COMP9334
Capacity Planning for Computer Systems and Networks

Week 2: Operational Analysis and

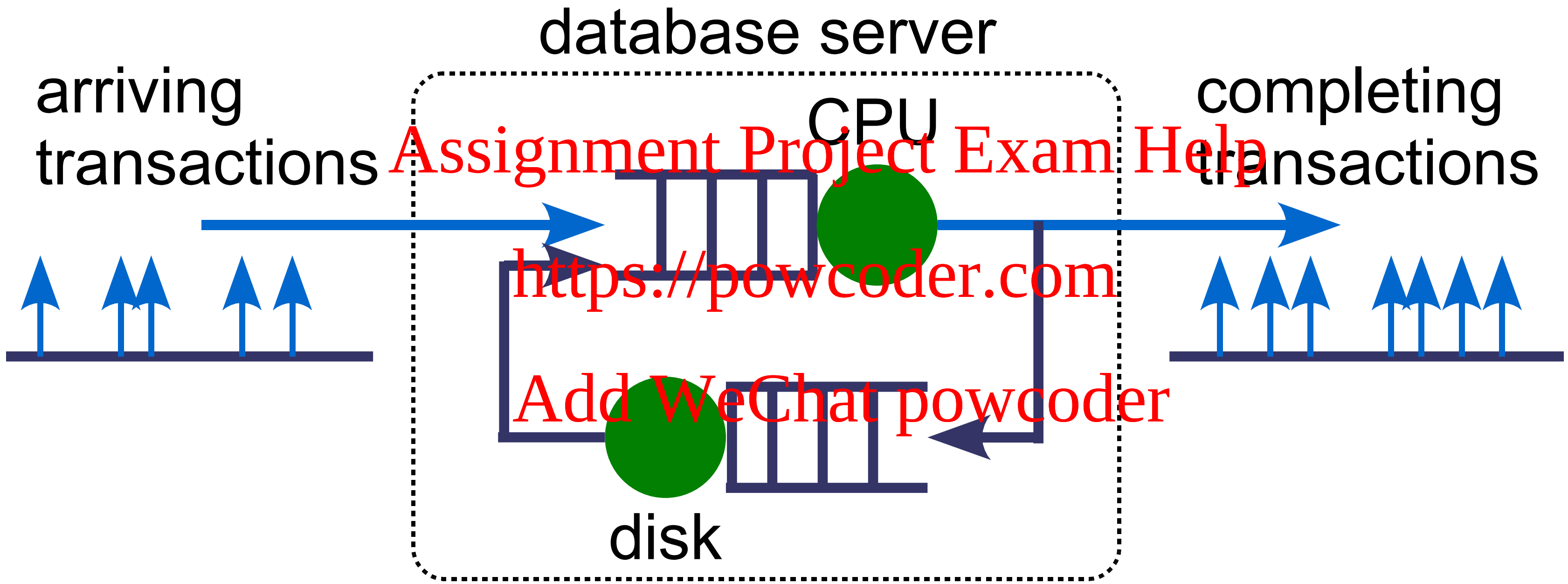
Workload Characterisation

Last lecture

- Modelling of computer systems using Queueing Networks
  - Open networks
  - Closed networks

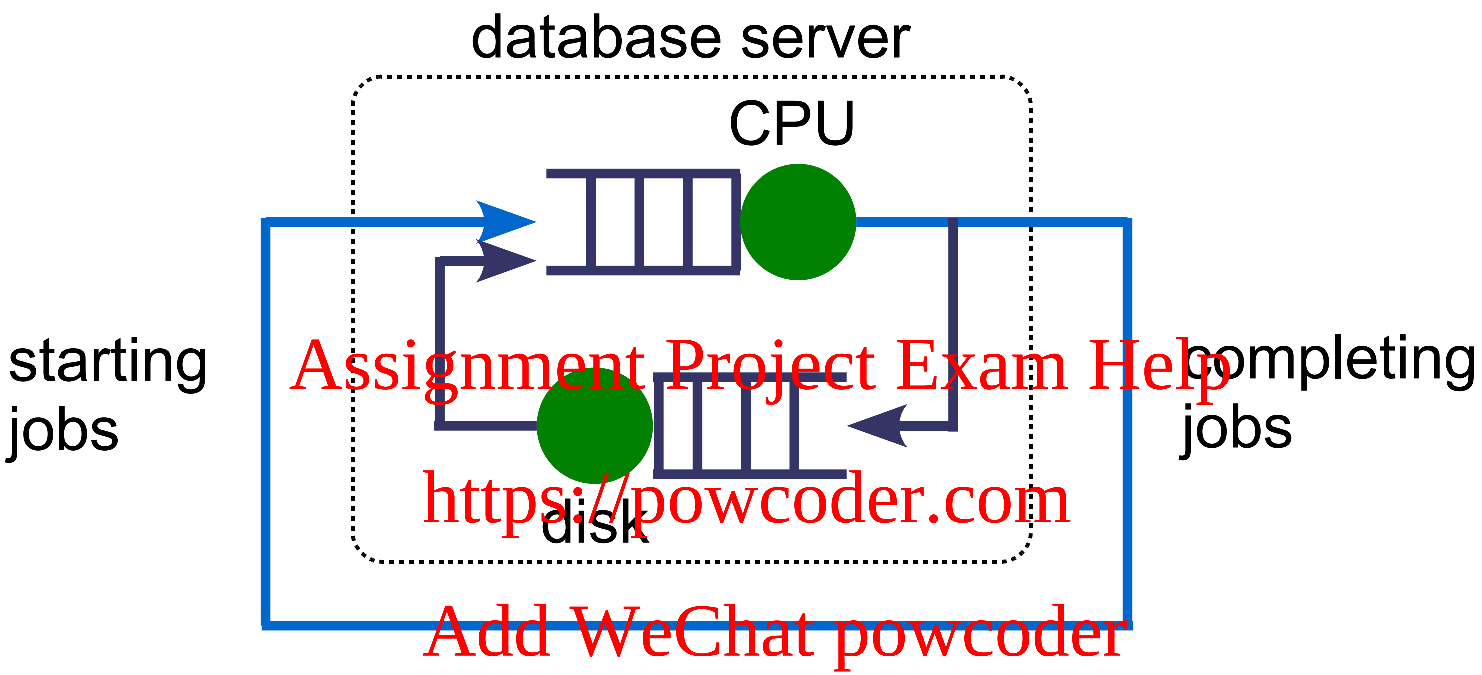
Open networks

Example: The server has a CPU and a disk.



A transaction may visit the CPU and disk multiple times.
An open network is characterised by external transactions.

Closed queuing networks



Closed queueing networks model
- Running batch jobs overnight
- Once a job has completed, a new job starts.

Good performance means high throughput.
#jobs in the system = multi-programming level

This lecture

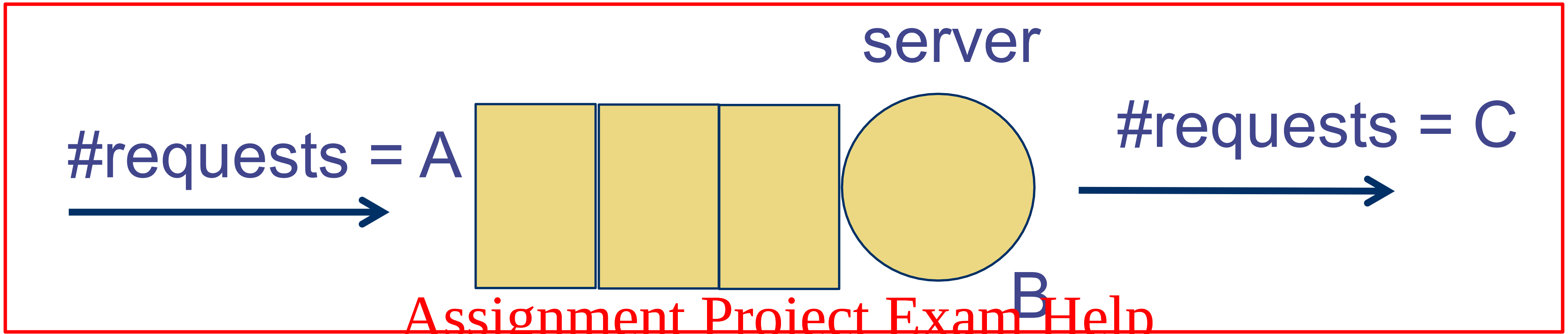
- The basic performance metrics
  - Response time, Throughput, Utilisation etc.
- Operational analysis
  - Fundamental Laws relating the basic performance metrics
  - Bottleneck and performance analysis
- Workload characterisation
  - Poisson process and its properties

Operational analysis (OA)

- "Operational"
  - Collect performance data during day-to-day operation
- Operation laws
- Applications:
  - Use the data for building queueing network models
  - Perform bottleneck analysis
  - Perform modification analysis

- iostat

Single-queue example (1)



In an observational period of T, server busy for time B

A requests arrived, C jobs completed

A, B and C are basic measurements

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 =
    - Utilisation =
    - Arrival rate =
    - Output rate =

Utilisation 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?
- Utilisation law is an example of operational law.

Application of OA

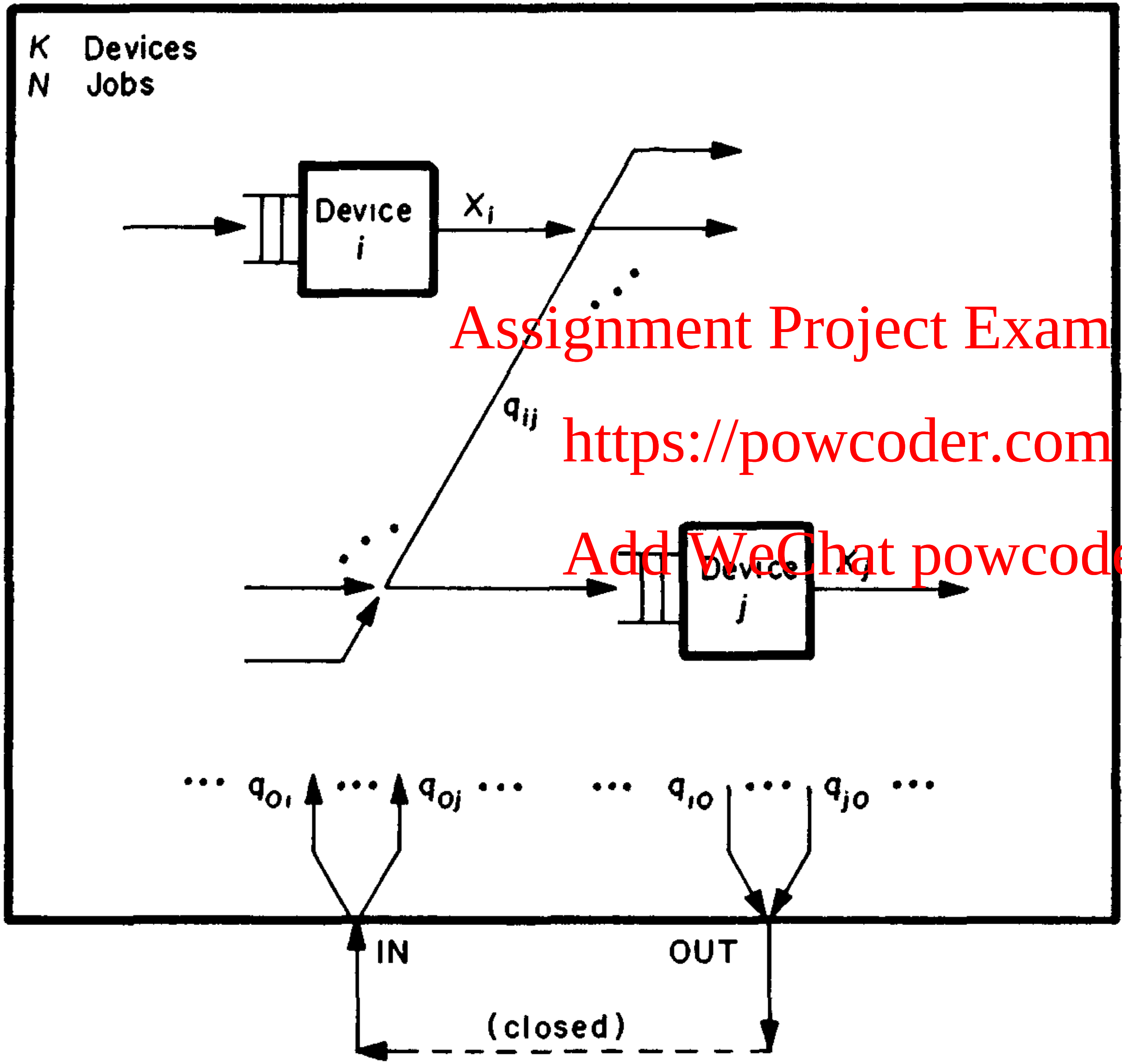
- Don't have to measure every operational quantities
  - Measure B to deduce U - don't have to measure U
- Consistency checks
  - If $U \neq S X$, something is wrong

- Operational laws can be used for performance analysis
  - Bottleneck analysis (today)
  - Mean value analysis (Later in the course)

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"

OA for Queueing Networks (QNs)



The computer system has K devices, labelled as 1,…,K.

