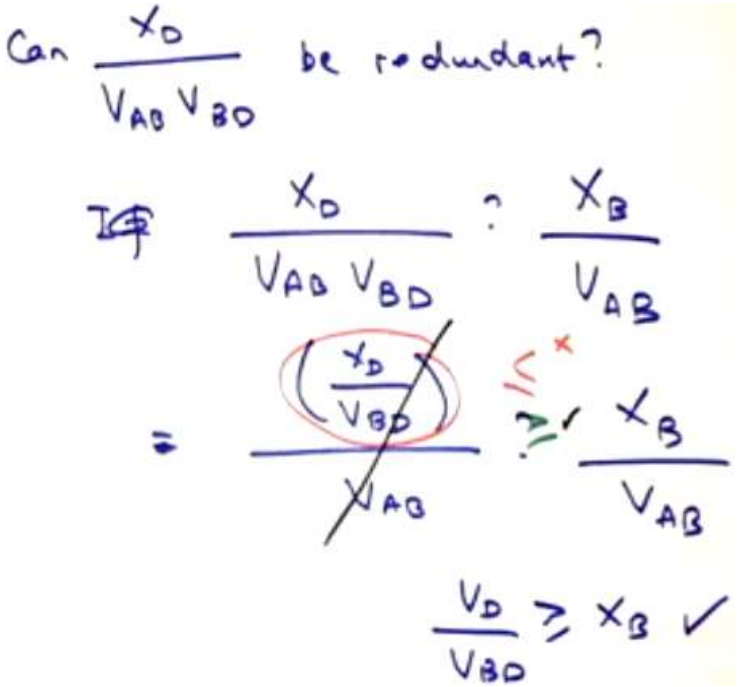
Figure 3. A more complex Web services flow graph. Web service A uses Web services B and C; B uses D and E; and C uses E and F.

What is the bound on throughput of web service A?

Bound on the throughput of web service A is:

$$X_A \le \min\left\{ \frac{X_B}{V_{AB}}, \frac{X_C}{V_{AC}}, \frac{X_D}{V_{AB}V_{BD}}, \frac{X_F}{V_{AC}V_{CF}}, \frac{X_E}{V_{AB}V_{BE}+V_{AC}V_{CE}} \right\}$$





Response time analysis

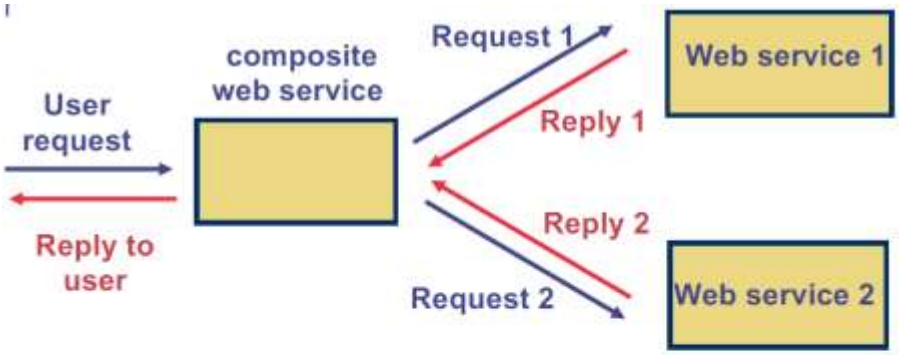
The bottleneck analysis only gives an upper bound on the throughput

Can we find the response time?

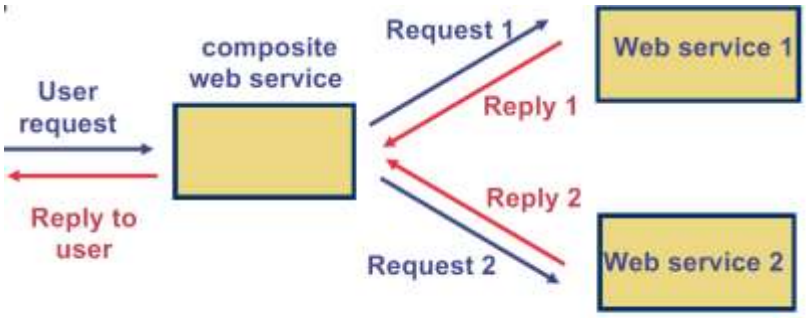
- Markov chain
- Approximate MVA

We begin with a motivating example

A simple web service scenario



- A composite web service uses two web services
- Sequence of events
  1. Composite web service receives a user request
  2. Composite web service sends Request 1 and Request 2
  3. The web services reply *independently*
     - *That is, Reply 1 and Reply 2 may arrive at different times*
  4. After the composite web service receives *both* replies, it responds to the user

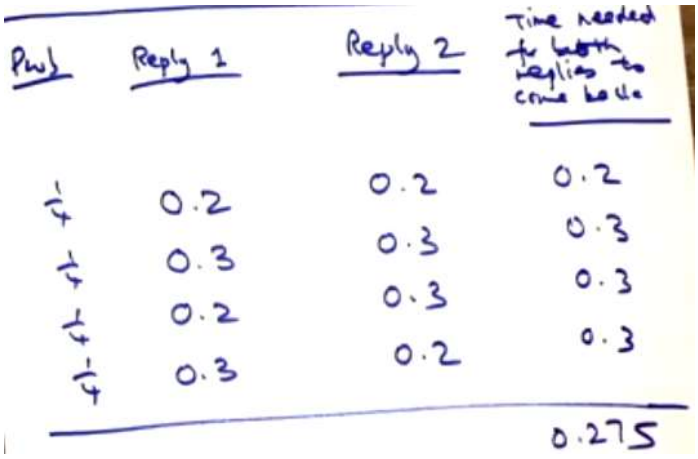


- Recall the definition of response time
- Response time of Web Service 1
  = Time at which composite web service receives Reply 1 *minus*
  Time at which composite web service sends Request 1
- Similarly for Web Service 2.

Assuming that:

- Web service 1 has a response time distribution of
  - 0.2s with probability 0.5
  - 0.3s with probability 0.5
- Web service 2 has a response time distribution of
  - 0.2s with probability 0.5
  - 0.3s with probability 0.5

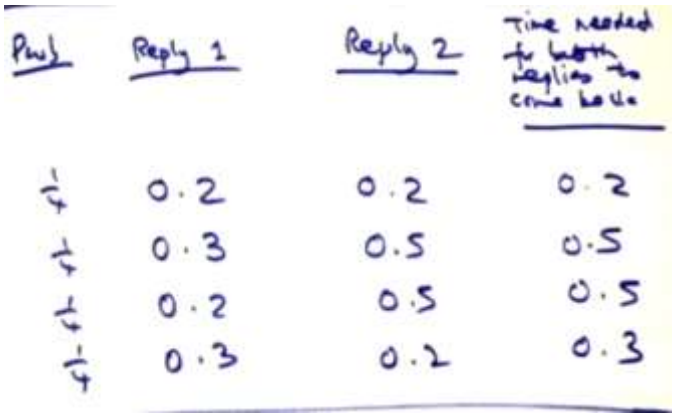
What is the average time that the composite web service has to wait until both replies are returned?



Prob	Reply 1	Reply 2	Time needed for both replies to come back
1/4	0.2	0.2	0.2
1/4	0.3	0.3	0.3
1/4	0.2	0.3	0.3
1/4	0.3	0.2	0.3
			0.275

What if the service time distribution is:

- Web service 1 has a response time distribution of
  - 0.2s with probability 0.5
  - 0.3s with probability 0.5
- Web service 2 has a response time distribution of
  - 0.2s with probability 0.5
  - 0.5s with probability 0.5



Prob	Reply 1	Reply 2	Time needed for both replies to come back
1/4	0.2	0.2	0.2
1/4	0.3	0.5	0.5
1/4	0.2	0.5	0.5
1/4	0.3	0.2	0.3

Analysis scenario

Lesson learnt: Slow web services can become the bottleneck for composite web service

We consider Composite Web Services (illustration next slide)

- With parallel invocation of N services
- Web services 1 through N-1 have a mean service time of S (exponentially distributed)
- Web service N has a mean service time of g x S (exponentially distributed)
- The next service step can only be completed after all these N steps have been completed.

Servers 1 to N-1 : mean response time = S
Server N: mean response time = g x S

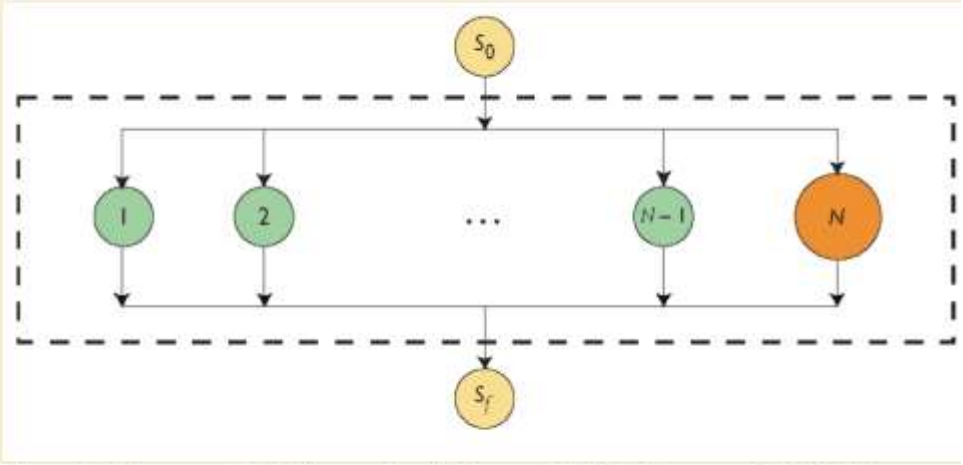
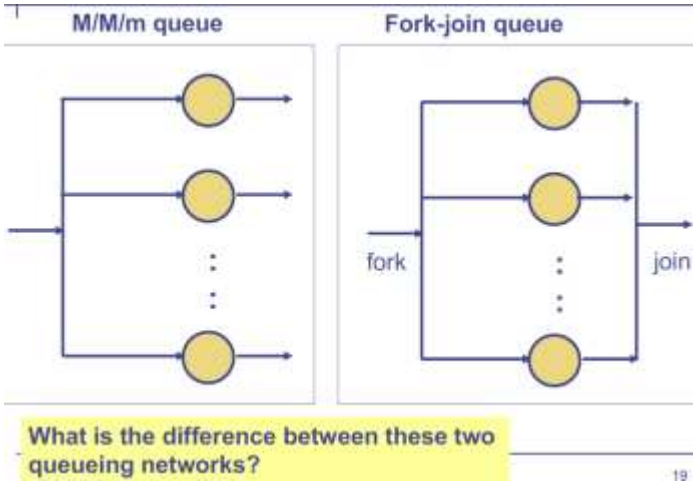


Figure 1. A composite Web service. After an initialization step S_0, N Web services are invoked in parallel. Service N takes longer than the others, and the final step S_f can only be carried out after all N services have completed.

Fork-join system

The type of system described earlier is known as fork-join system

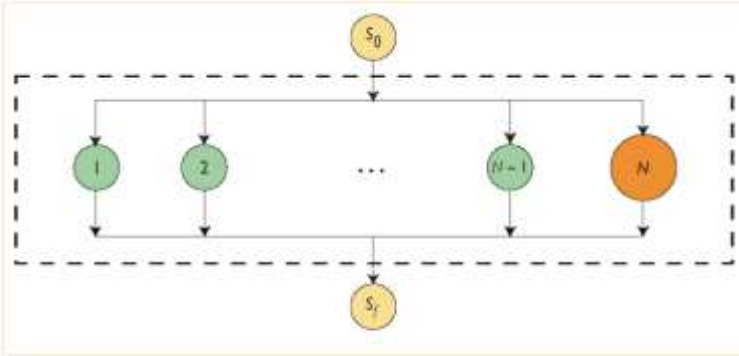
- Fork is referring to the parallel invocation
- All services must complete at the joining point before the next service can start



Fork-join queue 并行处理所有并且等到所有处理完成才进行下一步，而 M/M/m queue 只是一次处理一个响应并且处理完后就进行下一步

Servers 1 to N-1 : mean response time = S

Server N: mean response time = g x S



- We want to understand how g affects the response time of the composite web services

$$T(g) = \text{Response time of this system}$$

What is T(1)?

In this case, all constituent web services have the same response time distribution

If all mean response times are exponentially distributed with mean S

$$T(1) = \underbrace{(1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{N})}_{=H_N} S$$

$$H_N = N\text{-th harmonic number}$$

(We will explain this is obtained later.)

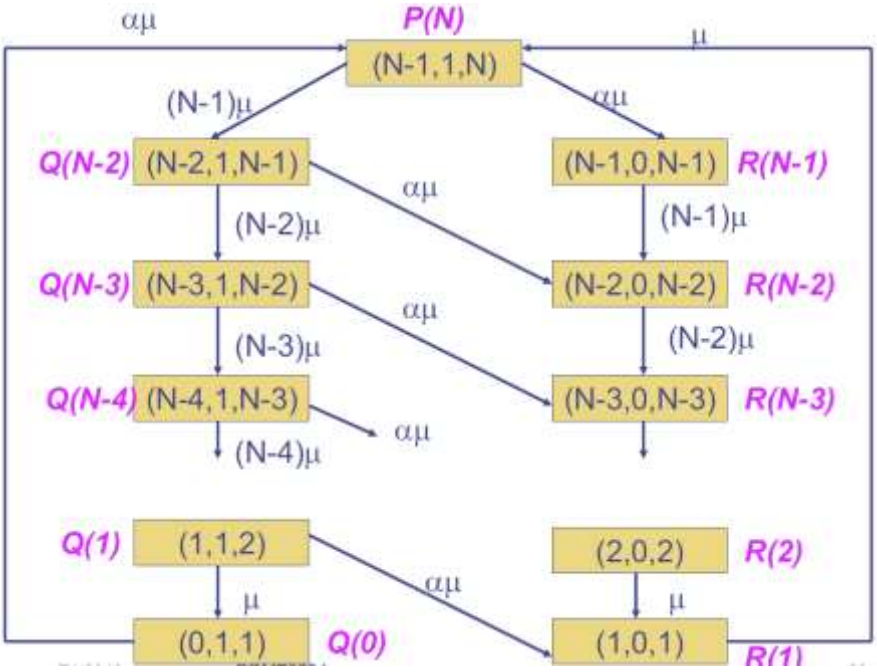
How about T(g) for g>1?