The convention is to add an additional device 0 to represent the outside world.

OA for QNs (cont' d)

- We measure the basic operational quantities for each device (or other equivalent quantities) over a time of T
  - A(j) = Number of arrivals at device j
  - B(j) = Busy time for device j
  - C(j) = Number of completed jobs for device j
- In addition, we have
  - A(0) = Number of arrivals for 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 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) = $\boxed{}$

  - V(j) = Visit ratio of device j
    - = Number of times a job 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 two disk accesses to be completed. One disk access takes 20ms and the other takes 30ms.
    - D(j) = 50ms.
    - What are V(j) and S(j)?
      - Recall that S(j) = mean service time of device j
    -
  - Service demand D(j) = V(j) S(j)

Service demand law (1)

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)?

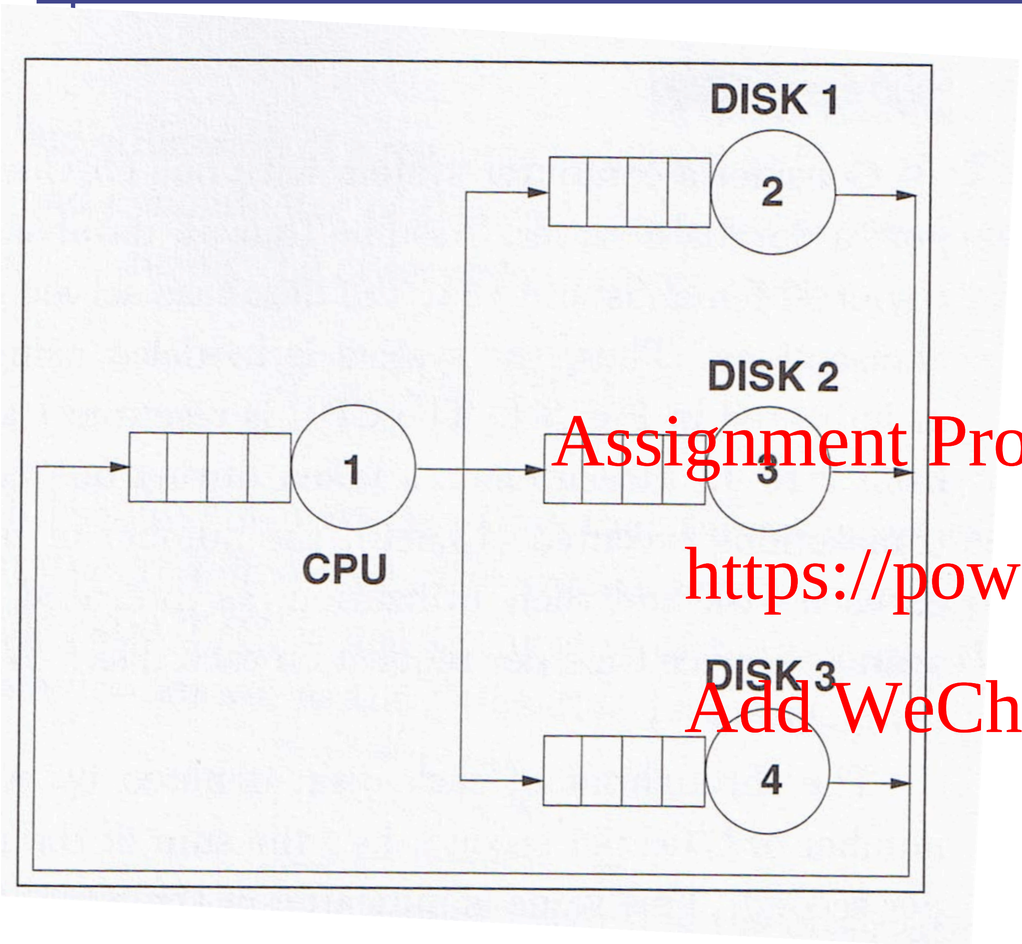
Service demand law $$D(j) = \frac{U(j)}{X(0)}$$

Service demand law (2)

- 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 today

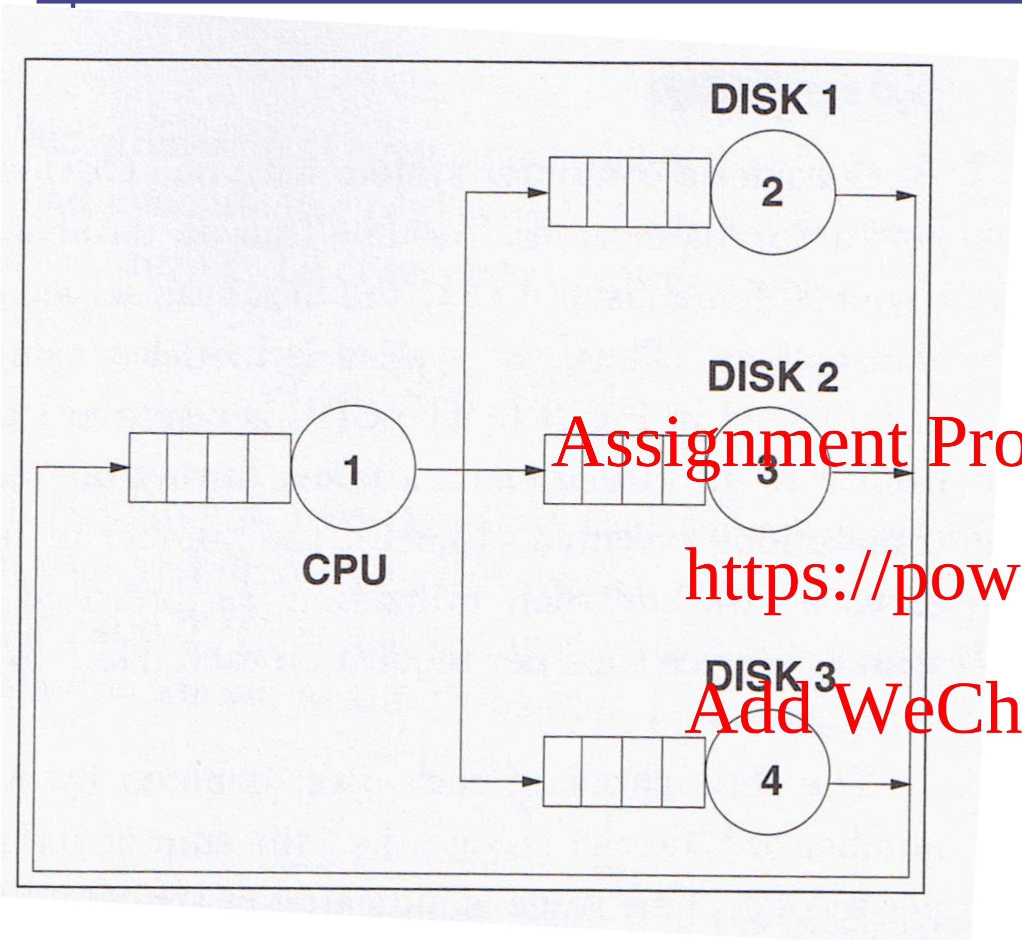
Server example exercise



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?

Server example solution



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		

Service time = U2/X2

System throughput

Service demand

Visit ratio

Little's law (1)

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

Little's Law (2)

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

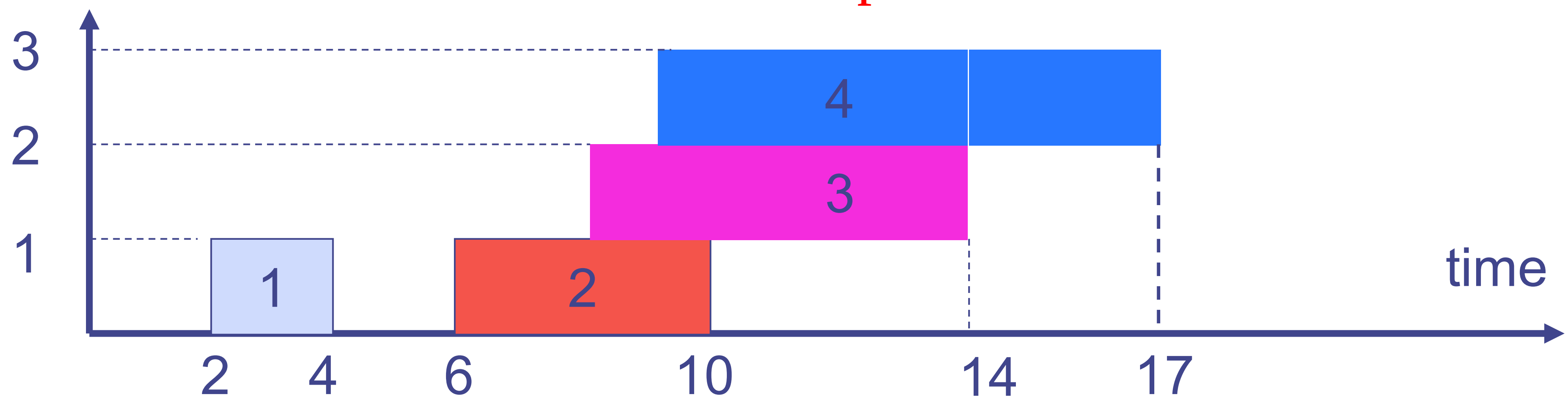
$$Navg = X * Ravg$$

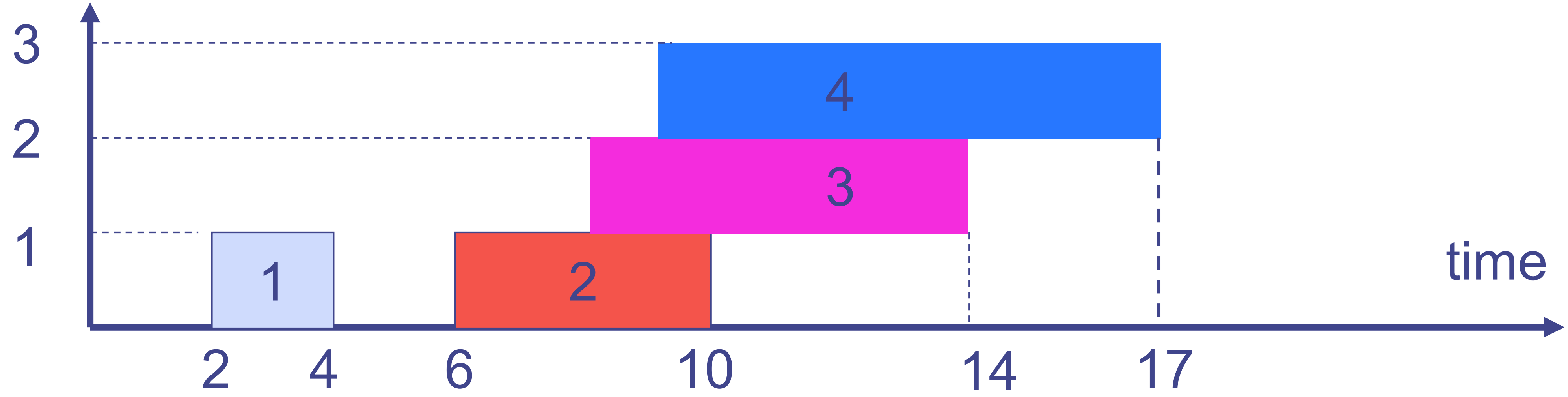
We will argue the validity of Little's Law using a simple example.

Consider the single sever queue example from Week 1

Job 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 jobs, i.e. width of each block = response time of the job

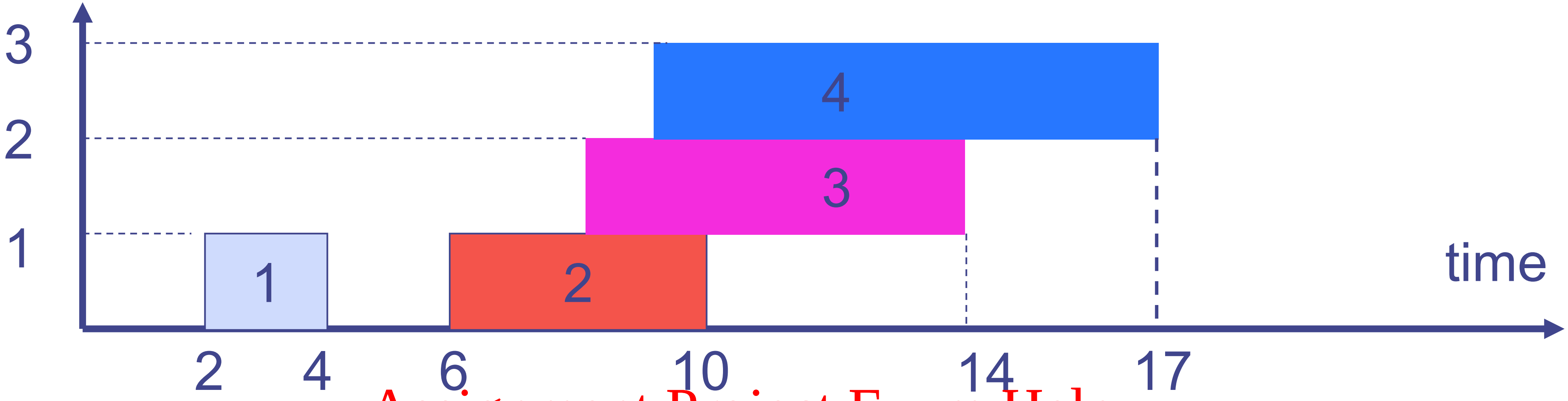




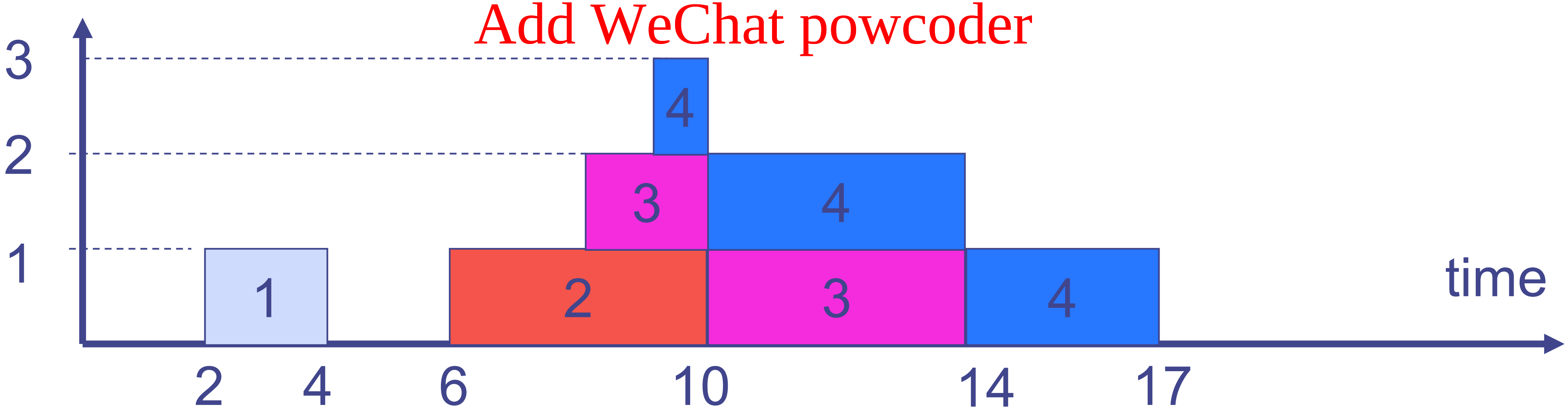
Assuming that in the measurement time interval [0,20] these 4 jobs arrive arrive and depart from this device, i.e. the device is in equilibrium.

Total area of the blocks
= Response time of job 1 + Response time of job 2 + Response time of job 3 + Response time of job 4
= Average response time over the measurement interval * Number of jobs departing over the measurement interval

This is one interpretation. Let us look at another.

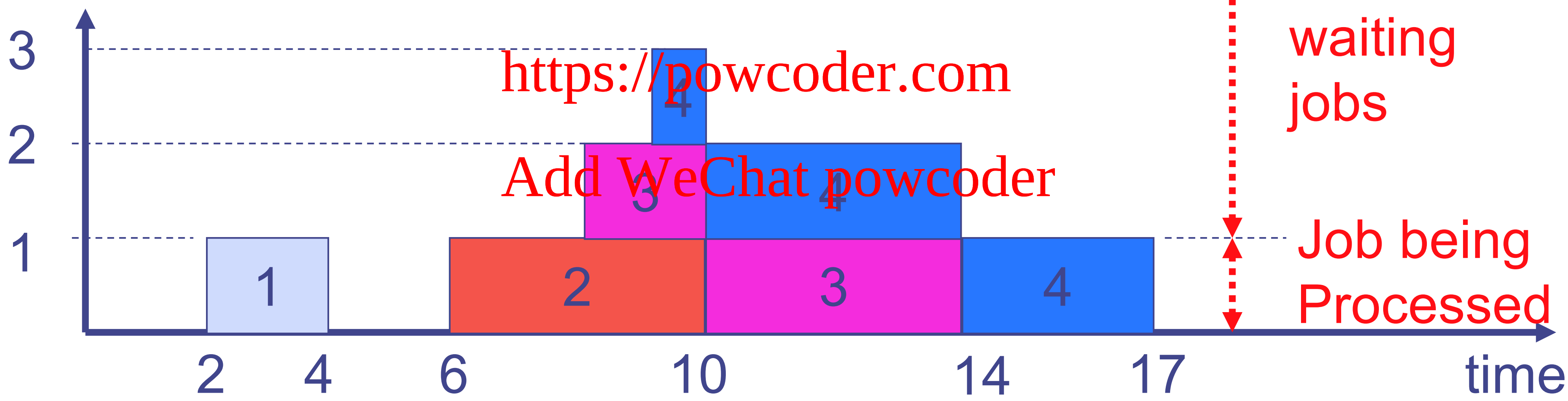


Let us assume these blocks are "plastic" and let them fall to the ground. Like this.



There is an interpretation of the height of the graph.

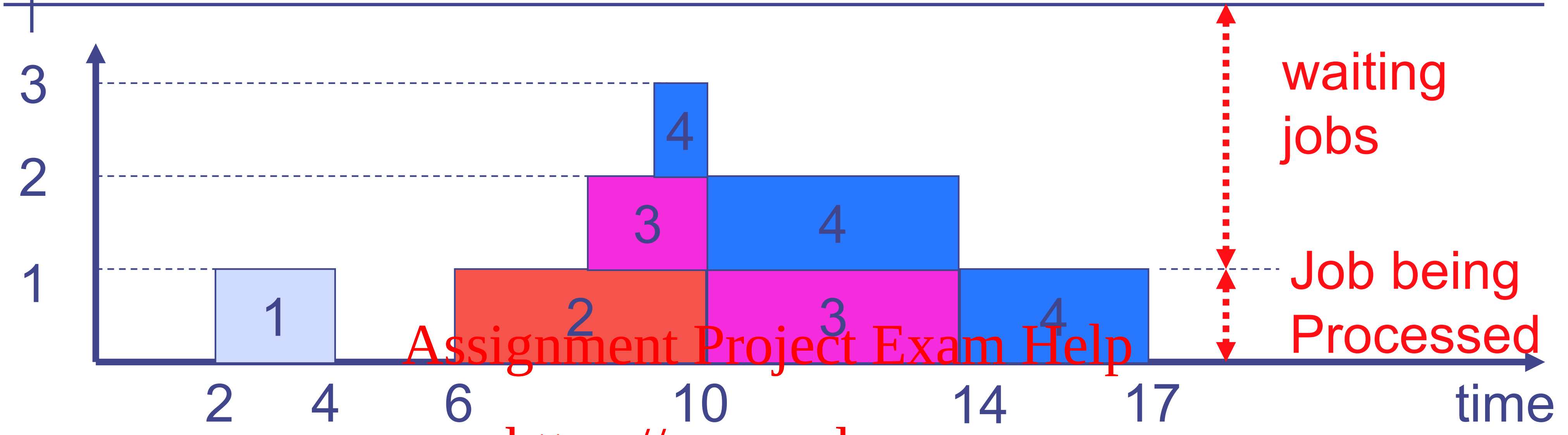
Job index	Arrival time	Service time
1	2	2
2	6	4
3	8	4
4	9	3



Interpretation: Height of the graph = number of jobs in the device

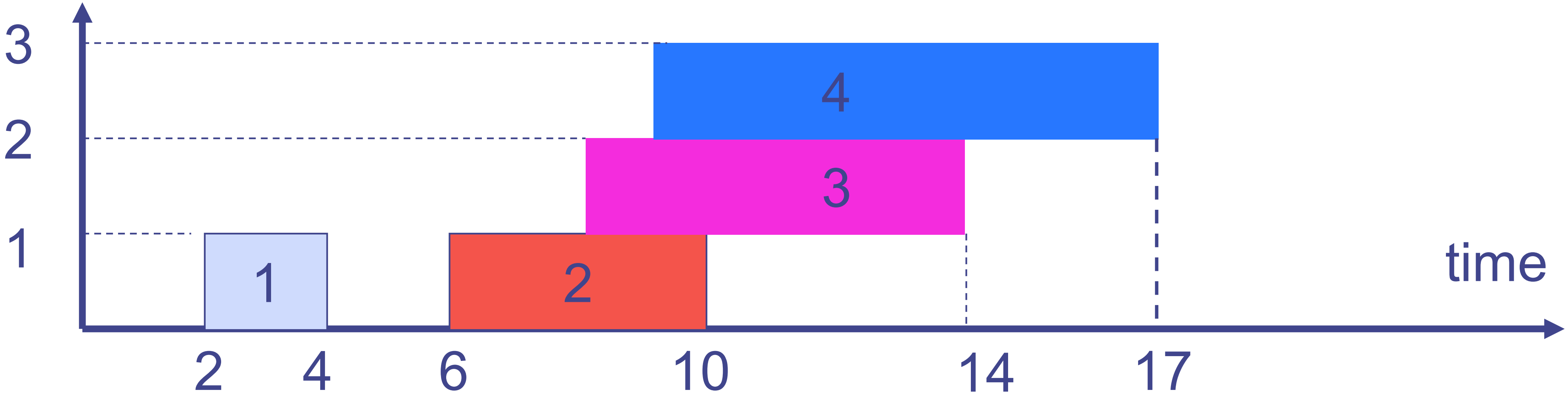
E.g. Number of jobs in [9,10] = 3

E.g. Number of jobs in [11,12] = 2 etc.



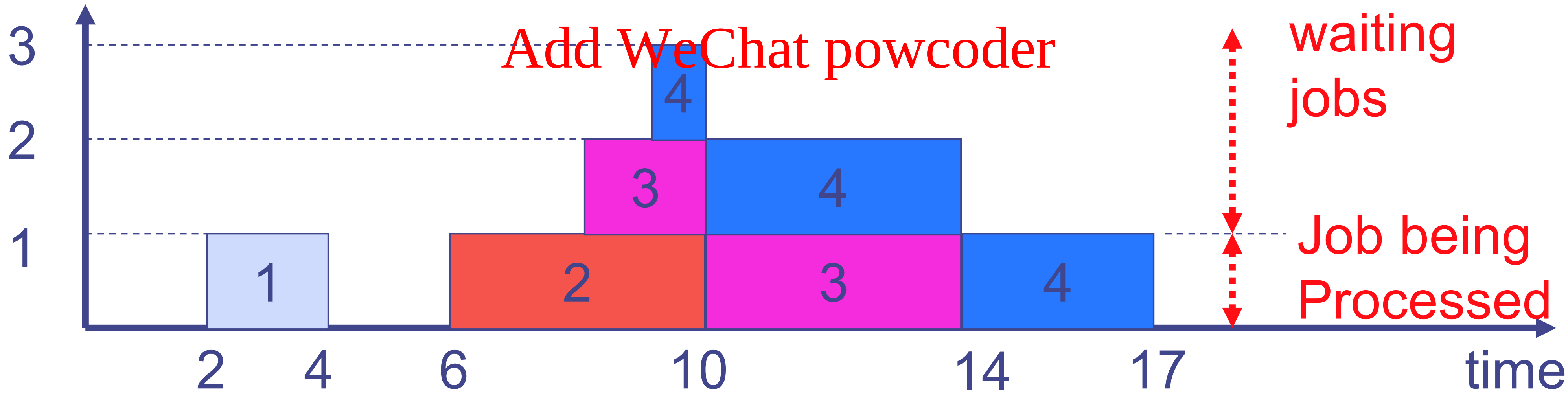
Again, consider the measurement time interval of [0,20].