We use Markov chain.

States (i,j,k)

- i (i = 0,…,N-1) is the number of web services still running in fast Web services
- j (j = 0,1) is the number of web services running on the slow Web service
- k (k = 1,2,..,N) is the number of web services yet to complete

K 是尚未完成的服务数



$$T(g) = \frac{S}{\left(N - 1 + \frac{1}{g}\right)P(N)}$$

where

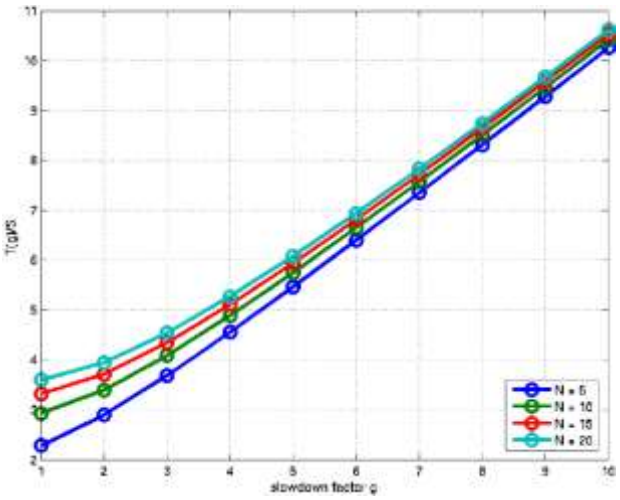
$$P(N) = \left[1 + \sum_{i=1}^{N-2} F(i) + gF(1) + V\right]^{-1},$$

$$V = \frac{1}{g}\sum_{j=1}^{N-1}\frac{1}{j}\sum_{i=j}^{N-1} F(i),$$

$$F(i) = \prod_{j=1}^{N-i-1}\frac{N-j}{N-j-1+\frac{1}{g}}.$$

Note: When g = 1, $T(g) = H_N S$

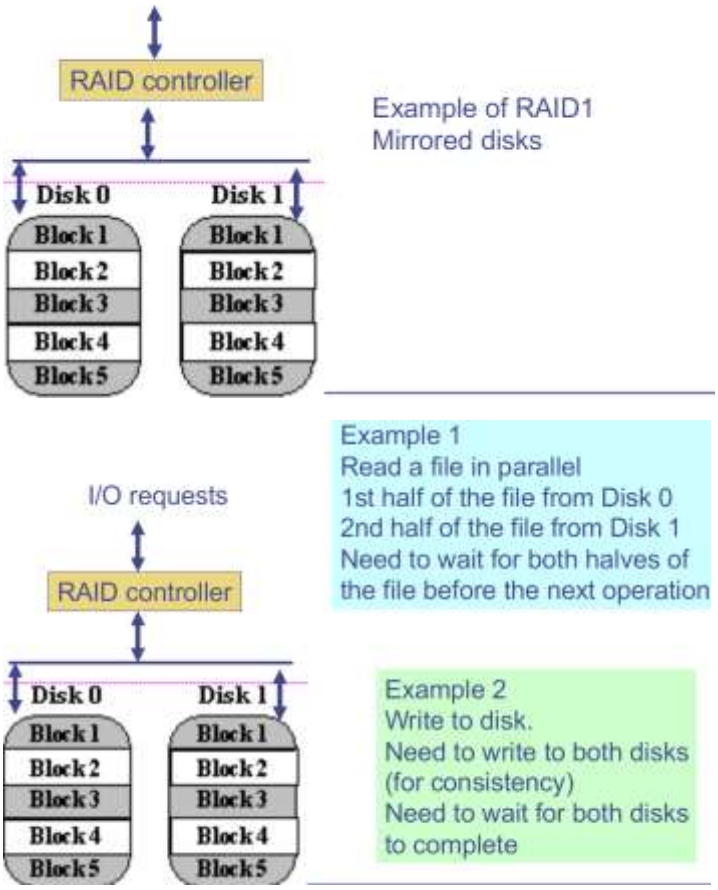
How T(g)/S varies with g?



只要有一个很慢（g=10 或者更大），就会使得不管你增加多少个服务器（N），响应时间都是趋于一样

Other examples of fork-join QNs

- Disk array, e.g. RAID (= Redundant Array of Independent Disks)

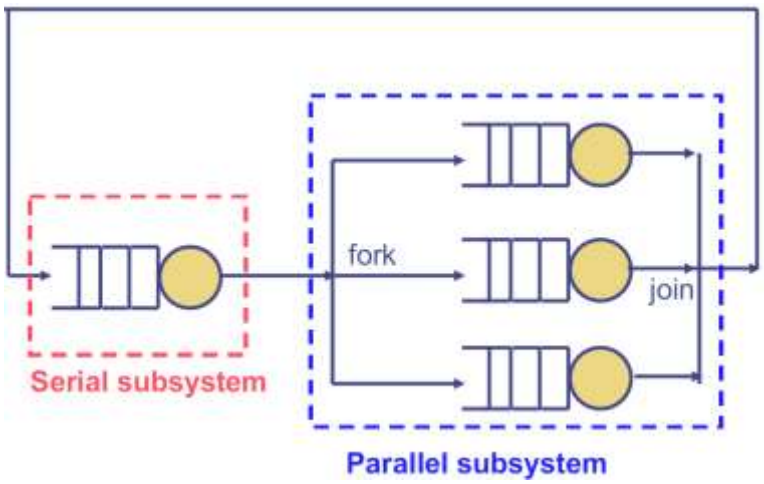


Fork-join queueing networks

Exact results are hard to come by

Approximate solution methods are used

A Queueing network with a fork-join subsystem



Approximate MVA for fork-join queueing networks

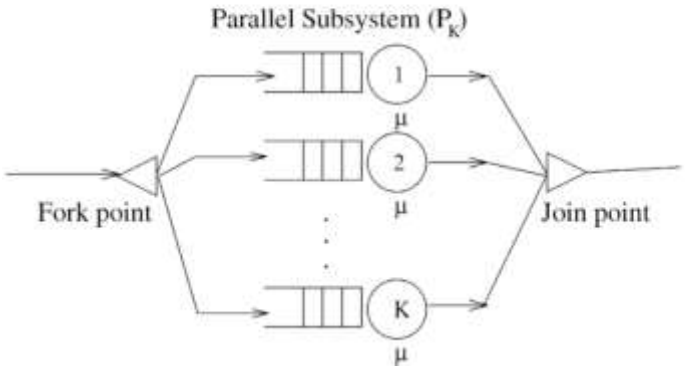
For MVA with fork-join, the basic unit is a subsystem

- A subsystem can be either a serial subsystem (= a device) or parallel one
  - A serial subsystem is a special case of parallel subsystem
- In comparison, the basic unit for MVA before is a device

Arrival Theorem for Parallel Subsystems

Consider a parallel subsystem with *k* parallel service centres

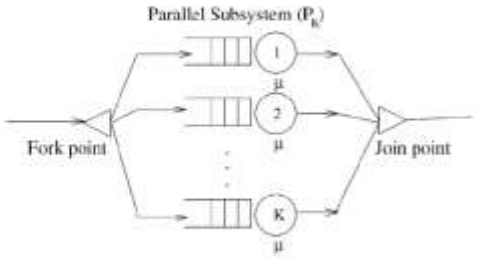
The average time each job requires at each service centre is S (exponentially distributed)



$\mu = 1/S$

When there are $n-1$ jobs in the whole QN, the average number of jobs in the subsystem is z. When there're n jobs in the system

One of the
n jobs (customers)



$$\text{Waiting time} = S \times z; \text{ Service time} = S \times H_k$$

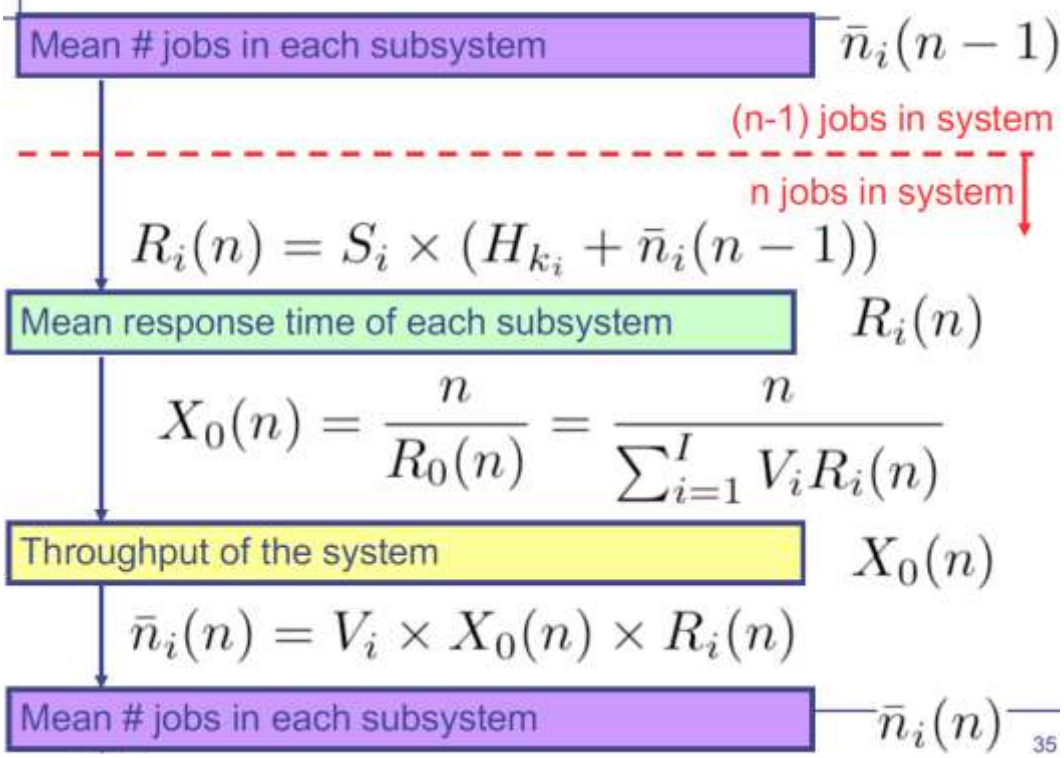
$$\Rightarrow \text{Response time} = S \times (H_k + z)$$

Note that if k = 1, the subsystem is serial and is identical to a device in MVA analysis that we have seen before.

$$\begin{aligned}\text{Response time} &= S \times (H_1 + z) \\ &= S \times (1 + z) \\ &\quad (\text{Since } H_1 = 1)\end{aligned}$$

This is the same arrival theorem that we've seen before.

MVA for fork-join systems:



I = Number of subsystems in the QN

S_i = Avg. service time of a station in subsystem i

k_i = # parallel stations in subsystem i

$R_i(n)$ = Response time at subsystem i when there're n jobs in the QN

$\bar{n}_i(n)$ = Avg. # of jobs at subsystem i when there're n jobs in the QN

V_i = Visit ratio of subsystem i

Example

- A system consists of a processor and 2 disk arrays
- Disk arrays operate under synchronous workload
  - Transactions are blocked until I/O are completed

	Service demand	# parallel systems
Processor	0.01	1
Disk array 1	0.02	2
Disk array 2	0.03	3

What is the system response time when there are 50 transactions? How many transactions can the system have if the system response time should not exceed 1s?

The MVA algorithm on p.35 assumes that you have both visit ratios V_i and mean service time S_i available

You may recall that service demand $D_i = V_i * S_i$

Now, let us assume that you are only given the service demands D_i. That is, you know only D_i but you do not know V_i and S_i. How can you modify the MVA algorithm on p.35 so that it can work with knowing service demands only?

$$X_0(n) = \frac{n}{\sum_{i=1} V_i R_i(n)}$$

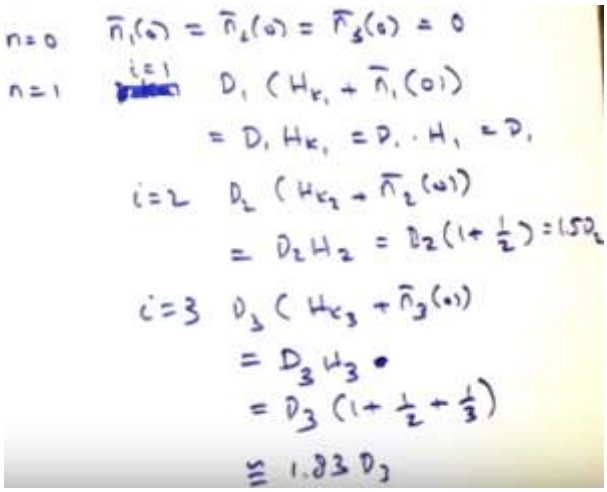
$$= \frac{n}{\sum_{i=1} V_i S_i (H_{K_i} + \bar{n}_i(n-1))}$$

$$= \frac{n}{\sum_{i=1} D_i (H_{K_i} + \bar{n}_i(n-1))}$$

$$\bar{n}_i(n) = V_i X_0(n) R_i(n)$$

$$= V_i X_0(n) S_i (H_{K_i} + \bar{n}_i(n-1))$$

$$= D_i X_0(n) (H_{K_i} + \bar{n}_i(n-1))$$



求 Hk 的话，看有多少个平行的 stations 在 system 里面就可以了

Week 7A: Revision problems

Consider a composite web service which makes use of three web services. You can model this as a fork-join queue with 3 servers. Assuming that all the 3 servers have exponentially distributed service time with mean service rate μ. Assuming also that the service time distributions at all servers are independent. We showed in the lecture, by using Markov chain that the mean service time of the fork-join queue is given by

$$(1 + \frac{1}{2} + \frac{1}{3})\frac{1}{\mu} \tag{1}$$

In this question, you will derive the same result using probability distribution.