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]
= #jobs in [0,1] + #jobs in [1,2] + … + #jobs in [19,20]
= Average number of jobs in [0,20] * 20



Area = Average response time over [0,T] *

Number of jobs leaving in [0,T]



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

Deriving Little's Law

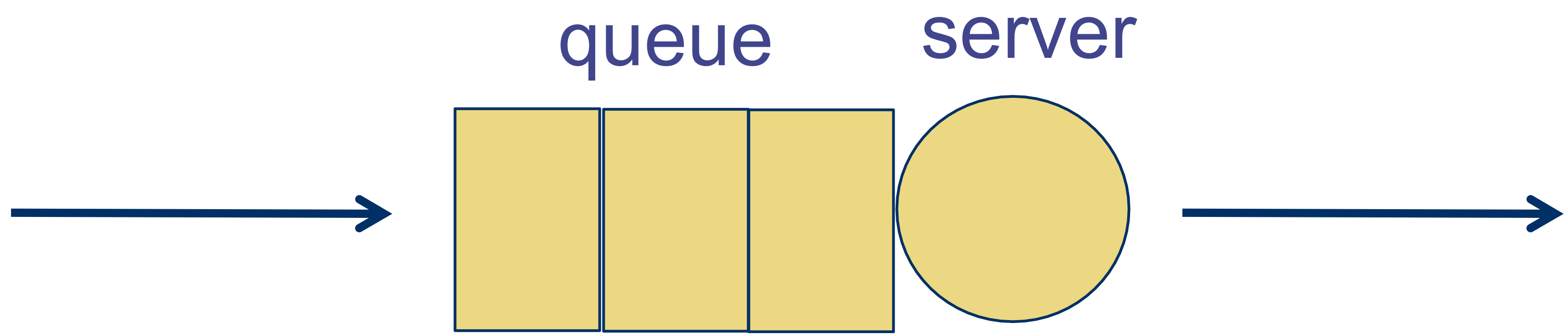
Area = Average response time of all jobs *
Number of jobs leaving in [0,T] (Interpretation #1)
= Average number of jobs in [0,T] * T (Interpretation #2)

Since Number of jobs leaving in [0,T] / T
= Device throughput in [0,T]

We have Little's Law.

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

Using Little's Law (1)



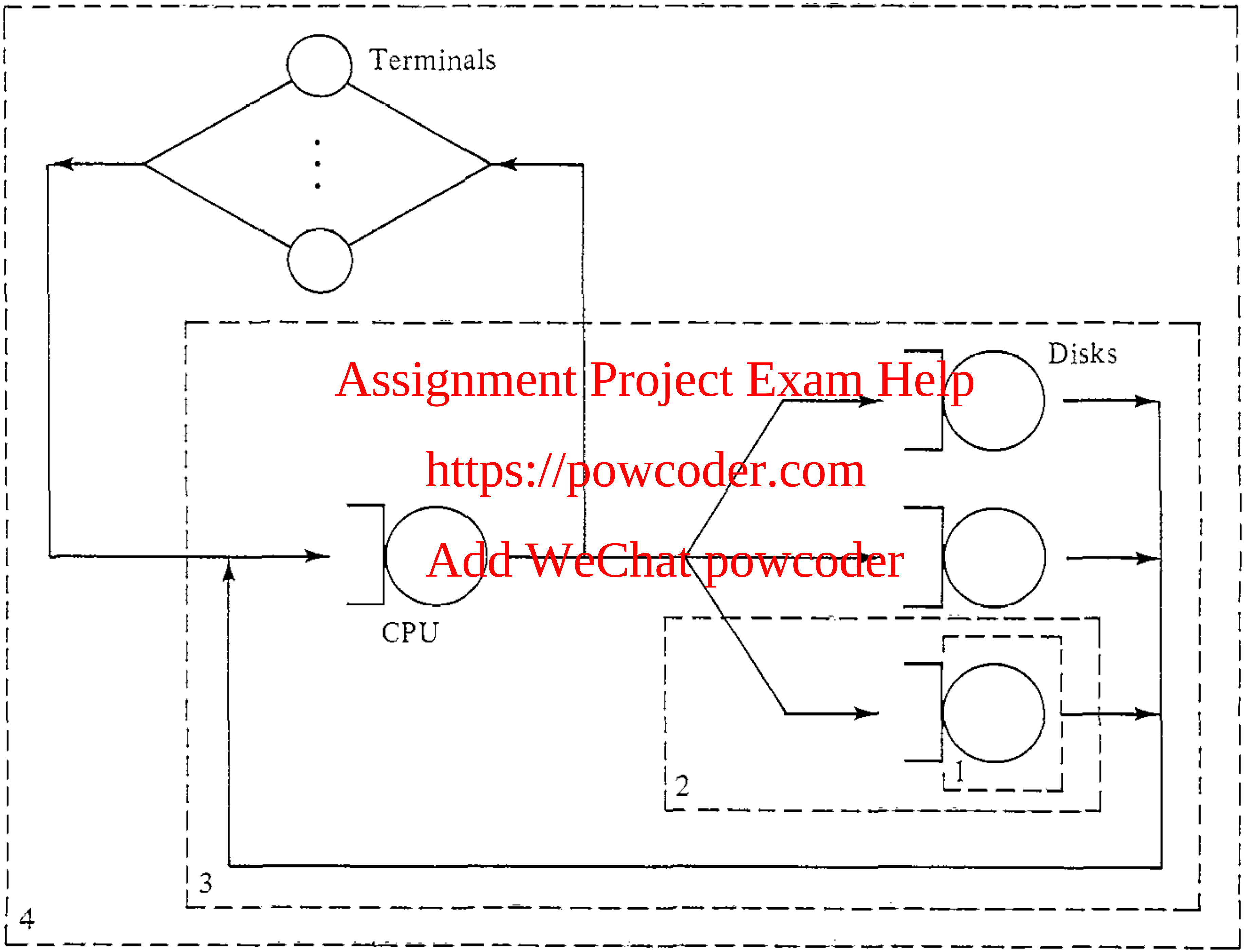
- A device consists of a server and a queue
- The device completes on average 8 requests per second
- On average, there are 3.2 requests in the device
- What is the response time of the device?

Intuition of Little's Law

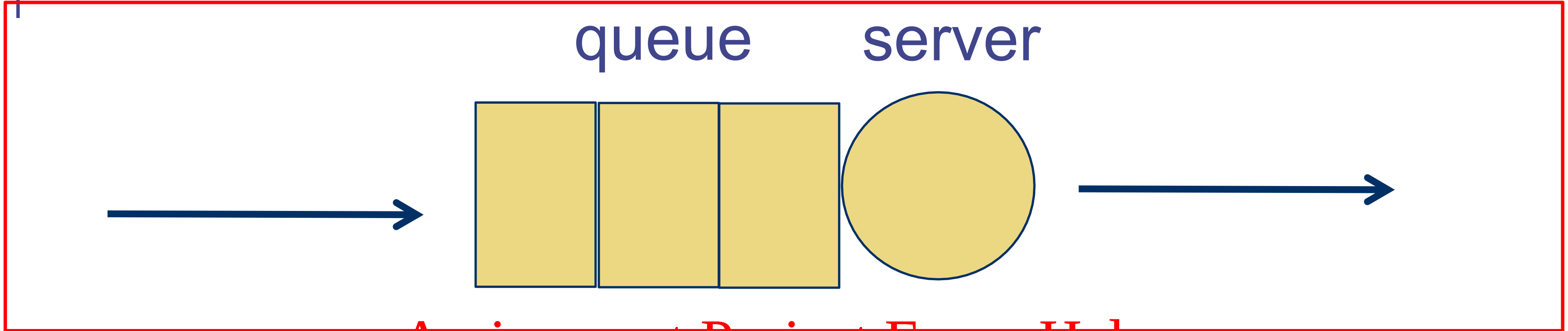
- Little's Law
  - Mean #jobs = Mean response time * Mean throughput
- If # jobs in the device ↑ , then response time ↑
  - And vice versa

Applicability of Little' s Law

- 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) = #jobs in device j
  - Average number of jobs in the system Navg = Navg(1) + …. + Navg(K)
  - Average response time of device j = Ravg(j)
  - Average response time of the system = Ravg
- We can also apply it to an entire system
  - Navg = Ravg * X(0)

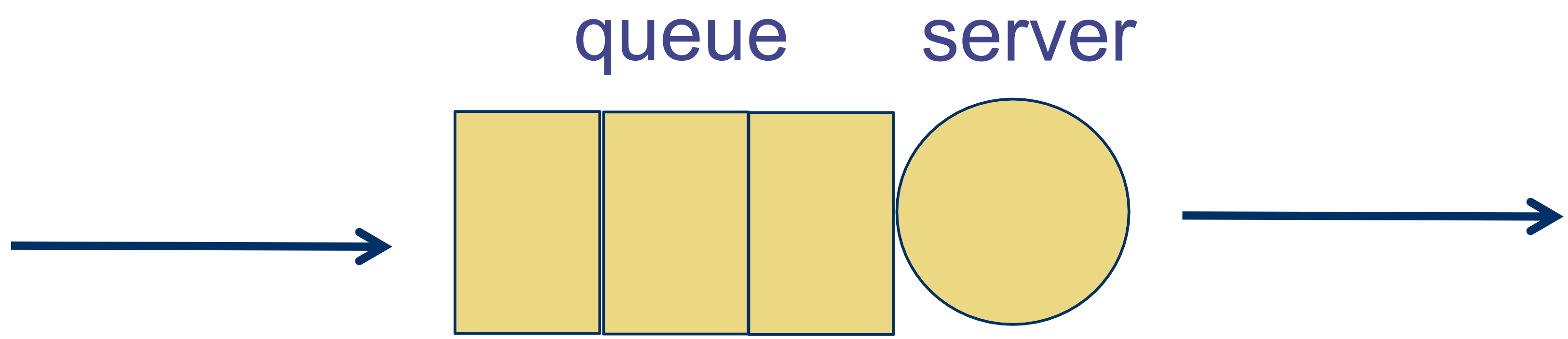

Terminals
Disks
CPU
1
2
3
4

Using Little's Law (2)



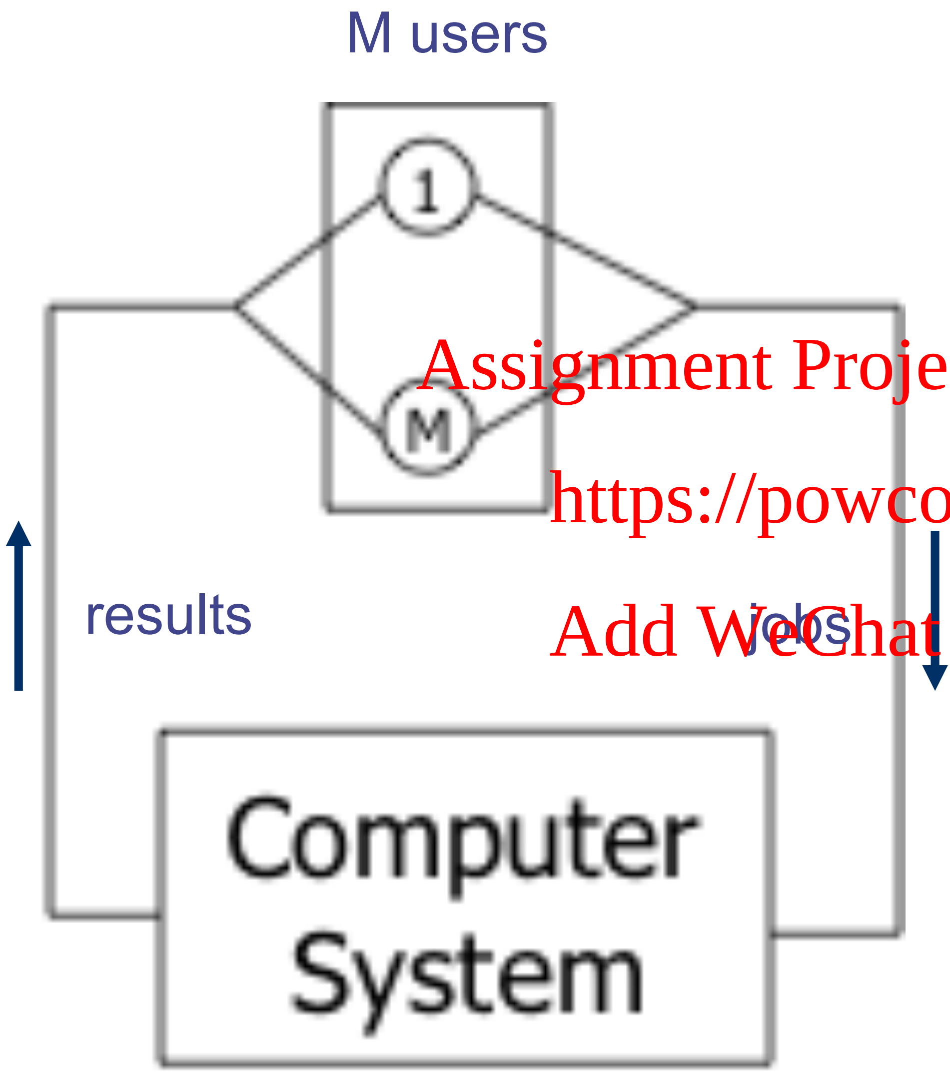
- The device completes on average 8 requests per second
- On average, there are
  - 3.2 requests in the device
  - 2.4 requests in the queue
  - 0.8 requests in the server
- What is the mean waiting time and mean service time?
- Hint: You need to draw "boxes" around certain parts of the device and interpret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

Using Little's Law (2)



- 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?