(a) What is the probability that the service time of the fork join queue is less than x? (Hint: If the service time of the fork join queue is less than x, it means that the service time at each server must also be less than x.)

(b) You have in fact derived the cumulative probability density in Part (a). Find the probability density of the service time of the fork join queue.

(c) Using your result in Part (b), find the mean service time at the fork join queue.

Remark: An interesting exercise is to do that for n servers. I leave it to you.

(a) If the service time of the fork join queue is less than x, then it means that the service time at each web service must be less than x.

Therefore, probability that the service time of the fork join queue is less than x is the same as the probability that the service times at the 3 servers are all less than x. Since the probability that the service time at each server is less than x is $1-\exp(-\mu x)$ (this is the cumulative probability density function of exponential distribution with rate μ) and since we assume that the service time distributions at the servers are independent, we have the probability that the service times at all the 3 servers is less than x is $(1-\exp(-\mu x))^3$.

(b) Let X be the random variable on the service time of the fork join queue. We have derived in Part (a) that $\text{Prob}[X \leq x] = (1-\exp(-\mu x))^3$. This is in fact the cumulative probability density of X. The probability density function is the derivative of the cumulative probability density, which is $3\mu\exp(-\mu x)(1-\exp(-\mu x))^2$

(c) The mean service time of the fork join queue is therefore:

$$E[X] = \int_{x=0}^{\infty} x\, 3\mu\exp(-\mu x)(1-\exp(-\mu x))^2 dx \quad (1)$$

By doing the integration, we find that

$$E[X] = (1+\frac{1}{2}+\frac{1}{3})\frac{1}{\mu} \quad (2)$$

Remark: An interesting exercise is to do that for n servers. I leave it to you.

Week 7B: Optimization (1): Linear programming

The lectures for next three weeks will focus on optimization methods for network related design and applications

You will learn:

- How to formulate optimization problems
- Tools to solve optimization problems

An introduction only, because optimization is a big topic

- Emphasis is on applying optimization methods rather than the theory behind

Motivation

- A modern approach to managing computer networks is based on the concept of *software-defined networking*

- Two types of nodes:

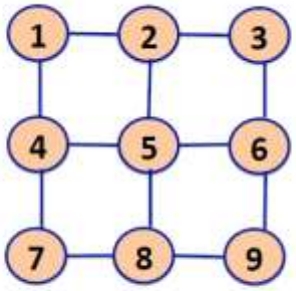
1. Simple packet switches
2. Controllers

- A controller can control a number of simple packet switches but they must be placed in a strategic location in the network

- If the delay between the controller and a packet switch is too long, then it can degrade the network performance

Consider the following network where there is a packet switch at each node and the delay on each link is 1 time unit.

Question: Assuming you want to place one controller in the network, where will you place the controller?



Question: What if you want to place two controllers?

Elements of an optimisation problem

- You want to maximise your WAM and still have a life

Maximise WAM(x_1,x_2,x_3,...)
x_1 hours/week on COMP9334
x_2 hours/week on COMPxxxx
x_3 hours/week on socialising
$x_1 \geq 10$
$x_2 \leq$ maxSocialHours
$x_1 + x_2 + x_3 + ... \leq$ totalAwakeHours

- Elements of an optimisation problem
  - Minimise or maximise an objective function
  - Decision variables: x_1,x_2, ... etc.
  - Constraints

What is optimization?

In mathematics, also known as mathematical programming

- The term programming refers to planning of activities to obtain an optimal result, not computer programming
- The amount or level of each activity can be represented as a variable whose value is to be determined

Optimization means solving problems in which we seek to minimize or maximize the value of an objective function of many decision variables, subject to constraints on the decision variables

Motivating example 1: Cloud/Grid computing

Service providers sell computing power as an utility

- Computing power measured in CPU cycles

Target customers

- Financial company, pharmaceutical company, etc.

Quality of Service in Cloud computing

- Different service providers might offer the service at different levels for different costs
- Optimization problem: How to select service providers (allocate resources) to achieve the best level of service without exceeding budget

Cloud computing resource allocation

Resource 1
Speed: 1,000 million cycles/sec
Cost: 0.1 dollars/sec

Resource 2
Speed: 2,000 million cycles/sec
Cost: 0.25 dollars/sec

Resource 3
Speed: 3,000 million cycles/sec
Cost: 0.6 dollars/sec

A computation job:

- Requires 10^7 million cycles
- Must be completed in at most 4,800 sec
- Cost must not exceed 1,500 dollars

Exercises: For the time being, let us ignore the constraint on the completion time and cost.

- If you use Resource 1 only, what is the completion time and cost?
- Repeat for Resources 2 and 3.

Completion time and cost for each resource:

- Resource 1: Completion time = 10,000 sec, cost = 1,000 dollars
- Resource 2: Completion time = 5,000 sec, cost = 1,250 dollars
- Resource 3: Completion time = 3,333 sec, cost = 2,000 dollars

Assume the computation job can be arbitrarily split into up to three parallel tasks

Question: How should the job be split, so that completion time T is minimized subject to two constraints:

- Completion time constraint: $T \leq 4{,}800$ sec
- Cost constraint: $C \leq 1{,}500$ dollars

Trial and error: Solution 1 （no Resource 2）

48% to Resource 1, 52% to Resource 3

- Resource 1: Completion time = 4,800 sec, cost = 480 dollars
- Resource 3: Completion time = 1,733 sec, cost = 1,040 dollars

Job completion time = 4,800 sec (remember jobs run in parallel)

cost = 1,520 dollars, Infeasible solution

A solution is feasible if all the constraints are satisfied

A solution is infeasible if not all the constraints are satisfied

Trial and error: Solution 2 (no Resource 1)

70% to Resource 2, 30% to Resource 3

- Resource 2: Completion time = 3,500 sec, cost = 875 dollars
- Resource 3: Completion time = 1,000 sec, cost = 600 dollars

Job completion time = 3,500 sec, cost = 1,475 dollars

Feasible solution

Trial and error: Solution 3

Resource 1
Speed: 1,000 million cycles/sec
Cost: 0.1 dollars/sec

Resource 2
Speed: 2,000 million cycles/sec
Cost: 0.25 dollars/sec

Resource 3
Speed: 3,000 million cycles/sec
Cost: 0.6 dollars/sec

30% to Resource 1, 30% to Resource 2, 40% to Resource 3

- Resource 1: Completion time = 3,000 sec, cost = 300 dollars
- Resource 2: Completion time = 1,500 sec, cost = 375 dollars
- Resource 3: Completion time = 1,333 sec, cost = 800 dollars

Job completion time = 3,000 sec, cost = 1,475 dollars

Feasible solution

Optimizing resource allocation

Given:

- Job requirement = 10^7 million cycles
- Completion time $\leq$ 4,800 sec
- Budget $\leq$ 1,500 dollars

Let:

- x_1 = fraction of the job to Resource 1
- x_2 = fraction of the job to Resource 2
- x_3 = fraction of the job to Resource 3

Find x_1, x_2 and x_3 such that

- All requirements are met
- Completion time is minimized

Completion time:

- Resource 1 = $\frac{10^7 \times x_1}{1000} = 10000 \times x_1$
- Resource 2 = $\frac{10^7 \times x_2}{2000} = 5000 \times x_2$
- Resource 3 = $\frac{10^7 \times x_3}{3000} = \frac{10000}{3} \times x_3$
- Job completion time $T = \max(10000 \times x_1, 5000 \times x_2, \frac{10000}{3} \times x_3)$

Cost:

- Resource 1 = $0.1 \times 10000 \times x_1 = 1000 \times x_1$
- Resource 2 = $0.25 \times 5000 \times x_2 = 1250 \times x_2$
- Resource 3 = $0.6 \times \frac{10000}{3} \times x_3 = 2000 \times x_3$
- Cost $C = 1000 \times x_1 + 1250 \times x_2 + 2000 \times x_3$

Mathematically, the optimization problem can be formulated as

$$\min \ T$$

subject to

$$\begin{aligned} T &\geq 10000 \times x_1 \\ T &\geq 5000 \times x_2 \\ T &\geq \frac{10000}{3} \times x_3 \\ T &\leq 4800 \\ 1000 \times x_1 + 1250 \times x_2 + 2000 \times x_3 &\leq 1500 \\ x_1 + x_2 + x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Components of an optimization problem

Given parameters

Decision variables

- In this example, they are x_1, x_2, x_3 and T

Objective function

- Can be minimization or maximization
- Can be single objective or multi-objective

Constraints

Exercise

Consider the following optimization problem where x is the decision variable:

$$\min_x \quad 2x - 1$$

subject to

$$\begin{aligned} x &\leq 20 \\ x &\geq 8 \end{aligned}$$

What are the feasible solutions?

What is the optimal solution?

LP solvers

Many commercial and free software are available for solving LP problems

Commercial software

- Capable of solving large LP problems, e.g. millions of variables
- A 50,000-variable LP problem takes about 5 seconds on a standard linux PC

LP solvers require the user to write the problem in fixed format

Can be embedded in C, C++ or Java, e.g.

```
model.add(IloMinimize(env,-9*x[0]+x[1]+4*x[2]));
model.add(-x[0]+x[2] == -3);
model.add( x[0]-x[1] <= 1);
```

Can be used with some modeling languages

- AMPL
- MPS
- GAMS

Week 7B: Revision problems

A company has two servers, which we will refer to as Servers 1 and 2. The response time of each server depends on the workload on the server.

For Server 1, if the workload is W_1, then its response time is $3+5W_1$. For Server 2, if the workload is W_2, then its response time is $1+7W_2$.

Assuming that you have a workload of W units, which you can split between the two servers with the following restrictions:

- A minimum of $W_{1,\text{base}}$ units of workload must be sent to Server 1.
- A minimum of $W_{2,\text{base}}$ units of workload must be sent to Server 2.
- The total workload of the two servers must be W

The response time of the system is defined as the maximum of the response times from the two servers because you need to wait for the workloads from both servers to complete.

Your tasks are:

(a). Formulate a linear programming problem to determine how the workload W should be split between the two servers so that the response time of the system is minimised.

(b). Assuming that $W = 6$, $W_{1,\text{base}} = 0$ and $W_{2,\text{base}} = 0$, solve the linear programming numerically.

(c). Calculate the response times of the two servers using the optimal splitting that you have computed in Part (b). What do you notice? Why do you think this should happen?

(d). Assuming that $W = 6$, $W_{1,\text{base}} = 2.8$ and $W_{2,\text{base}} = 2.8$, solve the linear programming numerically.

(e). Calculate the response times of the two servers using the optimal splitting that you have computed in Part (d). What do you notice? Why do you think this should happen?

(a). The decision variables are W_1 and W_2. The linear programming problem is:

$$\min \quad R$$

subject to

$$\begin{aligned} 3 + 5W_1 &\leq R \\ 1 + 7W_2 &\leq R \\ W_1 &\geq W_{1,\text{base}} \\ W_2 &\geq W_{2,\text{base}} \\ W &= W_1 + W_2 \end{aligned}$$

(b). The files `hw_lp.mod` and `hw_batch` are available at the course website. The workload allocation should be $W_1 = 3.3333$ and $W_2 = 2.6667$.

(c). If $W_1 = \frac{10}{3}$, then the response time of Server 1 is $3 + 5 \times \frac{10}{3} = \frac{59}{3}$. If $W_2 = \frac{8}{3}$, the response time of Server 2 is $1 + 7 \times \frac{8}{3} = \frac{59}{3}$. You will notice that the response times of the two servers are identical.

In order to explain why the response times are identical, let us work out some examples to understand the problem a bit more.

Let us say if we allocate 3 units of workload to each server. The response time of Server 1 is $3 + 5 \times 3 = 18$. The response time of Server 2 is $1 + 7 \times 3 = 22$. The system response time is the bigger of 18 and 22, which is 22.

Since our goal is to minimise the system response time, let us see how we can reduce the system response time. Since both response times are strictly increasing function of the workload, you can increase (resp. decrease) the response time by increasing (decreasing) the workload. Since Server 2 has the higher response time, it determines the system response time. (Think about bottleneck!) Therefore, if we want to try to reduce the system response time, we should try to reduce the response time of Server 2. This can be done by shifting some workload from Server 2 to Server 1. Let us try to decrease the response time of Server 2 by shifting some workload to Server 1. Let us try $W_1 = 3.1$ and $W_2 = 2.9$. With this allocation, we have:

response time of Server 1 is $3 + 5 \times 3.1 = 18.5$ and the response time of Server 2 is $1 + 7 \times 2.9 = 21.3$. The system response time is the bigger of 18.5 and 21.3, which is 21.3.

The example in the last paragraph shows that we can decrease the system response time by shifting some workload from Server 2 to Server 1. We can keep doing that until the response time of the two servers are the same. When this happens, we can no longer decrease the system response time any longer because any shift in the workload between the servers will increase the system response time. This is the minimum system response time.

(d). To achieve the minimum system response time, W_1 should be 3.2 and W_2 should be 2.8.