Interactive systems



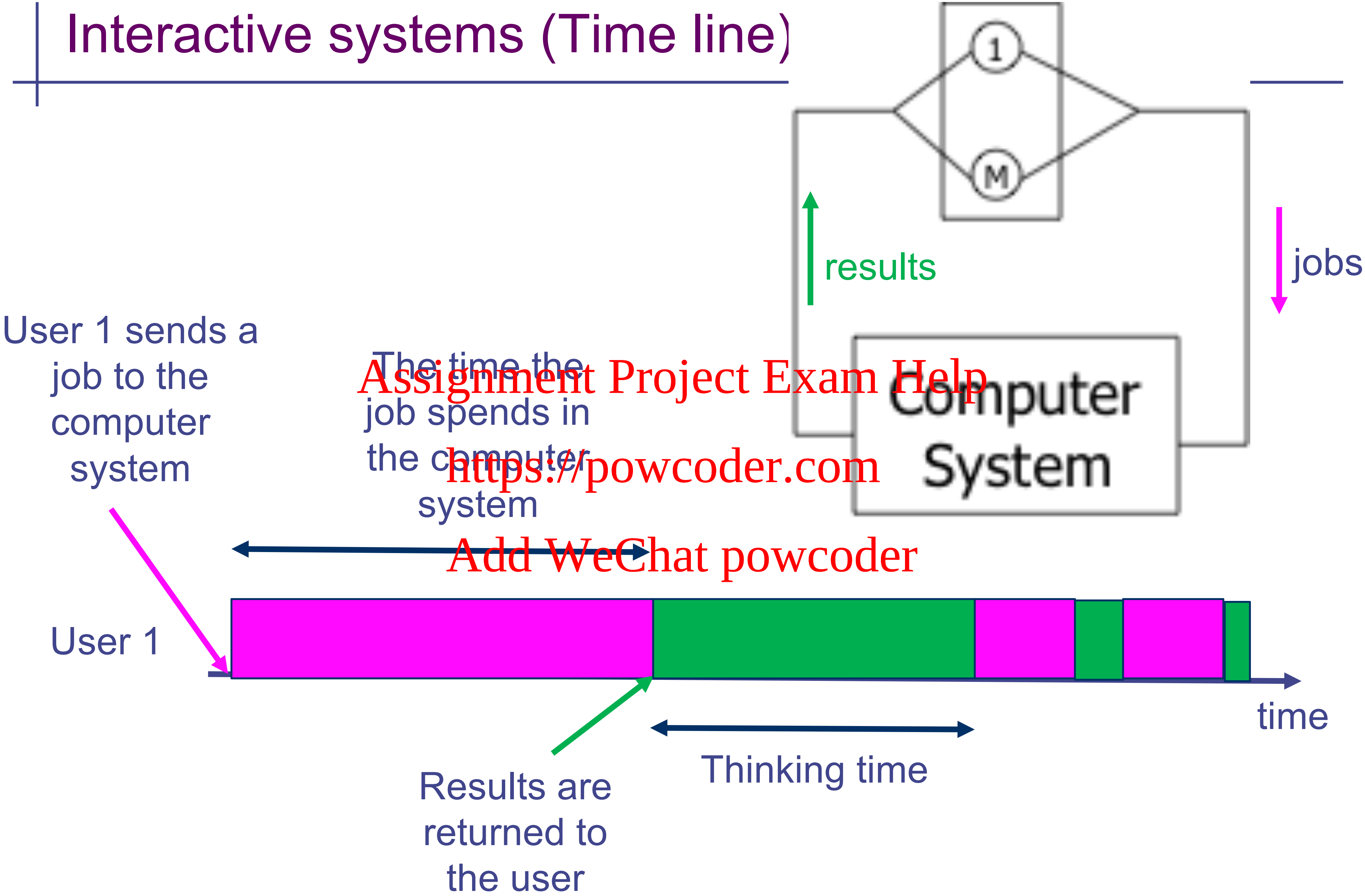
Each user sends a job to the system

The system sends the results to the user.

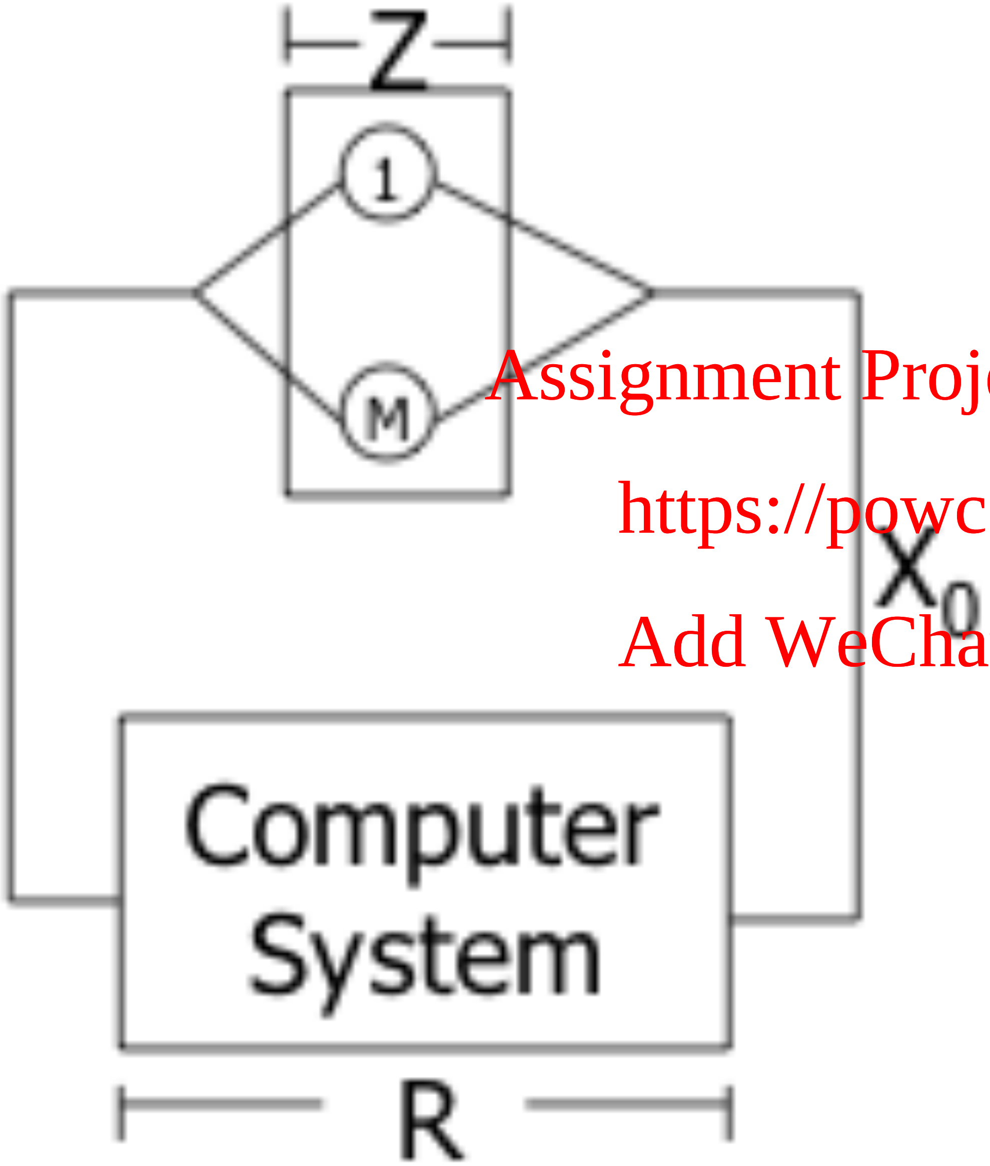
The user after a thinking time, sends another job to the system.
- Thinking time = time spent by the user

An interactive system is an example of closed system.

Interactive systems (Time line)

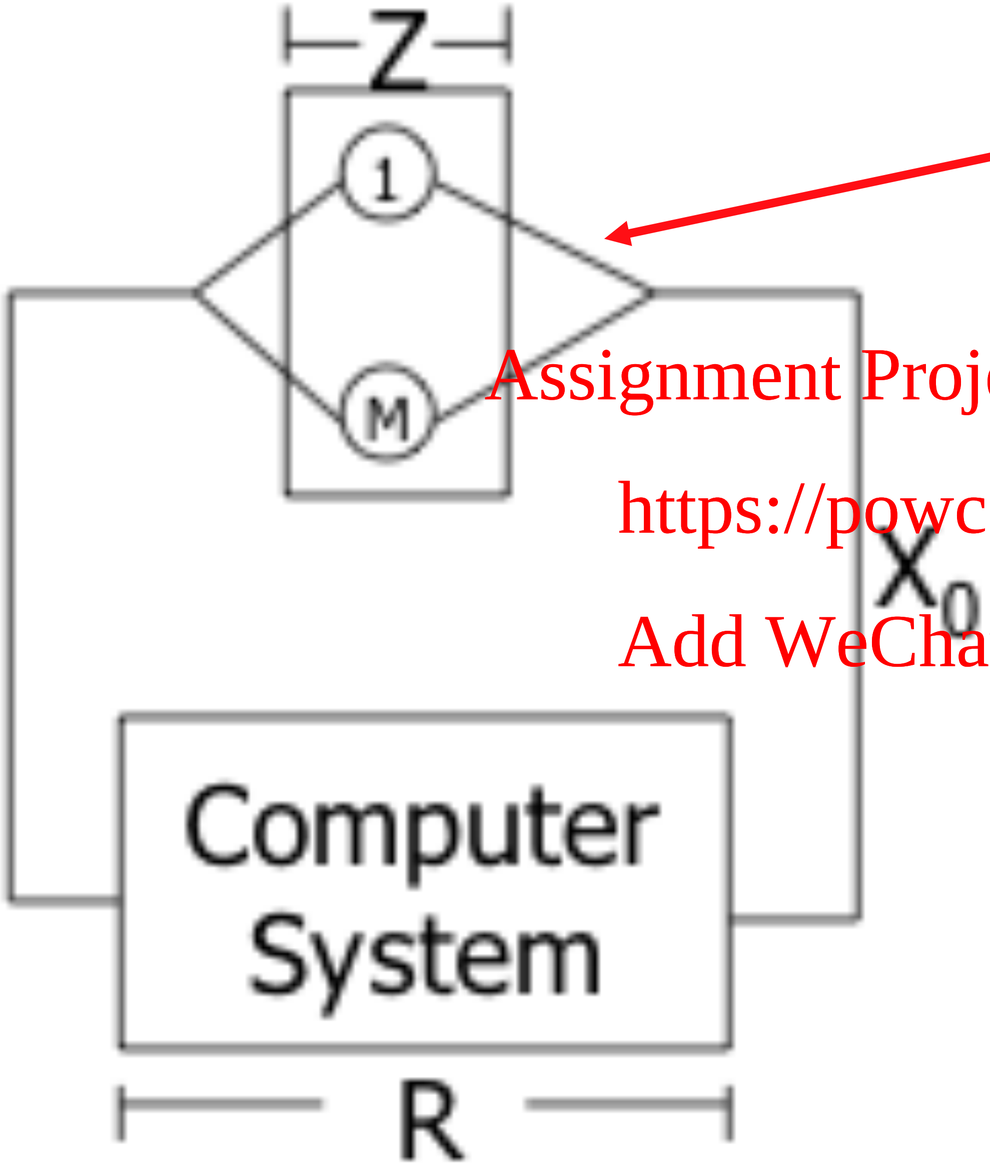


Interactive system (1)