(e). If $W_1 = 3.2$, then response time of Server 1 is $3 + 5 \times 3.2 = 19$. If $W_2 = 2.8$, the response time of Server 2 is $1 + 7 \times 2.8 = 20.6$. Server 2 has the higher response time so it determines the system response time. If you want to further reduce the response time, you need to shift some workload from Server 2 to Server 1. However, the workload on Server 2 is the minimum allowed, so we cannot shift any more workload from Server 2 to Server 1.

Week 8A: Optimization (2): Integer programming

What have we covered so far?

- Formulation of linear programming (LP) problems
- Using AMPL to solve LP problems

This lecture

- Different types of LP problems
- Formulation of integer programming (IP) problems
- Using AMPL to solve IP problems

Linear programming

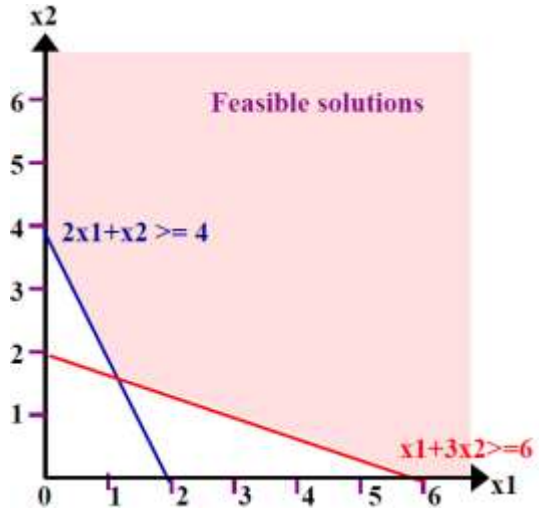
Linear programming (LP) is a tool for solving a type of optimization problems where

- Decision variables are real numbers
- Objective function is linear in the decision variables
- All the constraints are linear in the decision variables

LP problems with 2 variables have a nice graphical solution, e.g.

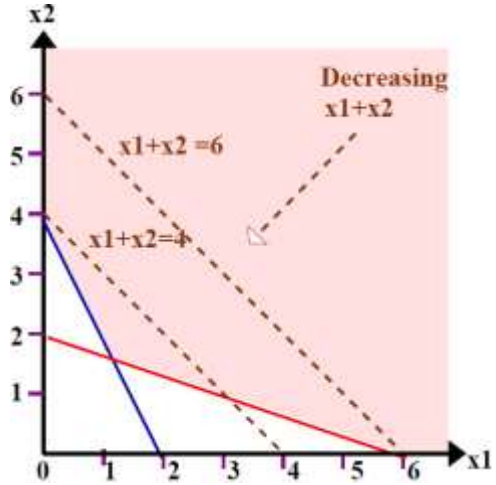
$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_1 + 3x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Feasible solutions



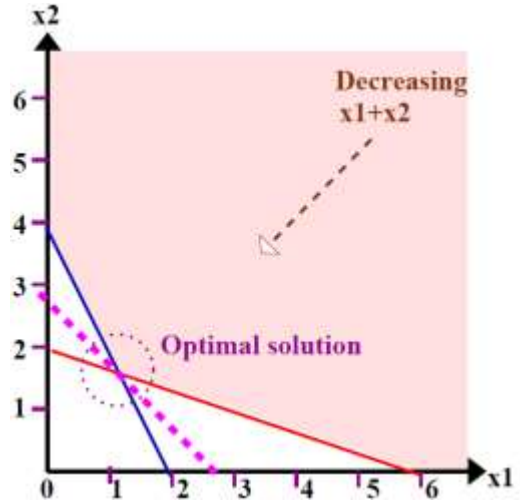
- Any (x_1, x_2) that satisfies all the constraints is a feasible solution. Otherwise, it is an infeasible solution
- For LP problems with 2 variables, the feasible region is a polygon
- In general, it is a polyhedron

Objective function



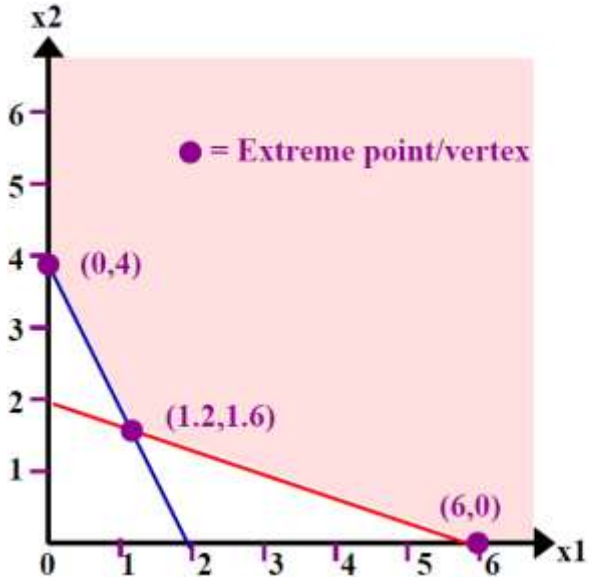
- Since the aim is to minimize $x_1 + x_2$, we look for contour of $x_1 + x_2$

Optimal solution



- Optimal solution of this LP problem is $x_1 = 1.2$, $x_2 = 1.6$, with objective function value = 2.8

Extreme point



- Regardless of the objective function, the optimal solution of an LP problem (if it exists) must be at one of the extreme points or vertices of the polyhedron formed by the constraints

Special cases: Multiple optimal solutions

What is the optimal solution to the following LP problem?

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & x_1 + x_2 \geq 4 \\ & x_1 + 3x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Special cases: Infeasible LP

What is the optimal solution to the following LP problem?

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & x_1 + 3x_2 \leq 4 \\ & x_1 + 3x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Special cases: Unbounded LP

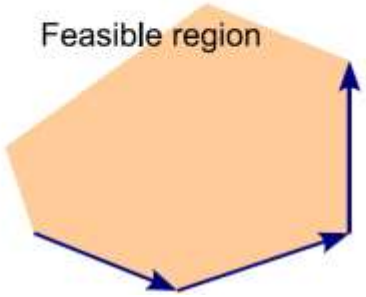
What is the optimal solution to the following LP problem?

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_1 + 3x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Algorithms for LP

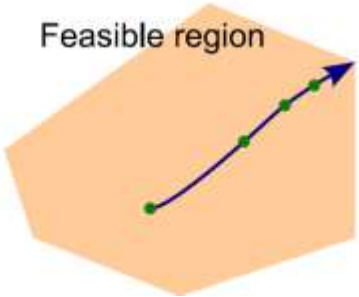
Simplex method (Dantzig 1947)

- Found in many LP software
- Principle: Move from an extreme point to another
- Worst-case complexity: Exponential
- Average performance on practical LP problems: Very good



Interior-point method (Karmarkar 1984)

- Very competitive compared with the simplex method
- Principle: Move from an interior point to another
- Worst-case complexity: Polynomial
- The algorithm for large LP problems



Motivating example 2: Cloud computing

Now, each resource further charges a set up cost of fixed amount

- Resource 1: Set up cost = 200 dollars
- Resource 2: Set up cost = 100 dollars
- Resource 3: Set up cost = 50 dollars

Again, the computation job has the following requirements:

- Requires 10^7 million cycles
- Completion time $\leq$ 4,800 sec
- Cost $\leq$ 1,500 dollars
- Minimize the completion time

Problem faced by the job:

- From which resource should we buy the computing power?
- How many cycles to buy from each chosen resource?

Yes-or-no decision

What is the cost of buying cycles from a chosen resource?

Yes-or-no questions: Is Resource i chosen?

- E.g. is Resource 3 chosen?
  - Yes $\Rightarrow x_3 > 0 \Rightarrow$ Cost = $2000 \times x_3 + 200 \times 1$
  - No $\Rightarrow x_3 = 0 \Rightarrow$ Cost = $2000 \times x_3 + 200 \times 0$

This type of optimization problems involves 0-or-1 logical decisions among other decisions

You will learn in the rest of this lecture

- How to formulate this type of optimization problems
- How to solve this type of optimization problems

Formulating optimization problem

Given:

- n resources
- Resource i offers computing power at a speed of p_i million cycles/sec with cost c_i dollars/sec
- Set up cost for using Resource i is s_i dollars
- Customer requires N million cycles
- Completion time $\leq T_{\max}$
- Cost $\leq C_{\max}$

Decision variables:

$$\begin{aligned} y_i &= \begin{cases} 1 & \text{if Resource } i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \\ x_i &= \text{fraction of the job allocated to Resource } i \\ T &= \text{completion time, which is } \max_i \frac{Nx_i}{p_i} \end{aligned}$$

Exercise: Express the cost of using Resource i in terms of c_i, N, x_i, p_i, s_i, y_i

Cost:

$$C = \sum_{i=1}^{n} \left(\frac{c_i N x_i}{p_i} + s_i y_i \right)$$

Constraint: Cannot have cycles from Resource i if it is not chosen

$$x_i \leq y_i, \quad i = 1, 2, \ldots, n$$

- Note that we require $x_i \geq 0$, thus
  - Exercise: If $y_i = 0$, what value(s) can x_i take?
  - Exercise: If $y_i = 1$, what value(s) can x_i take?

The problem formulation is

$$\min \ T$$

subject to

$$\begin{aligned} T &\geq \frac{Nx_i}{p_i}, \quad i = 1, 2, \ldots, n \\ T &\leq T_{\max} \\ \sum_{i=1}^{n} \left(\frac{c_i N x_i}{p_i} + s_i y_i \right) &\leq C_{\max} \\ x_i &\leq y_i, \quad i = 1, 2, \ldots, n \\ \sum_{i=1}^{n} x_i &= 1 \\ x_i &\geq 0, \quad i = 1, 2, \ldots, n \\ y_i &\in \{0, 1\}, \quad i = 1, 2, \ldots, n \end{aligned}$$

Integer programming

LP in which some or all decision variables can only take non-negative integer values

Specific type of integer programming (IP):

- Mixed integer programming (MIP): Some decision variables are integers, others are real
- Pure integer programming: All decision variables are integers
- Binary integer programming: All decision variables must be either 0 or 1

LP relaxation

Relaxing the integer constraints may not give you the right solution

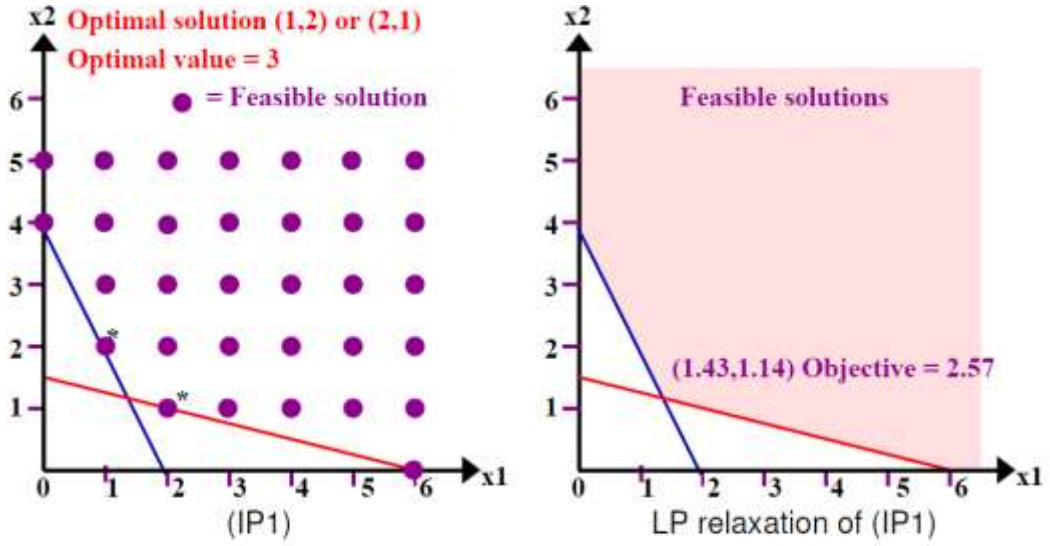
IP problem (IP1)

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_1 + 4x_2 \geq 6 \\ & x_1, x_2 \geq 0, \text{ integer} \end{aligned}$$

LP relaxation of (IP1)

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_1 + 4x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Feasible solutions



- Question: Can we round the LP relaxation solution to obtain a solution to (IP1)?

Results and observations

Rounding the LP relaxation solution may produce an infeasible solution for the original IP problem

However, for minimization problems, it is always true that:
Optimal value of an IP problem $\geq$ Optimal value of its LP relaxation

If the LP relaxation gives an integer solution

- The optimal solution to the LP relaxation is also an optimal solution to the original problem
- This is true for some special classes of IP problems e.g. network flow problems

Solving IP exactly

Complete enumeration will require a lot of computation

A smarter way is to use the branch-and-bound method (you are encouraged to read Winston sections 9.3, 9.4 in your own time)

In principle, the branch-and-bound method can find the optimal solution but practically it may take a very long time

- Some IP problems with 60 variables may take hours to run

Week 8A: Revision problems

1. A grid computing customer has a job which requires 10^7 Mcycles to process and must be completed within 4000s. The customer can choose between three companies:

   - Company 1: 1000 Mcycles/s at \$0.1/s with a set up cost of \$500.
   - Company 2: 2000 Mcycles/s at \$0.3/s with a set up cost of \$100.
   - Company 3: 3000 Mcycles/s at \$0.8/s with a set up cost of \$0.

   Assuming the job can be divided arbitrarily among the three companies for processing. Your goal is to find the combination of companies and the number of cycles to be bought from each company so that the job is completed on time and the cost is minimised.

   (a) Formulate this as a mixed integer programming (MIP) optimization problem.
   (b) Solve this problem by using any MIP solver you choose.

1. (a) Let us define a few notation:

- $T_{\max}$ = computation time limit = 4000s
- P = total number of cycles required = 10^7 Mcycles.
- p_i = Speed (in Mcycles/s) of company i, e.g. $p_1 = 1000$ Mcycles/s
- c_i = Per-second charge for company i, e.g. $c_1 = \$0.1$/s.
- s_i = Set up cost for company i, e.g. $s_1 = 500$.

The decision variables are y_i and x_i (for $i = 1, 2, 3$):

$$y_i = \begin{cases} 1 & \text{if Company } i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

$$x_i = \text{fraction of cycles to be bought from Company } i \text{ and is } \geq 0$$

Based on these decision variables, we know that

- Completion time T is:

$$T = \max_i \frac{Px_i}{p_i}$$

- Total cost C is:

$$C = \sum_{i=1}^{n} \left(\frac{c_i P x_i}{p_i} + s_i y_i \right)$$

The problem formulation is

$$\begin{aligned} \min \quad & C \\ \text{subject to} \quad & T_{\max} \geq \frac{Px_i}{p_i} && 1 \leq i \leq n \\ & C = \sum_{i=1}^{n} \left(\frac{c_i P x_i}{p_i} + s_i y_i \right) \\ & x_i \leq y_i && 1 \leq i \leq n \\ & \sum_{i=1}^{3} x_i = 1 \\ & x_i \geq 0 && 1 \leq i \leq n \\ & y_i \in \{0, 1\} && 1 \leq i \leq n \end{aligned}$$

(b) By using an integer programming solver, we find that we should buy 8×10^6 Mcycles from Company 2 and 2×10^6 Mcycles from Company 3. This costs a total of \$1833. The files used to solve this problem are `hw_grid.mod`, `hw_grid.dat` and `hw_grid_batch`.

2. A company has 8 databases that it can store at three different locations. The size and frequency of access to the databases are showed in Table 1. These databases can be stored in any of the three locations. Each location has a different capacity and access time and they are shown in Table 2. Note the following:

   - The size of the databases and the capacity of each location are measured in the same unit.
   - The time to access a location is assumed to be constant and is as given in Table 2.

   Your goal is to minimise the average access time (weighted by frequency of access) by assigning the databases to the three locations. The allocation must satisfy the following constraints:

   - Each database can only be assigned to exactly one location. In other words, you are not allowed to split a database between two locations nor to replicate it.
   - The total size of all the databases at one location cannot exceed the capacity of that location.

   The decision variables for this optimisation problem are binary. Let x_{ij} be the decision variable:

$$x_{ij} = \begin{cases} 1 & \text{if Database } j \text{ is stored in Location } i \\ 0 & \text{otherwise} \end{cases}$$

   Answer the following questions:

Database	Size	Frequency of Access (number of times per second)
1	7	10
2	2	4
3	3	5
4	5	6
5	3	2
6	6	3
7	3	4
8	5	3

Table 1: For Question 2

Location	Capacity	Access time (s)
1	10	0.1
2	15	0.3
3	100	0.8

Table 2: For Question 2

(a) Consider the decision variables for database 1, which are x_{11}, x_{21} and x_{31}. A constraint for the optimisation problem is that each database can only be assigned to one of the three locations. This constraint means that there are three possible combinations for (x_{11}, x_{21}, x_{31}). What are the three combinations? This also means that the sum $x_{11} + x_{21} + x_{31}$ has a constant value. What is this constant value?

(b) Formulate an integer programming problem to determine where the databases should be stored

(c) Solve this problem by using any solver you choose.

2. (a) Since Database 1 can only be placed at one of the three locations, there are three possible combinations for for x_{11}, x_{21} and x_{31}:
   i. $x_{11} = 1$, $x_{21} = 0$ and $x_{31} = 0$
   ii. $x_{11} = 0$, $x_{21} = 1$ and $x_{31} = 0$
   iii. $x_{11} = 0$, $x_{21} = 0$ and $x_{31} = 1$

   This means $x_{11} + x_{21} + x_{31} = 1$.

   (b) The decision variables are

$$x_{ij} = \begin{cases} 1 & \text{if Database } j \text{ is stored in Location } i \\ 0 & \text{otherwise} \end{cases}$$

   We use the notation:
   - D_i = capacity of Location i
   - t_i = access time of Location i
   - s_j = size of Database j
   - f_j = frequency of accessing Database j
   - F = total number of accesses per second

   The (binary) integer programming problem is:

$$\min \frac{1}{F} \sum_{i=1}^{3} \sum_{j=1}^{8} f_j t_i x_{ij}$$

   subject to

$$\begin{aligned} \sum_{j=1}^{8} s_j x_{ij} &\leq D_i \quad \text{for } 1 \leq i \leq 3 \\ \sum_{i=1}^{3} x_{ij} &= 1 \quad \text{for } 1 \leq j \leq 8 \\ x_{ij} &\in \{0, 1\} \end{aligned}$$

   Note the first set of constraints enforces capacity limit of each location. The second set of constraints ensures that each database is stored at exactly one location.

   (c) The location is as showed below.
   - Databases 2,3,4 at Location 1.
   - Databases 1,7,8 at Location 2.
   - The rest at Location 3.

Week 8B: Optimization (3): Network flow

Linear programming (LP)

- Real values for decision variables, linear in objective function, linear in constraints
- Large LP problems can be solved routinely

Integer programming (IP)

- Some decision variables can only take integer values
- Some decision variables can only take binary values, e.g. for making yes-or-no decisions
- IP problems can be solved using branch and bound in principle
- Computation complexity is generally exponential except for unimodular problems

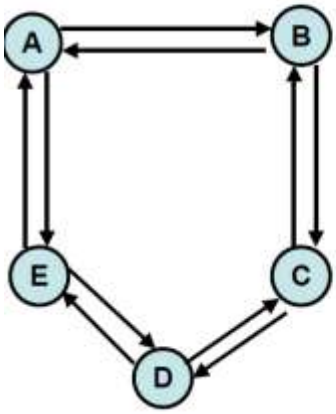
Overview

Applications of integer programming for network flow problems

- Traffic Engineering
- Dimensioning problem
- Topology design

Traffic Engineering Example

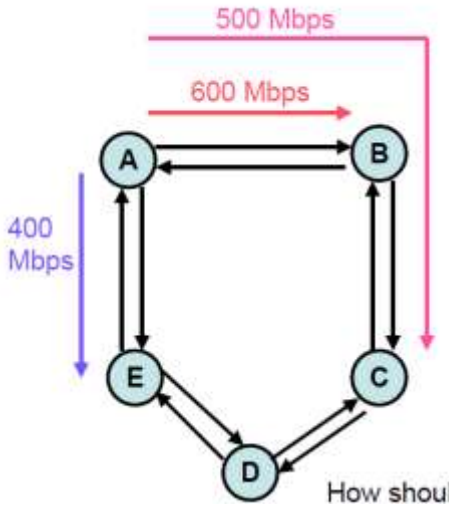
An ISP owns the following network which connects
5 cities A, B, C, D and E.
Capacity of each link is 1000 Mbps



The traffic demands between cities are:
A to B: 600 Mbps
A to E: 400 Mbps
A to C: 500 Mbps

Question: How should we route the traffic so that the links are at most 80% utilised and we use the minimum amount of resources?

Capacity of each link is 1000 Mbps

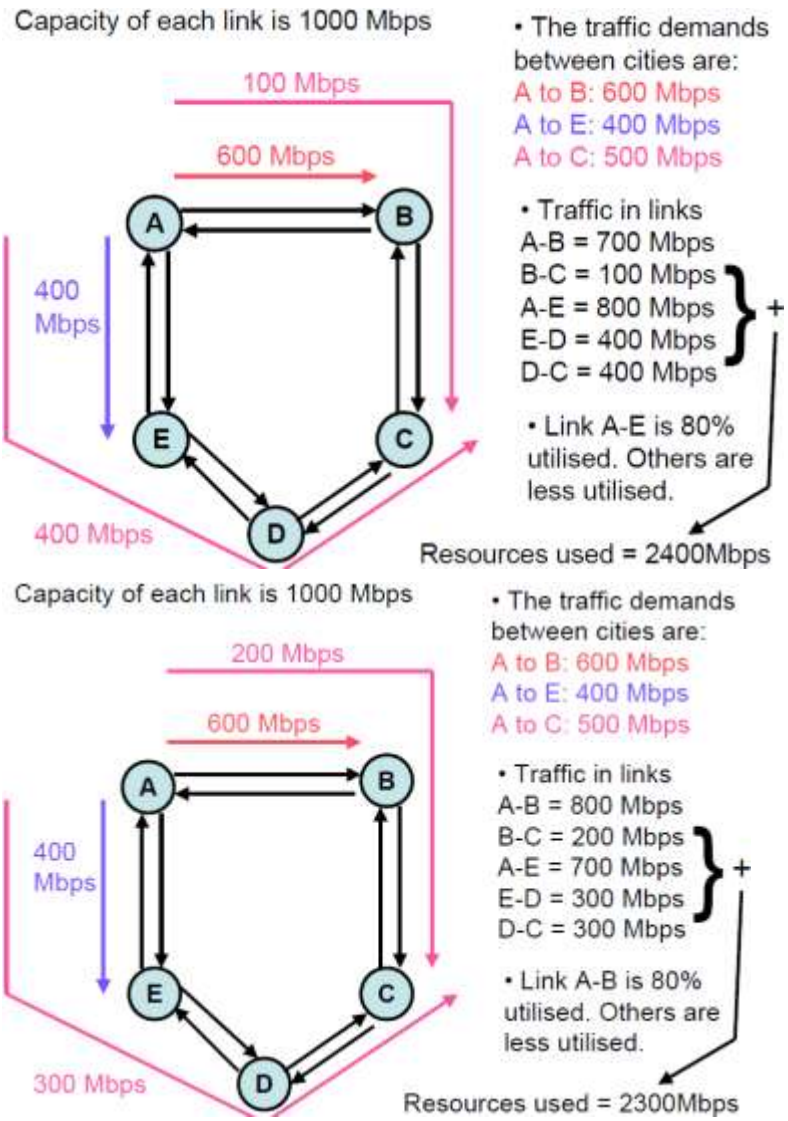


The traffic demands between cities are:
A to B: 600 Mbps
A to E: 400 Mbps
A to C: 500 Mbps

Let us assume that the traffic demand will take the shortest path between its end points

But 1100 Mbps in link A-B! The link is over-utilised!

How should you route the traffic?



Traffic Engineering

General traffic engineering problem:

- Given:
  - A network (i.e. nodes, links and their capacites)
  - The traffic demand between each pair of nodes.
- Find: how to route the traffic to best utilise the resource

The traffic engineering example earlier was simple, but for a commercial carrier (Next slide shows the network map of a commercial carrier.), it's no longer so.

- Traffic engineering problems can be solved systematically using integer programming
- These problems are generally known as network flow problems. *Note: flow is synonymous with traffic demand between a pair of nodes.*
- We will start with the simplest network flow problem, finding the shortest path for one flow.

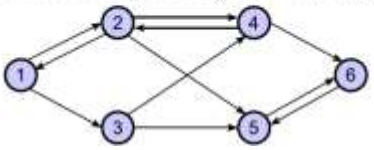
Network flow problems

Network flow problems are important applications of integer programming

- Move some entity from one point to another in the network
- Given alternative ways, find the most efficient one, e.g. minimum cost, maximum profit, etc.

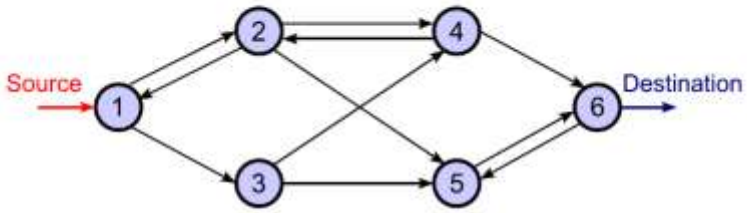
Network is represented as a directed graph $G = (N, E)$

- N = the set of nodes, e.g. $N = \{1, 2, 3, 4, 5, 6\}$
- E = the set of directed edges, e.g. $E = \{\langle 1, 2\rangle, \langle 1, 3\rangle, \langle 2, 1\rangle, \ldots\}$



Finding the shortest path

Aim: Find the shortest path from the source node to the destination node of a flow



Cost is generally assumed to be additive, i.e. cost of a path = sum of the cost of using each edge in the path

- E.g. cost of using edges $\langle 1, 2\rangle$, $\langle 2, 4\rangle$ and $\langle 4, 6\rangle$
  = cost of edge $\langle 1, 2\rangle$ + cost of edge $\langle 2, 4\rangle$ + cost of edge $\langle 4, 6\rangle$

Shortest path problem (SPP)

Given

- A directed graph $G = (N, E)$
- A flow of size 1 enters at node s (source) and leaves at node d (destination)
- It costs $c_{i,j}$ for using directed edge $\langle i, j \rangle$

Find which directed edges the flow should use in order that

- The total cost is minimized
- The entire flow must use only one path

Logical decision: Should I use a directed edge or not?