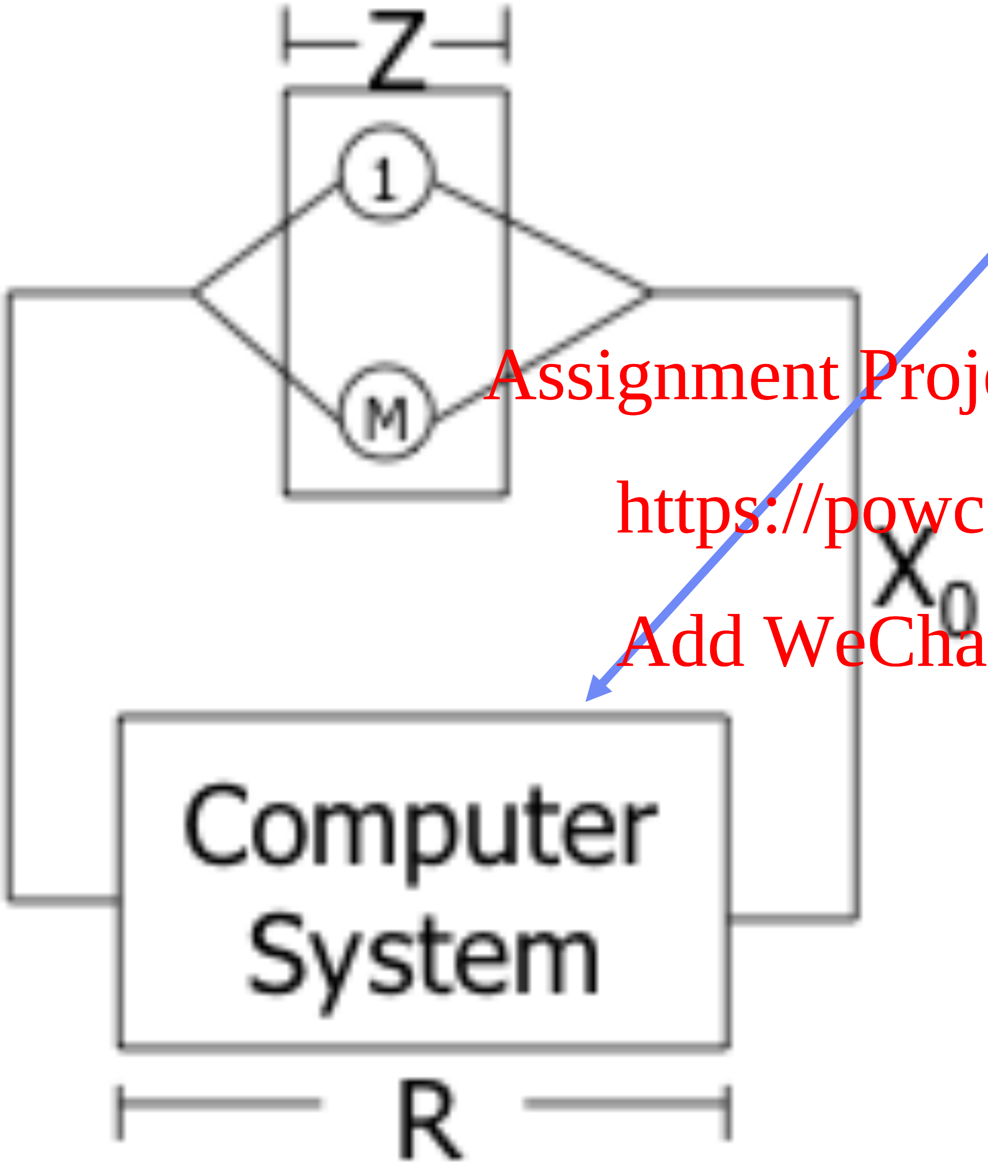
- M interactive clients
- Z = mean thinking time
- R = mean response time of the computer system
- X0 = throughput

Interactive system (2)



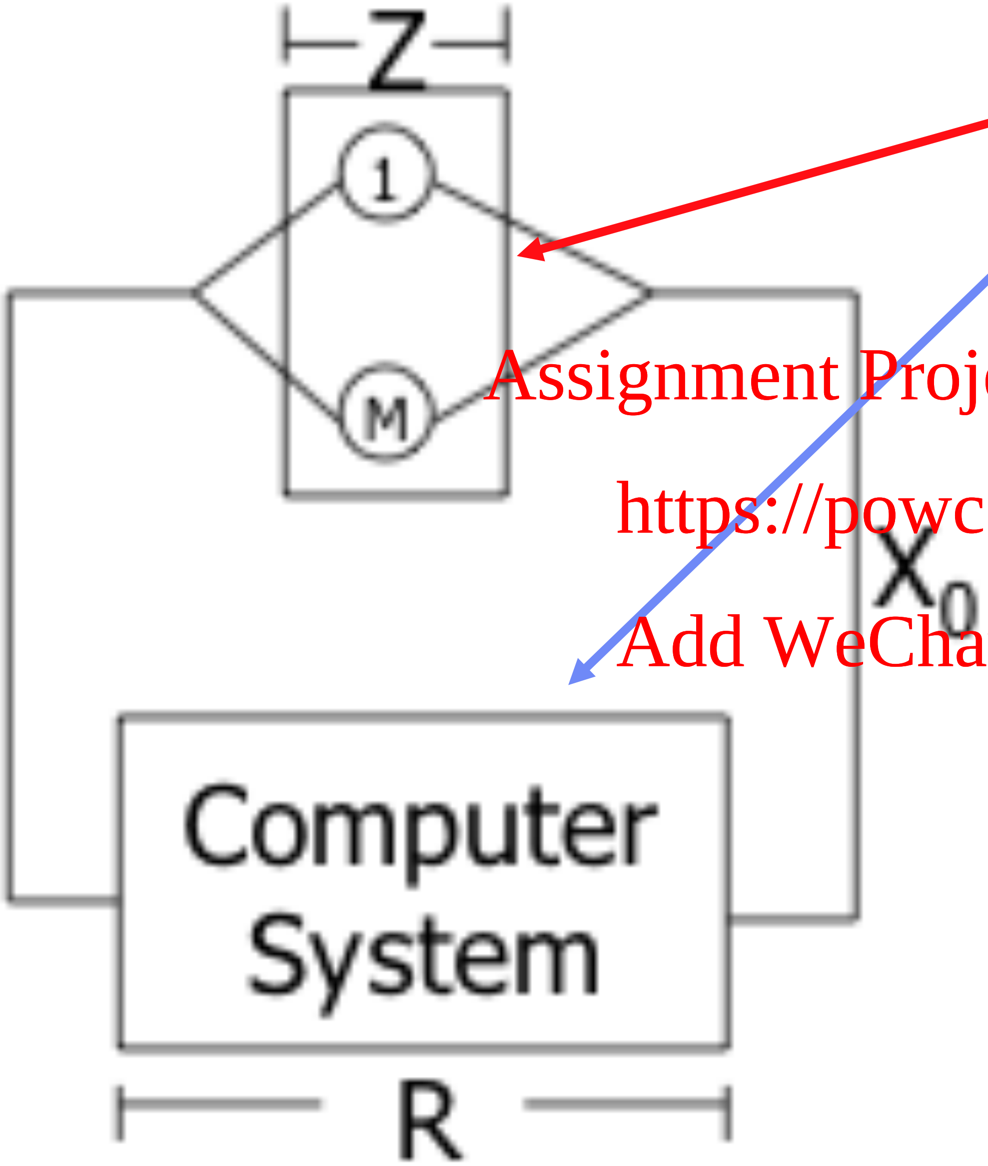
- Mavg = mean # interactive clients
- Z = mean thinking time
- X0 = throughput
- Apply Little's Law to the interactive part, we have Mavg = Z * X0

Interactive system (3)



- Navg = average # clients in the computer system
- R = mean response time at the computer system
- X0 = throughput
- Apply Little's Law to the computer system, we have Navg = R * X0

Interactive system (4)

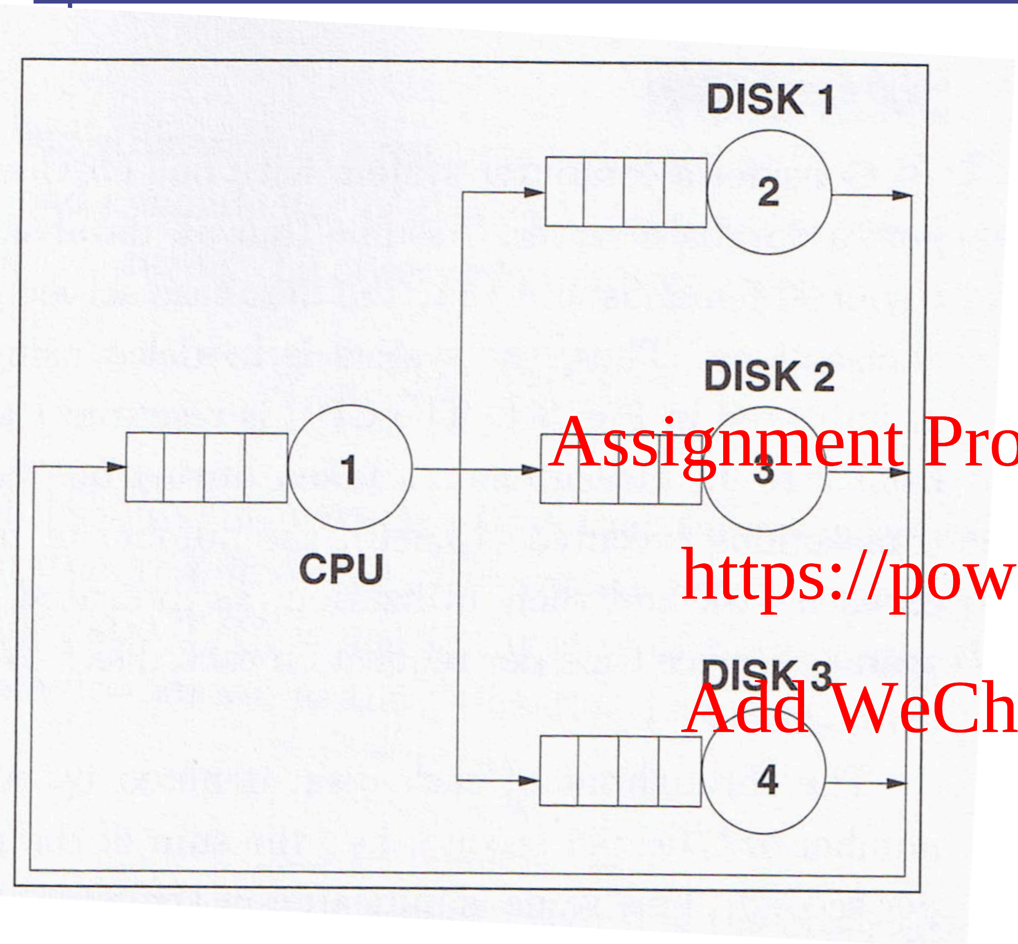


- Mavg = X0 * Z
- Navg = X0 * R
- The system is closed, the total number of users M is a constant, we have
- M = Mavg + Navg
- Therefore,
- M = X0 * (Z+R)

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

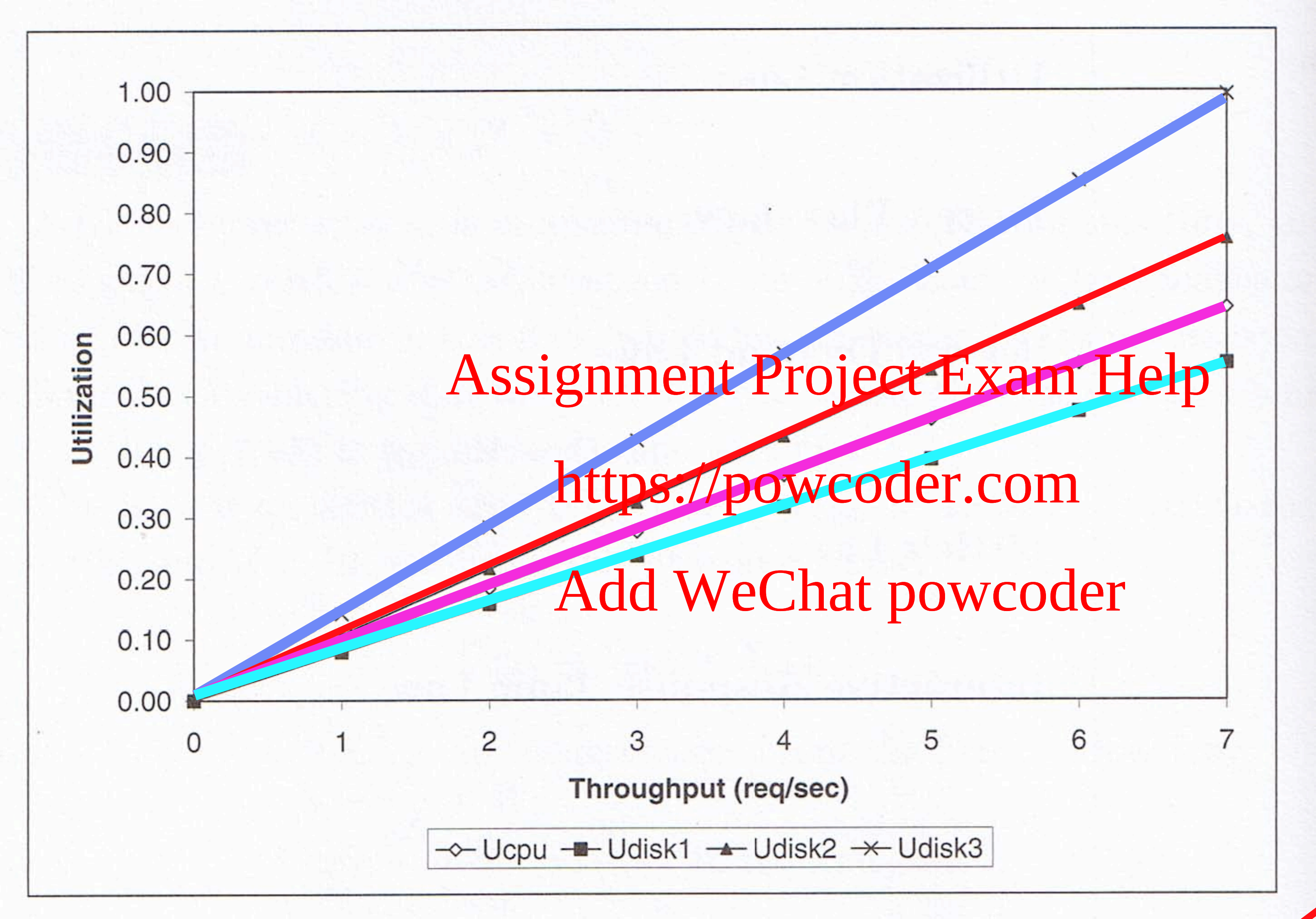
Bottleneck analysis - motivation



	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

Utilisation vs. throughput plot U(j) = D(j) X(0)



Disk 3

Disk 2

CPU

Disk 1

What determines this order?