Formulating SPP

Decision variables

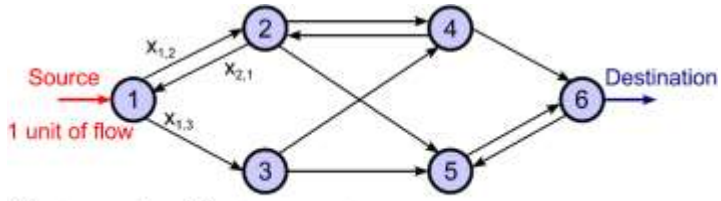
$$x_{i,j} = \begin{cases} 1 & \text{if directed edge } \langle i, j \rangle \text{ in } E \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

We assume 1 unit of flow from the source to the destination

An important part of the formulation is to make sure the directed edges selected actually form a connected path from the source to the destination

- This is by adding the conservation of flow constraints

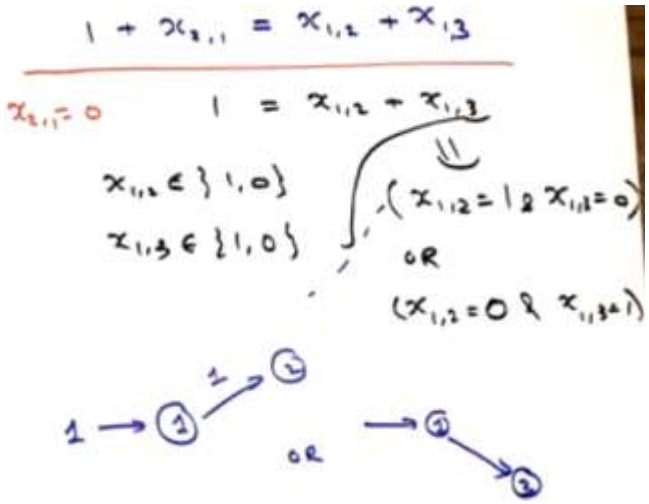
Conservation of flow: Source node



What goes in = What goes out

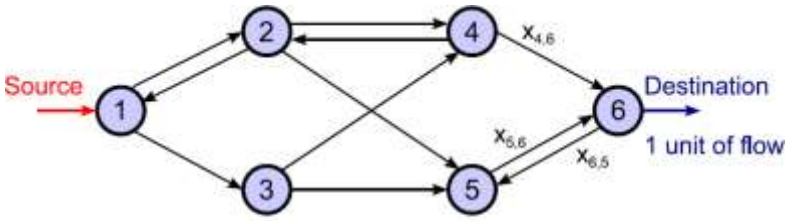
- Flow going into node 1 from external = 1
- Flow going into node 1 from neighboring nodes = $x_{2,1}$
  - Second index is "1"
- Flow going from node 1 to neighboring nodes = $x_{1,2} + x_{1,3}$
  - First index is "1"
- Therefore, we have

$$1 + x_{2,1} = x_{1,2} + x_{1,3}$$



选择一些是 0 就可以了，比如让 x2,1=0 相当于没选这条链路

Conservation of flow: Destination node

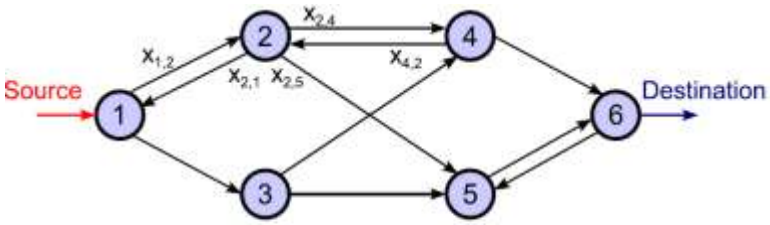


What goes in = What goes out

- Flow going into node 6 from neighboring nodes = $x_{4,6} + x_{5,6}$
  - Second index is "6"
- Flow going from node 6 to neighboring nodes = $x_{6,5}$
  - First index is "6"
- Flow going from node 6 to external = 1
- Therefore, we have

$$x_{4,6} + x_{5,6} = x_{6,5} + 1$$

Conservation of flow: Other nodes



E.g. flow conservation at node 2: What goes in = What goes out

- Flow going into node 2 from neighboring nodes = $x_{1,2} + x_{4,2}$
  - Second index is "2"
- Flow going from node 2 to neighboring nodes = $x_{2,1} + x_{2,4} + x_{2,5}$
  - First index is "2"
- Therefore, we have

$$x_{1,2} + x_{4,2} = x_{2,1} + x_{2,4} + x_{2,5}$$

Conservation of flow constraints

In our example network, the source is node 1, so the constraint is

$$1 + x_{2,1} = x_{1,2} + x_{1,3}$$

- This can be rewritten as

$$\sum_{j:\langle 1,j\rangle \in E} x_{1,j} - \sum_{j:\langle j,1\rangle \in E} x_{j,1} = 1$$

The destination is node 6, so the constraint is

$$x_{4,6} + x_{5,6} = x_{6,5} + 1$$

- This can be rewritten as

$$\sum_{j:\langle 6,j\rangle \in E} x_{6,j} - \sum_{j:\langle j,6\rangle \in E} x_{j,6} = -1$$

For all other nodes (neither a source or a destination), e.g. node 2, the constraint is

$$x_{1,2} + x_{4,2} = x_{2,1} + x_{2,4} + x_{2,5}$$

- This can be rewritten as

$$\sum_{j:\langle 2,j\rangle \in E} x_{2,j} - \sum_{j:\langle j,2\rangle \in E} x_{j,2} = 0$$

The flow conservation constraints can be written in a compact form

$$\sum_{j:\langle i,j\rangle \in E} x_{i,j} - \sum_{j:\langle j,i\rangle \in E} x_{j,i} = 0, \quad i \in N - \{s,d\}$$

IP formulation for SPP

SPP can be formulated as

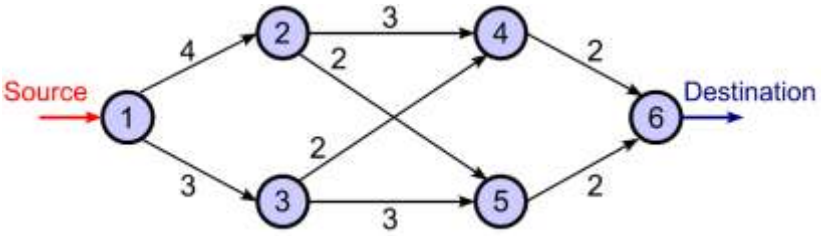
$$\min \sum_{\langle i,j \rangle \in E} c_{i,j} x_{i,j}$$

subject to

$$\sum_{j:\langle i,j \rangle \in E} x_{i,j} - \sum_{j:\langle j,i \rangle \in E} x_{j,i} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in N - \{s,d\} \\ -1 & \text{if } i = d \end{cases}$$

$$x_{i,j} \in \{0,1\} \text{ for all } \langle i,j \rangle \in E$$

SPP example



We will use AMPL/CPLEX for solving SPP in this example network

Note:

- It is far more efficient to use Dijkstra's algorithm for solving SPP
- The reason of using integer programming here is for illustration only

Introducing non-unit flow and link capacity

(Note: A dot point preceded by ★ indicates that it is different from the setting of the shortest path problem.)

- Given
  - A directed graph (N, E)
  - ★ A flow of size f with source node s and destination node d
  - It costs c_{ij} (per unit flow) for the flow to use directed edge (i, j)
  - ★ The capacity of the directed edge (i, j) is b_{ij}
- Find which directed edges the flow should use in order that
  - The total cost is minimised
  - The entire flow must use only one path
  - ★ The flow on any directed edge does not exceed its capacity

Decision variables are the same as before

$$x_{ij} = \begin{cases} 1 & \text{if directed edge } (i,j) \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

The amount of flow on directed edge (i, j) will be fx_{ij}

The problem formulation is

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

subject to

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in N - \{s,d\} \\ -1 & \text{if } i = d \end{cases}$$

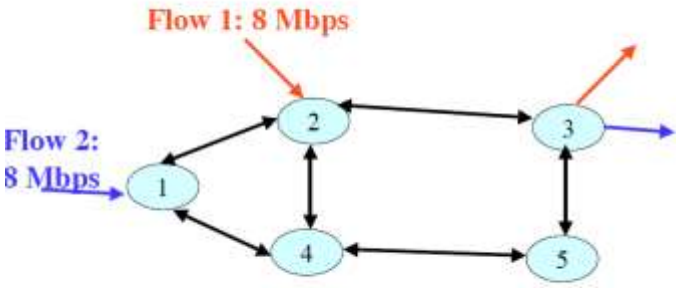
$$fx_{ij} \leq b_{ij} \text{ for all } (i,j) \in E \qquad (***)$$

$$x_{ij} \in \{0,1\} \text{ for all } (i,j) \in E$$

Note: $(***)$ — this constraint ensures that only links with sufficient capacity may be chosen to carry the flow.
Solution: Eliminate edges with insufficient capacity, then Dijkstra.

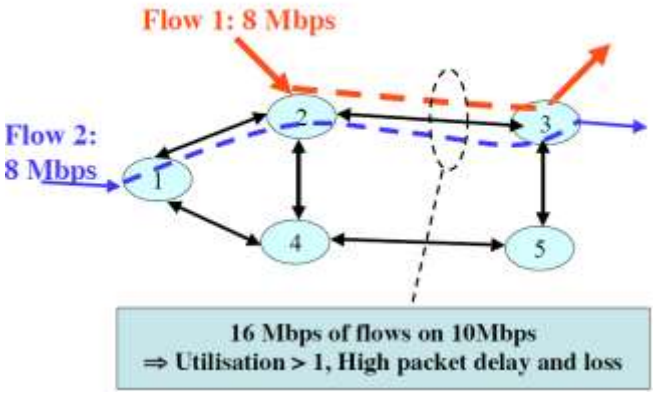
Multiple flows

Given: Each link has capacity of 10 Mbps



Assuming cost for each link is 1.
What if both flows use the shortest path?

Given: Each link has capacity of 10 Mbps



Traffic engineering problem

- Given
  - A directed graph (N, E)
  - ★ m flows (indexed by $k = 1, 2, ...m$)
  - ★ Flow k has size f_k, source node s_k, destination d_k
  - ★ It costs c_{ij} for a unit of flow to use directed edge (i, j)
  - The capacity of directed edge is b_{ij}
- Find the directed edges that **each flow** should use in order that
  - The total cost is minimised
  - The entire flow must use only one path
  - ★ The *total flow* on a directed edge does not exceed its capacity

Digression（离题）: Integral（积分，完整的） versus continuous traffic engineering

There are two versions of traffic engineering problem

The *integral* version where each flow must use only one path, i.e. all packets in a flow must use the same path

- In order to ensure that the packets use a certain path, you can use source routing (available in IP version 6) or MPLS (multi-protocol label switching - covered in COMP9332)

The *continuous* version where each flow may use multiple paths, e.g. the one described on pages 6 – 7 of this lecture.

- In order to split the flow, a classifier will be required at the router to send packets on different paths

We will see how we can formulate the integral traffic engineering problem

Traffic engineering IP formulation

Decision variables: m sets of decision variables, one for *each flow*

$$x_{ijk} = \begin{cases} 1 & \text{if flow } k \text{ uses directed edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

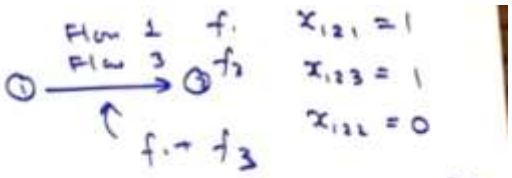
The flow on directed edge (i,j) will be

$$\sum_{k=1}^{m} f_k x_{ijk}$$

Ex: $m = 3$. Flows 1 and 3 use edge (1,2) but flow 2 doesn't.

Total flow in edge (1,2) = $f_1 + f_3$

$\sum_{k=1}^{m} f_k x_{12k} = f_1 \times \underbrace{x_{121}}_{=1} + f_2 \times \underbrace{x_{122}}_{=0} + f_3 \times \underbrace{x_{123}}_{=1} = f_1 + f_3$



$\sum_{k=1}^{m} f_k x_{12k} = f_1 \overset{=1}{x_{121}} + f_2 \overset{=0}{x_{122}} + f_3 \overset{=1}{x_{123}}$

$= f_1 + f_3$

The problem formulation is

$$\min \sum_{(i,j)\in E} \sum_{k=1}^{m} c_{ij} f_k x_{ijk}$$

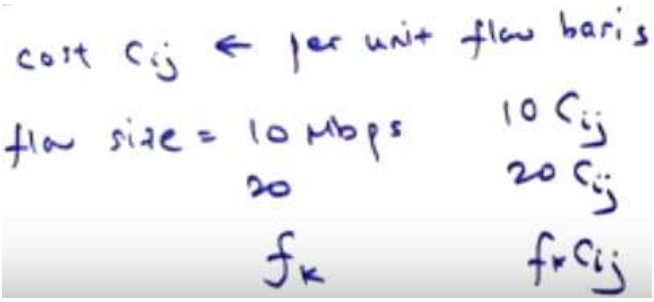
subject to

$$\sum_{j:(i,j)\in E} x_{ijk} - \sum_{j:(j,i)\in E} x_{jik} = \begin{cases} 1 & \text{if } i = s_k \\ 0 & \text{if } i \in N - \{s_k, d_k\} \\ -1 & \text{if } i = d_k \end{cases} \quad k = 1, ..., m \ (*)$$

$$\sum_{k=1}^{m} f_k x_{ijk} \leq b_{ij} \quad \text{for all } (i,j) \in E \ (**)$$

$$x_{ijk} \in \{0,1\} \text{ for all } (i,j) \in E, \ k = 1, ..., m$$

$(*)$ – One set of flow balance constraint per flow. Enforces flow k is from s_k to d_k
$(**)$ – Total flow on a link does not exceed its capacity.



Cost 是单位 flow 的 cost

Traffic engineering problem

Also known as

- The multi-commodity flow problem in operations research
- Flow assignment problem

Essence: assign a flow to a path so that performance is met

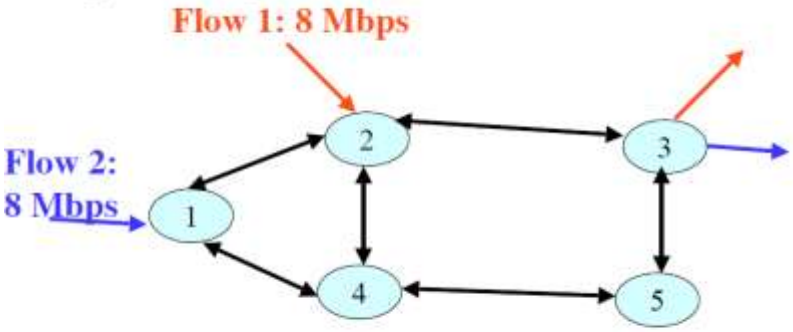
- i.e. routing problem

Many variations possible

- Constraint on the path delay / number of hops
- Constraint on packet loss rate

Network design problem

In flow assignment, we assume the network topology and link capacities are given.



Why should we choose capacity 10Mbps? Why not 100Mbps? Why should we choose to have a link between (2,3) but not (2,5)?

Given

- A set of nodes N
- m flows of size f_k, source s_k, destination d_k
- Minimum network building cost

Design options in network design problems

- Topology: Which directed links to include
- Capacity of the link
- How the flows are routed?

There are a few different network design problems

Different network design problems

Flow Assignment Problem

- Given: flows, topology, capacity
- Find: paths for the flows

Capacity and Flow Assignment Problem

- Given: flows, topology, network cost
- Find: paths for the flows, capacity

Topology, Capacity and Flow Assignment Problem

- Given: flows, network cost
- Find: paths for the flows, capacity, topology

Week 8B: Revision problems

1. A network is represented as a directed graph $G = (N, E)$ where $N = \{1, 2, ..., n\}$ is the set of nodes and E is the set of directed edges. The cost of using link $e_{ij} \in E$ is c_{ij} and the remaining capacity on link e_{ij} is r_{ij}. The propagation delay of link e_{ij} is d_{ij}. A customer of the network wants the network to carry a flow of size b for it. The customer has the following requirements:

   - The flow's source and destination are respectively node $n_1 \in N$ and node $n_2 \in N$
   - The network must provide 2 different paths with for the flow. The flow normally uses only the first path but if it fails, it is switched to the second (or backup) path.
   - Both paths begin at the source and end at the destination.
   - All the links of both path must have at least a capacity b, i.e. links with a residual capacity less than b cannot be used.
   - The total propagation delay in each path must not be greater than $d_{\max}$
   - The two paths must not have any common link.
   - The total cost of the two paths is minimised.

   (a) Formulate an integer programming problem which solves for both paths simultaneously.

   (b) Using the data given below, find the paths for the customer.

   - Number of nodes = 6. The nodes and edges in the network are defined overleaf in AMPL format.
   - The cost, propagation delay and residual bandwidth are given overleaf in AMPL format.
   - Source node = 1. Destination node = 4;
   - $b = 2$.
   - $d_{\max} = 8$.

You will find the following pre-ample useful if you are using AMPL.
In the "mod" file:

```
set NODES; #set of nodes
set EDGES within {i1 in NODES,i2 in NODES: i1 <> i2}; #set of edges
param cost {(i,j) in EDGES};
param delay {(i,j) in EDGES};
param remaining_bandwidth {(i,j) in EDGES};
```

In the "dat" file:

```
set NODES := 1,2,3,4,5,6;
set EDGES := (1,2),(2,1),(2,3),(3,2),(3,4),(4,3),(4,5),(5,4),(5,6),(6,5),
(1,6),(6,1),(2,6),(6,2),(2,5),(5,2),(3,6),(6,3),(3,5),(5,3);

param:    cost    delay  remaining_bandwidth    :=
[1,2]      1        5         9
[2,1]      3        1         6
[2,3]      2        4         4
[3,2]      3        3         3
[3,4]      2        1         4
[4,3]      3        3         7
[4,5]      3        3         4
[5,4]      2        4         4
[5,6]      4        3         5
[6,5]      1        1         6
[1,6]      3        3         3
[6,1]      4        4         4
[2,6]      2        1         5
[6,2]      3        2         6
[2,5]      1        2         7
[5,2]      2        2         9
[3,6]      1        2         3
[6,3]      3        1         4
[3,5]      2        3         1
[5,3]      2        3         3;
```

1. (a) The decision variables are

   - $x_{ij1} = 1$ if the primary path uses link (i, j), otherwise 0
   - $x_{ij2} = 1$ if the backup path uses link (i, j), otherwise 0

   The object is to minimise

   $$\sum_{(i,j)\in E} c_{ij}(x_{ij1} + x_{ij2}) \tag{1}$$

   subject to

   $$\sum_{j:(i,j)\in E} x_{ijk} - \sum_{j:(j,i)\in E} x_{jik} = \begin{cases} 1 & \text{if } i = n_1 \\ 0 & \text{if } i \in N - \{n_1, n_2\} \\ -1 & \text{if } i = n_2 \end{cases} \qquad k = 1, 2$$

   $$\sum_{(i,j)\in E} d_{ij}x_{ij1} \leq d_{\max}$$

   $$\sum_{(i,j)\in E} d_{ij}x_{ij2} \leq d_{\max}$$

   $$(x_{ij1} + x_{ij2})b \leq r_{ij}$$

   $$x_{ij1} + x_{ij2} \leq 1$$

   $$x_{ijk} \in \{0, 1\} \text{ for all } (i, j) \in E, k = 1, 2$$

   (b) The paths are 1−6−5−4 and 1−2−6−3−4. AMPL files are in `disjoint_hw.dat`, `disjoint_hw.mod` and `disjoint_hw_batch`

Week 9A: Optimization (4): Placement problems

A recurrent theme is to use integer programming to make *binary decisions*

Examples of binary decisions

- Week 8A: Grid computing problem
  - Choose a particular grid computing company or not
- Week 8B: Routing of flows
  - Should the flow be routed on a link or not?

Overview

Integer programming for placement problem

- Example: There are a number of potential places that I can put certain devices, what are the best places to put them?

We will study a placement problem in wireless networks

- Placement of wireless access points

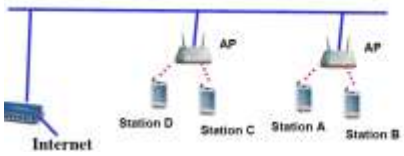
For the revision problem, we will look at the controller placement problem for software-defined networking

Wireless Local Area Networks

Commonly known as Wireless LAN, WiFi etc.

Formal standards in IEEE 802.11, IEEE 802.11a/b/g/n/ac

Infrastructure mode: Wireless Access Points (APs) and Wireless stations

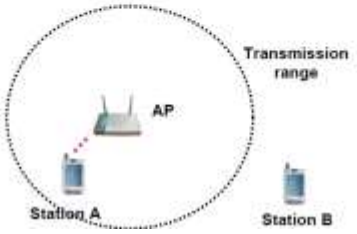


Wireless Access Point Coverage

Due to radio propagation loss and mandated limit on transmission power, wireless access points have only limited coverage

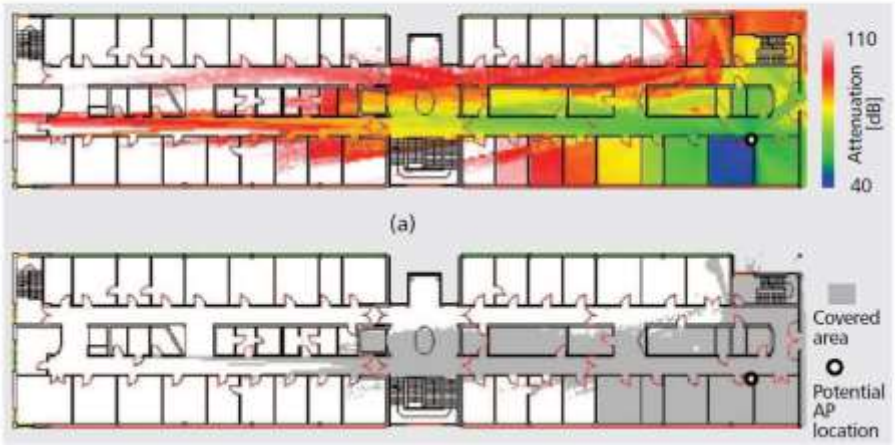
For example, a Cisco Aironet access point has a coverage of 304m when operating outdoor at 11 Mbps

Ideal coverage area is a circle. In the picture below, Station A can talk to the access point but not Station B



Coverage in practice

- Note: High attenuation = Poor signal = Poor coverage
- An area is covered only if the attenuation is smaller than a threshold

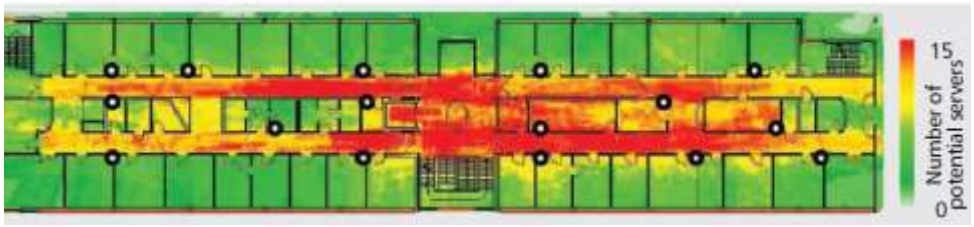


Covering a given area

Multiple access points may be required to cover a given area

Decisions to make

- The number of access points required to cover the given area
- The position of the access points



The coverage problem – definition

Given

- A number of potential access point locations $L_1, L_2, ..., L_p$
- A number of stations $s_1, s_2, s_3, ..., s_n$
- Binary constant δ_{ij} where

$$\delta_{ij} = \begin{cases} 1 & \text{if station } s_i \text{ is covered by an AP at location } L_j \\ 0 & \text{otherwise} \end{cases}$$

See the next page for a graphical explanation for δ_{ij}

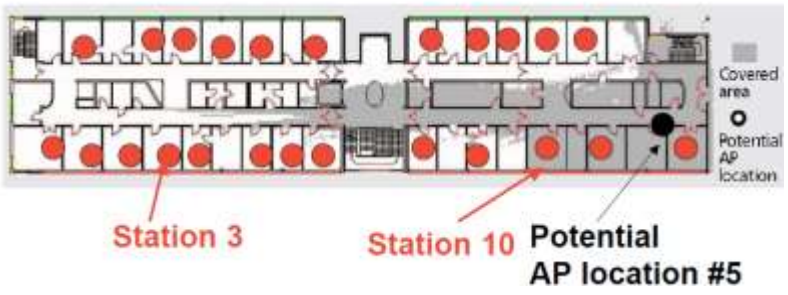
Find the minimum number of access points required so that

- All stations are covered

Explanation of δij

Example: $\delta_{10,5} = 1, \quad \delta_{3,5} = 0$

● = location of station



The coverage problem: Verbal formulation

Decision variables:

$$x_j = \begin{cases} 1 & \text{if an AP is to be installed at location } L_j \\ 0 & \text{otherwise} \end{cases}$$

Integer programming formulation:

$$\min \boxed{\text{The number of access points (An expression in } x_j)}$$

subject to

$$\boxed{\text{Each station is covered (One expression for each station, need } x_j \text{ and } \delta_{i,j})}$$
$$x_j \in \{0, 1\}$$

The coverage problem: Formulation

Decision variables:

$$x_j = \begin{cases} 1 & \text{if an AP is to be installed at location } L_j \\ 0 & \text{otherwise} \end{cases}$$

Integer programming formulation:

$$\min \sum_{j=1}^{p} x_j$$

subject to

$$\sum_{j=1}^{p} \delta_{ij} x_j \geq 1 \quad \forall i = 1, ..., n$$
$$x_j \in \{0, 1\}$$

Week 9A: Revision problems

1. In this question, you will formulate an integer programming problem for the *controller placement problem* in software defined networking (SDN). We briefly discussed this problem in the beginning of the lecture in Week 10, see pages 2-3 of the lecture notes for that week.

   In SDN, there are two types of "boxes": (1) Simple packet switches; and (2) Controllers. At each node of the network, there is a simple packet switch. However, we do not need to have a controller at each network node. We only need to place controllers at some of the network nodes, and one controller can be used to control multiple switches. Since a controller needs to communicate with those switches that it controls, the communication delay between a controller and a switch under its control must not be too big.

   For the controller placement problem, you are given:

   - A network (N, E) where N is the set of nodes and E is the set of edges. There is a simple packet switch at each node in N. We assume there are n nodes and they are indexed by $1, 2, ..., n$.
   - The commuication delay d_{ij} between nodes i and j in the network, for all $i, j \in N$. This means that if a controller is placed at node i, then the communication delay between this controller and a switch at node j is d_{ij}.
   - The maximum allowable delay D between a controller and the switches that it controls.

   The requirements are:

   - Each switch must be controlled by a controller.
   - The delay between a controller and any switch that it controls cannot exceed D.

   Formulate an integer programming to minimise the number of controllers required subject to the above requirements.

1. The decision variables are

   - $$y_i = \begin{cases} 1 & \text{if a controller is to be placed at node } i \\ 0 & \text{otherwise} \end{cases}$$

   - $$x_{ij} = \begin{cases} 1 & \text{if the controller at node } i \text{ is to control the switch at node } j \\ 0 & \text{otherwise} \end{cases}$$

   The optimisation problem is:

$$\min \sum_{i=1}^{n} y_i$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \ \forall j = 1, .., n \quad (1)$$

$$x_{ij} d_{ij} \leq D y_i \ \forall i, j = 1, .., n \quad (2)$$

$$x_{ij} \in \{0, 1\} \quad (3)$$

$$y_i \in \{0, 1\} \quad (4)$$

Equation 1 enforces the constraint that a switch is associated with only one controller. Equation 2 ensures that: (i) If $y_i = 0$, i.e. there is no controller at node i, then $x_{ij} = 0$; (ii) If $y_i = 1$, then any switch that is connected to the controller at node i must have a communication delay of less than D. Equation 2 has combined (i) and (ii) into a set of inequalities. Alternatively, you can use

$$x_{ij} \leq y_i \ \forall i, j = 1, .., n \quad (5)$$

$$x_{ij} d_{ij} \leq D \ \forall i, j = 1, .., n \quad (6)$$

to separately enforce the conditions (i) and (ii).