Observation: For all system throughput:
Utilisation of Disk 3 > Utilisation of Disk 2 > Utilisation of CPU > Utilisation of Disk 1

Bottleneck analysis

- Recall that utilisation is the busy time of a device divided by measurement time
  - What is the maximum value of utilisation?
- Based on the example on the previous slide, which device will reach the maximum utilisation first?

Bottleneck (1)

- 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: $$U(j) \leq 1$$

$$\left.\begin{array}{l} X(0) = \frac{U(j)}{D(j)} \\ U(j) \leq 1 \end{array}\right\} \quad X(0) \leq \frac{1}{D(j)}$$

Bottleneck (2)

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

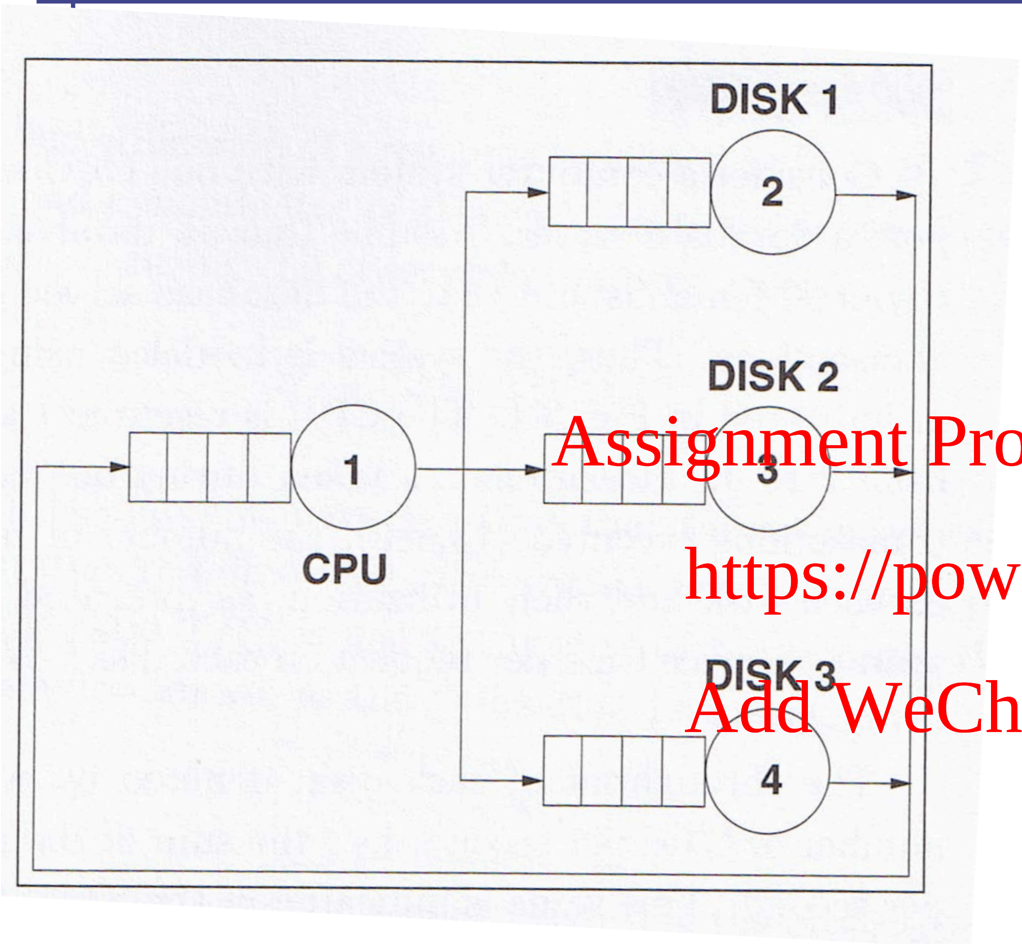
$$i.e. X(0) \leq \frac{1}{D(1)}, \ldots, X(0) \leq \frac{1}{D(K)}$$

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

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

Bottleneck throughput is limited by the maximum service demand

Bottleneck exercise



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

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

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

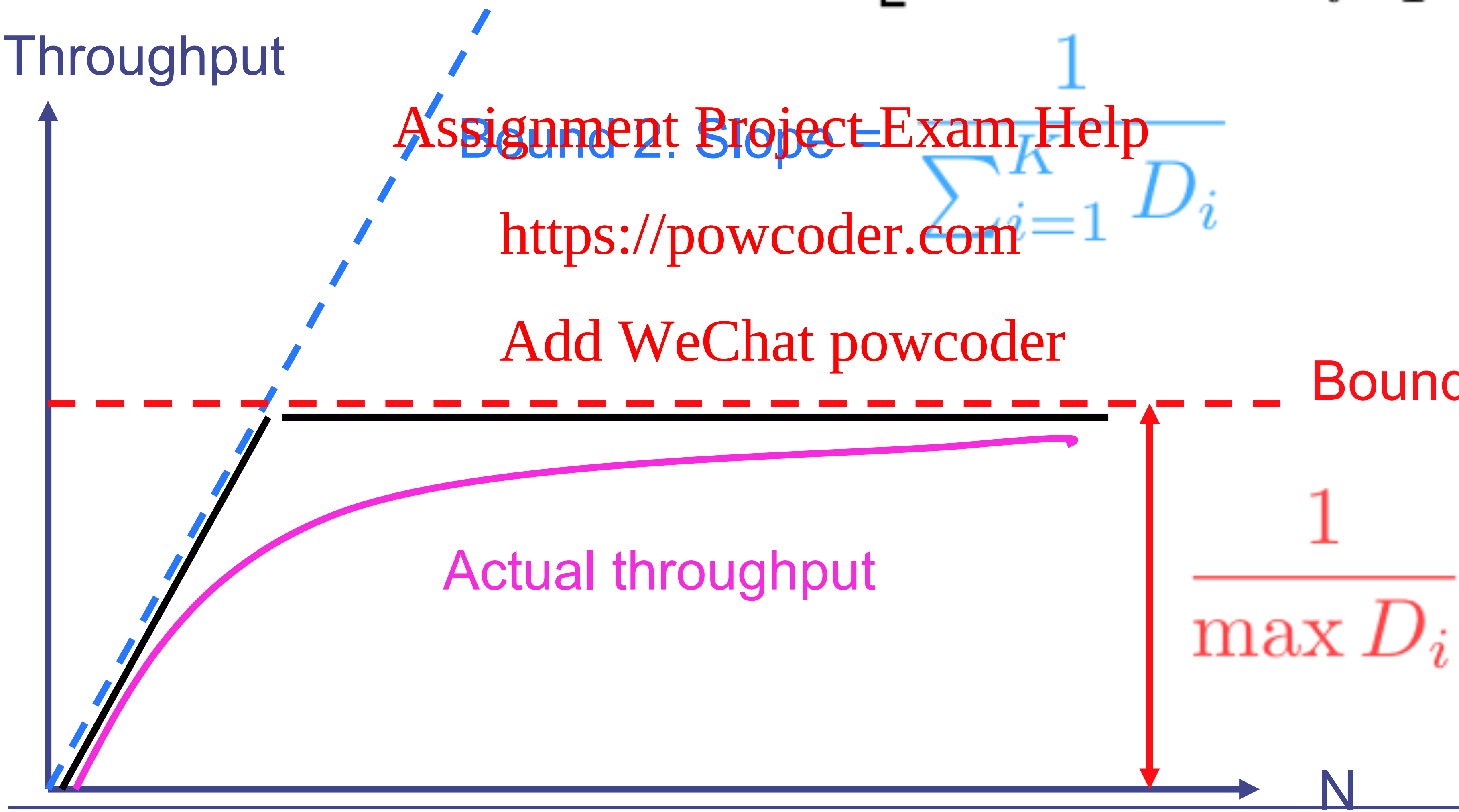
$$\Rightarrow X(0) \leq \frac{N}{\sum_{i=1}^{K} D_i}$$

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

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

Throughput bounds

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



Bottleneck analysis

- Simple to use
  - Needs only utilisation of various components
- Assumes service demand is load independent

Modification analysis (1)

- (Reference: Lazowska Section 5.3.1)
- 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:

System	Service demand (seconds)	
	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.

Modification analysis (2)

Operational analysis

- These are the operational laws
  - Utilisation law U(j) = X(j) S
  - 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)

- 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 performance bound earlier, the bounds depend on number of users and service demand
  - Queue response time depends on the job arrival rate and job service time

- One way of specifying workload is to use probability distribution.

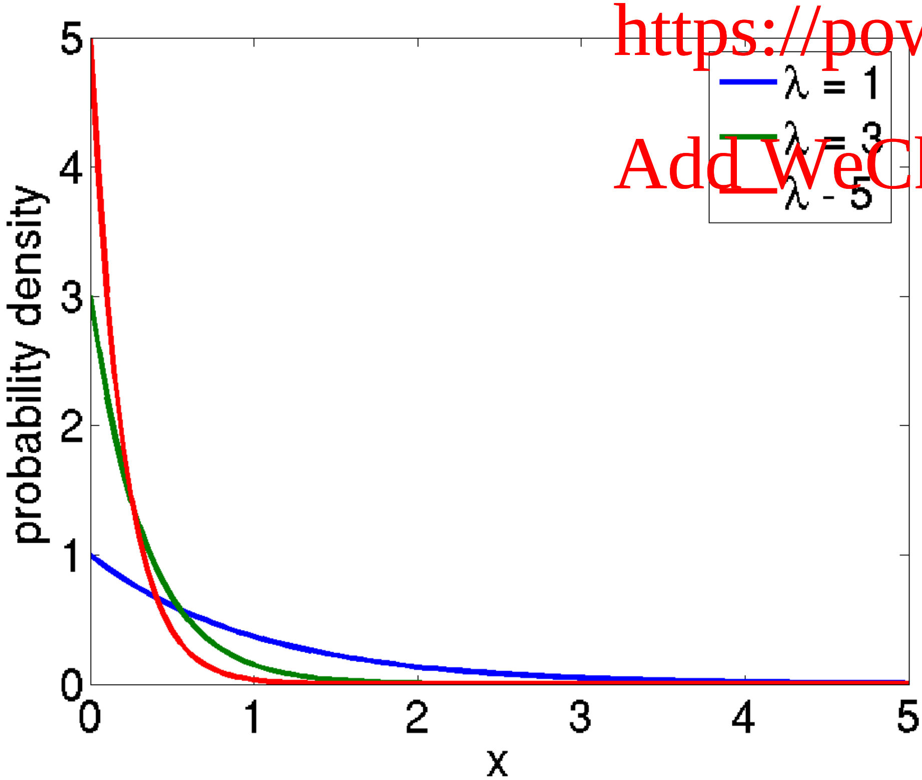
- We will look at a well-known arrival process called Poisson process today.

- We will first begin by looking at exponential distribution.