Week 9B: Optimization (5): Power of binary variables

A recurrent theme is to use integer programming to make *binary decisions*

Examples of binary decisions

- Week 8A: Grid computing problem
  - Choose a particular grid computing company or not
- Week 8B: Routing of flows
  - Should the flow be routed on a link or not?
- Week 9A: Placement problem
  - Should a location be chosen or not?

Overview

Not only for making yes-or-no type of decisions, binary variables can be used to capture many other requirements

- Restricted range of values
- Either-or constraints
- If-then constraints
- Piecewise linear functions

Restricted range of values

Some variables can only take certain values

- E.g. network links can only be of capacity 155 Mbps, 466 Mbps, 622 Mbps, etc

If decision variable x can only take values from $\{a_1, a_2, ..., a_m\}$, this can be modeled by using an additional set of binary decision variables

$$y_i = \begin{cases} 1 & \text{if } a_i \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

Then, the above requirement can be captured by

$$\begin{aligned} x &= \sum_{i=1}^{m} a_i y_i \\ \sum_{i=1}^{m} y_i &= 1 \\ y_i &\in \{0, 1\} \end{aligned}$$

E.g. if $a_1 = 155, a_2 = 466, a_3 = 622$, we have

- $y_1 = 1 \Rightarrow y_2 = y_3 = 0 \Rightarrow x = 155$
- $y_2 = 1 \Rightarrow y_1 = y_3 = 0 \Rightarrow x = 466$
- $y_3 = 1 \Rightarrow y_1 = y_2 = 0 \Rightarrow x = 622$

Either-or constraints

A Cloud computing service provider offers 3 different packages with different speed and cost for each package. You can buy any cycles from any package but the deal requires that

- # cycles from Package 1 + # cycles from Package 2 $\geq$ 10000, or,
- # cycles from Package 2 + # cycles from Package 3 $\geq$ 50000
  - At least one of these two inequalities must hold, but not necessarily both

Let w_i = number of cycles to be bought from Package i

The above requirement can be captured by using an additional binary decision variable p

$$\begin{aligned} w_1 + w_2 &\geq 10000p \\ w_2 + w_3 &\geq 50000(1-p) \\ p &\in \{0,1\} \\ w_i &\geq 0, \quad i = 1,2,3 \end{aligned}$$

Case 1: $p = 0$, we have

$$\begin{aligned} w_1 + w_2 &\geq 0 \leftarrow \text{Trivially satisfied} \\ w_2 + w_3 &\geq 50000 \\ w_i &\geq 0, \quad i = 1,2,3 \end{aligned}$$

Case 2: $p = 1$, we have

$$\begin{aligned} w_1 + w_2 &\geq 10000 \\ w_2 + w_3 &\geq 0 \leftarrow \text{Trivially satisfied} \\ w_i &\geq 0, \quad i = 1,2,3 \end{aligned}$$

In general, if one of the following two constraints must be satisfied

$$\begin{aligned} \sum_{i=1}^{n} a_{1,i} x_i &\geq b_1 \\ \sum_{i=1}^{n} a_{2,i} x_i &\geq b_2 \end{aligned}$$

where $a_{j,i}$ are given parameters, $x_i (\geq 0)$ are decision variables, b_j's are constants, then the either-or constraints can be modelled by

$$\begin{aligned} \sum_{i=1}^{n} a_{1,i} x_i &\geq b_1 p \\ \sum_{i=1}^{n} a_{2,i} x_i &\geq b_2 (1-p) \\ p &\in \{0,1\} \end{aligned}$$

If-then constraints

We may want to impose if-then constraints, e.g.

$$\text{if } x_1 + x_2 > 1, \text{ then } y \geq 4$$

where x_1, x_2 are binary variables, and $0 \leq y \leq 10$

The above if-then constraint can be captured by using an additional binary decision variable p

$$\begin{aligned} x_1 + x_2 - 1 &\leq 1 - p \\ -y + 4 &\leq 4p \\ p &\in \{0, 1\} \end{aligned}$$

To understand how this works, consider the two cases:

- Case 1: If $x_1 + x_2 > 1$ holds
  - Since $x_1 + x_2 > 1$, $x_1 + x_2 - 1 > 0$
  - Since p can only be 1 or 0, the inequality constraint $x_1 + x_2 - 1 \leq 1 - p$ forces p to be 0
  - Since $p = 0$, from the inequality constraint $-y + 4 \leq 4p$, we have $y \geq 4$ which is the condition that we want to impose when $x_1 + x_2 > 1$ holds
- Case 2: If $x_1 + x_2 > 1$ does not hold
  - In this case, since $x_1 + x_2 - 1 \leq 0$, p can be either 0 or 1
  - If $p = 0$, the inequality constraint $-y + 4 \leq 4p$ becomes $y \geq 4$
  - If $p = 1$, the inequality constraint $-y + 4 \leq 4p$ becomes $y \geq 0$
  - Thus, p can be chosen such that there is no restriction on the value of y

In general, the if-then constraint

$$\text{if } f(x_1, x_2, \ldots, x_n) > 0, \text{ then } g(x_1, x_2, \ldots, x_n) \geq 0$$

can be modeled by

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) &\leq M_1(1 - p) \\ -g(x_1, x_2, \ldots, x_n) &\leq M_2\, p \end{aligned}$$

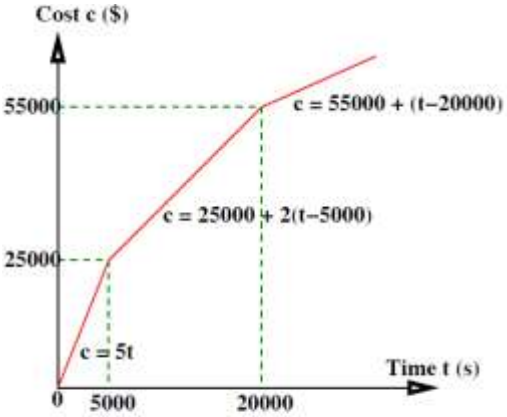
where p is a binary variable, M_1 and M_2 are constants chosen large enough such that $f(x_1, x_2, \ldots, x_n) \leq M_1$ and $-g(x_1, x_2, \ldots, x_n) \leq M_2$ hold for all possible choices of $x_1, x_2, \ldots, x_n$

Piecewise linear functions

We can use binary variables to model piecewise linear functions

Example: A Cloud computing service provider may use a progressive charging scheme

- 5 dollars/sec for the first 5,000 sec
- 2 dollars/sec for the next 15,000 sec
- 1 dollar/sec thereafter



- Decision variables

$$y_i = \begin{cases} 1 & \text{if segment } i \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

- Segment 1: $0 \leq t \leq 5000$, cost = $5t$
- Segment 2: $5000 \leq t \leq 20000$, cost = $2t$ + 15000
- Segment 3: $20000 \leq t$, cost = t + 35000

We have

- $y_1 = 1 \Rightarrow 0 \leq t \leq 5000$ and cost = $5t$
- $y_2 = 1 \Rightarrow 5000 \leq t \leq 20000$ and cost = $2t$ + 15000
- $y_3 = 1 \Rightarrow 20000 \leq t$ and cost = t + 35000
- $y_1 + y_2 + y_3 = 1$

We can rewrite these as

- $0 \leq ty_1 \leq 5000y_1$
- $5000y_2 \leq ty_2 \leq 20000y_2$
- $20000y_3 \leq ty_3$
- cost = $y_1(5t) + y_2(2t + 15000) + y_3(t + 35000)$
- $y_1 + y_2 + y_3 = 1$

Problem: non-linear constraints

Define $t_i = ty_i$ for $i = 1, 2, 3$

- cost = $5t_1 + 2t_2 + 15000y_2 + t_3 + 35000y_3$
- $0 \leq t_1 \leq 5000y_1$
- $5000y_2 \leq t_2 \leq 20000y_2$
- $20000y_3 \leq t_3 \leq My_3$
- $y_1 + y_2 + y_3 = 1$
- $t = t_1 + t_2 + t_3$

Note

- t_i is non-zero if the corresponding $y_i = 1$
- M is a sufficiently large number to enforce

$$y_3 = 0 \Rightarrow t_3 = 0 \quad \text{and} \quad t_3 \geq 20000 \Rightarrow y_3 = 1$$

- This is a non-standard expression
- An alternative expression can be found in Winston Chapter 9

Integer programming and optimization: Summary

What you have learnt

- How to formulate integer programming problems
- How to solve them using AMPL
- Examples of using integer programming for network design and analysis

There are a lot more to learn but this will give you a starting point ...

Week 9B: Revision problems

1. A carrier is planning to build a network between 4 cities, which are labelled as Cities 1,2,3 and 4. The amount of traffic flowing between these cities are given in Table 1. To build this network, the carrier can choose between three different types of links with different cost-capacity combinations, as showed in Table 2. Note that:

   - Each link is directed, i.e. it only carries traffic in one direction.
   - The decision to build links (i, j) and (j, i) are independent. In other words, if the carrier chooses to build link (i, j), it does *not* have to build link (j, i).
   - Each link, if built, must have sufficient capacity to carry the traffic assigned to it.
   - Each flow, i.e. the traffic between a pair of cities, can only use one path.

   The carrier aims to build a network with the minimum cost. Formulate an integer programming problem and solve it using AMPL/CPLEX to determine:

   (a) Which links the carrier should build and what type should it be?

   (b) How the traffic is to be routed in the network?

		To			
		City 1	City 2	City 3	City 4
From	City 1	–	1	2	3
	City 2	3	–	2	1
	City 3	2	4	–	2
	City 4	1	5	3	–

Table 1: Amount of traffic between cities.

Type of link	Capacity	Cost
1	10	10
2	20	18
3	30	25

Table 2: Capacity and cost for different types of links.

Hint

Define decision variables

$$x_{ijuv} = \begin{cases} 1 & \text{if the traffic from Node } u \text{ to Node } v \text{ uses link } (i, j) \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ijk} = \begin{cases} 1 & \text{if a link of type } k \text{ is to be built from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

If all y_{ij1}, y_{ij2} and y_{ij3} are zero, then this means a link won't be built from node i to node j. Impose constraints that if there isn't a link, no traffic can use that link etc.

1. Following the hint given in the question, we define the following decision variables:

- $$x_{ijuv} = \begin{cases} 1 & \text{if the traffic from Node } u \text{ to Node } v \text{ uses link } (i,j) \\ 0 & \text{otherwise} \end{cases}$$
- $$y_{ijk} = \begin{cases} 1 & \text{if a link of type } k \text{ is to be built from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

In addition, we define the following:

- $N = \{1, 2, 3, 4\}$ (i.e. the set of nodes)
- $E = \{(i,j) \in N \times N : i \neq j\}$ (i.e. the set of all possible links)
- c_k = the cost of link of type k (as given in Table 2)
- b_k = the capacity of link of type k (as given in Table 2)
- f_{uv} = the traffic demand from city u to city v (as given in Table 1)
- An indicator function

$$\delta_{pq} = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases}$$

The optimisation problem is:

$$\min \sum_{(i,j) \in E} \sum_{k=1}^{3} y_{ijk} c_k$$

subject to

$$\sum_{j:(i,j) \in E} x_{ijuv} - \sum_{j:(j,i) \in E} x_{jiuv} = \delta_{iu} - \delta_{iv} \quad \forall i \in N, (u,v) \in E \tag{1}$$

$$\sum_{(u,v) \in E} x_{ijuv} f_{uv} \leq \sum_{k=1}^{3} b_k y_{ijk} \quad \forall (i,j) \in E \tag{2}$$

$$\sum_{k=1}^{3} y_{ijk} \leq 1 \quad \forall (i,j) \in E \tag{3}$$

$$x_{ijuv} \in \{0,1\} \tag{4}$$

$$y_{ijk} \in \{0,1\} \tag{5}$$

Note:

- Equation 1 is equivalent to

$$\sum_{j:(i,j)\in E} x_{ijuv} - \sum_{j:(j,i)\in E} x_{jiuv} = \begin{cases} 1 & \text{if } i = u \\ 0 & \text{if } i \neq u, v \\ -1 & \text{if } i = v \end{cases}$$

  The expression $\delta_{iu} - \delta_{iv}$ evaluates to the expression on the right-end-side of the above equation.

- Equation 2 ensures: (1) There is enough capacity in the link if it is built; and (2) If a link is not built (i.e. all $y_{ijk} = 0$ for $k = 1, 2, 3$), no traffic is routed in (i, j) (i.e. $x_{ijuv} = 0$).
- Equation 3 ensures that only one type of link is chosen.

The AMPL code is given in `carrier.mod`, `carrier.dat` and `carrier_batch`.

The optimised network has a cost of 58 units. It has 4 links of type 1 and they are (1,4), (2,3), (3,1), (3,2). It also has a link of type 2 for link (4,3). No links of type 3 is used. Traffic from node 1 to node 2 follows the path (1,4),(4,3),(3,2). Traffic from node 2 to node 3 follows the path (2,3). These can be read from x_{ijuv}. See the file `carrier-solution` where you can read out the paths for the other traffic demands.

Note that there are multiple networks (with different links and routes) which also gives the minimum cost of 58 units. These alternatives are also acceptable.