Exponential distribution (1)

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

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



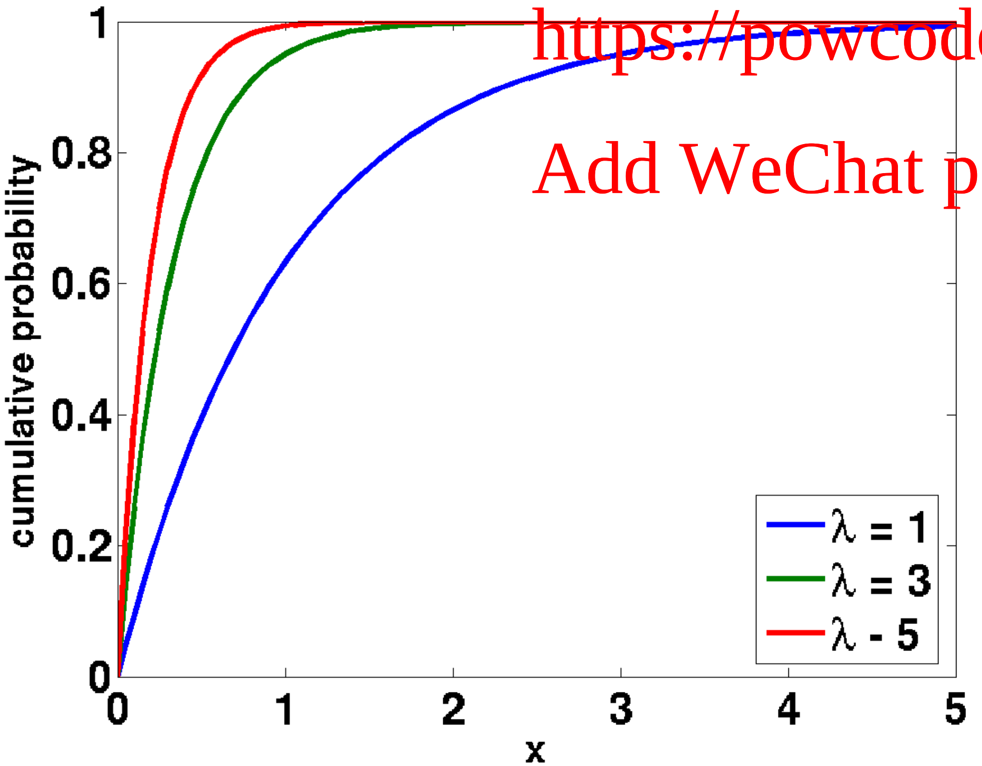
Probability that $x \leq X \leq x + \delta x$ is

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

Exponential distribution - cumulative distribution

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

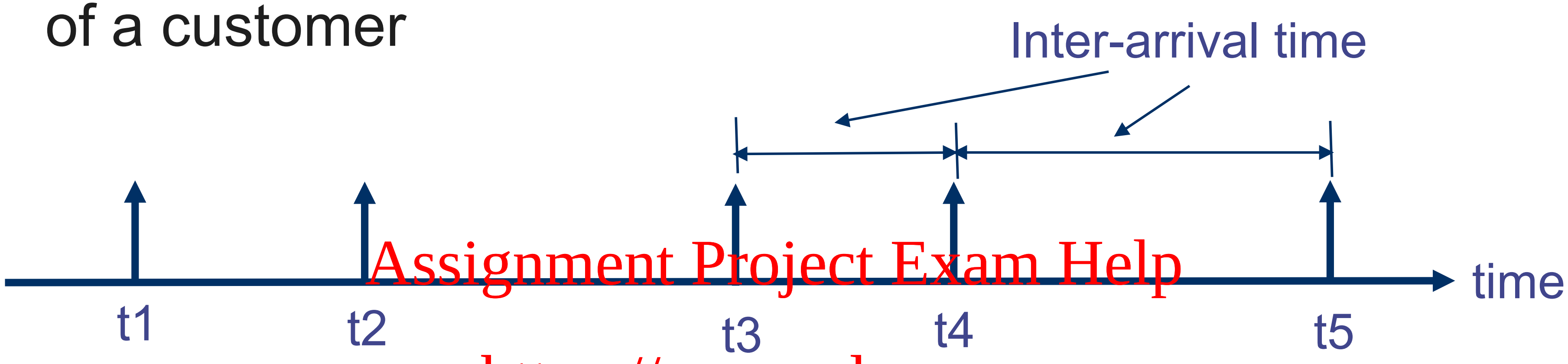
$$F(x) = \int_0^x \lambda e^{-\lambda z} dz = 1 - e^{-\lambda x} \text{ for } x \geq 0$$



What is Prob(X ≥ x)?

Arrival process

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



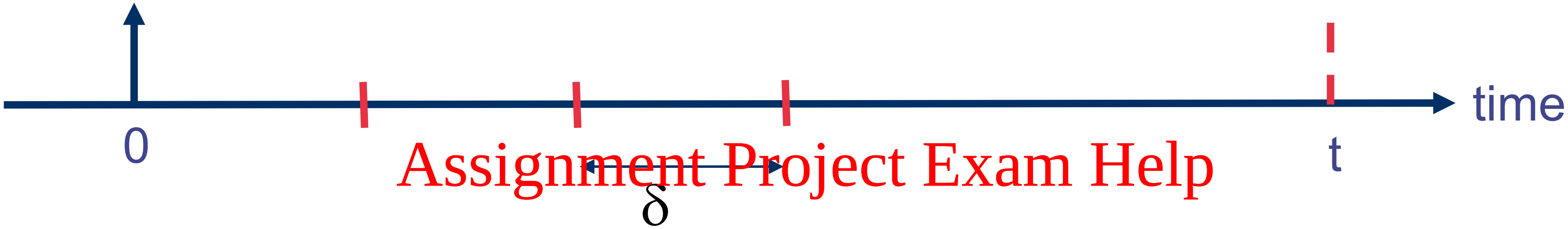
- An arrival can mean
  - A telephone call arriving at a call centre
  - A transaction arriving at a computer system
  - A customer arriving at a checkout counter
  - An HTTP request arriving at a web server
- The inter-arrival time distribution will impact on the response time.
- We will study an inter-arrival distribution that results from a large number of **independent** customers.

Many independent arrivals (1)

- Assume there is a large pool of N customers
- Within a time period of δ (δ is a small time period), there is a probability of $p\delta$ that a customer will make a request (which gives rise to an arrival)
- Assuming the probability that each customer makes a request is independent, the probability that a customer arrives in time period δ is $Np\delta$
- If a customer arrives at time 0, what is the probability that the next customer does not arrive before time t

Many independent arrivals (2)

- Divide the time t into intervals of width δ



- No arrival in [0,t] means no arrival in each interval δ
- Probability of no arrival in δ = 1 - $Np\delta$
- There are t / δ intervals
- Probability of no arrival in [0,t] is

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

Exponential inter-arrival time

- We have showed that the probability that there is no arrival in [0,t] is 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

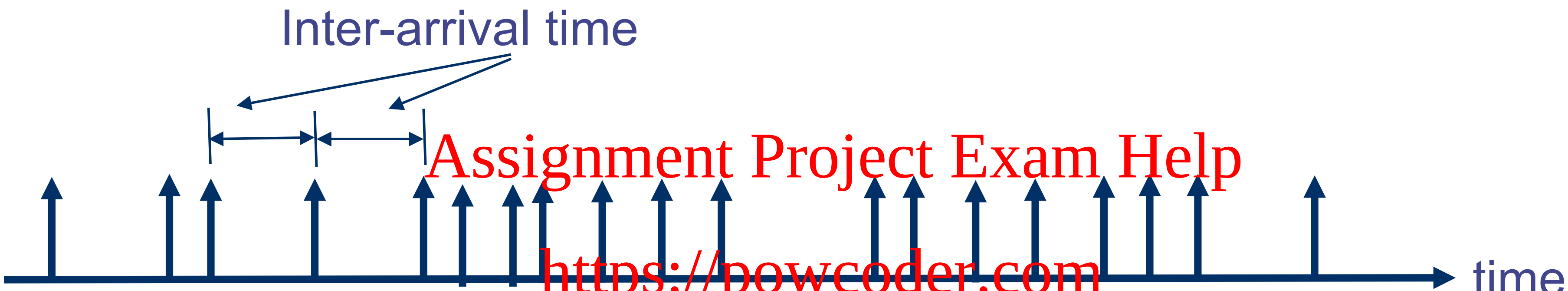
  Probability(inter-arrival time $\leq$ t) = 1 - exp(- N p t)

- 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

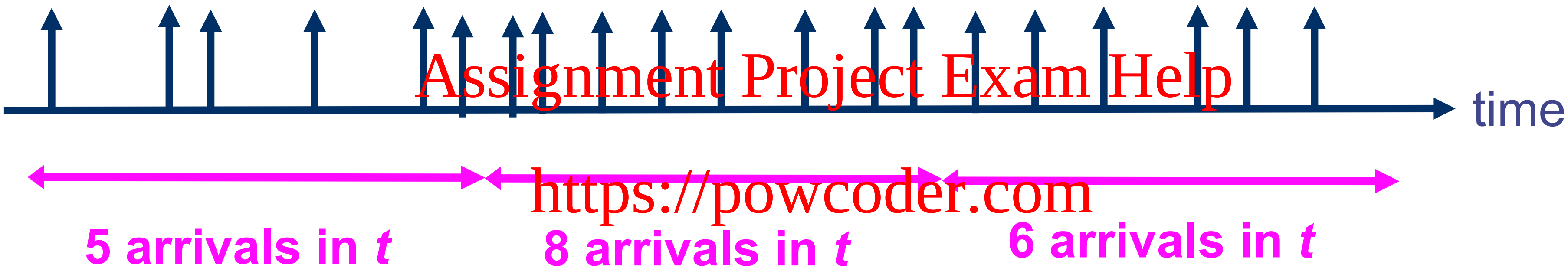
Two different methods to describe arrivals

Method 1: Continuous probability distribution of inter-arrival time



Two different methods to describe arrivals

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

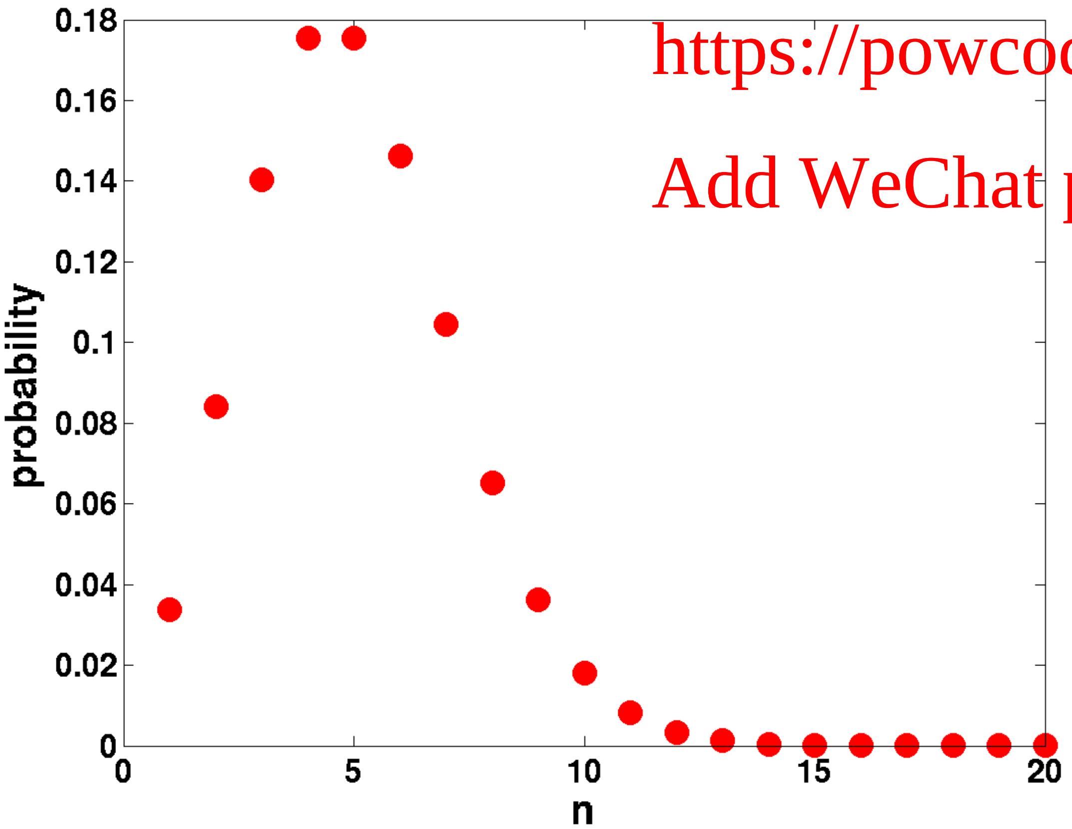


- The number of arrivals in *t* is random
- The number of arrivals must be an non-negative integer
- We need a discrete probability distribution:
  - Prob[#arrivals in t = 0]
  - Prob[#arrivals in t = 1]
  - etc.

Poisson process (1)

- 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:

λ= 5 and t = 1

Note: Poisson is a discrete probability distribution.

Poisson process (2)

- 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.

References

- Operational analysis
  - Lazowska et al, Quantitative System Performance, Prentice Hall, 1984. (Classic text on performance analysis. Now out of print but can be download from http://www.cs.washington.edu/homes/lazowska/qsp/
    - Chapters 3 and 5 (For Chapter 5, up to Section 5.3 only)
  - Alternative 1: You can read Menasce et al, "Performance by design", Chapter 3. Note that Menasce doesn't cover certain aspects of performance bounds. So, you will also need to read Sections 5.1-5.3 of Lazowska.
  - Alternative 2: You can read Harcol-Balter, Chapters 6 and 7. The treatment is more rigorous. You can gross over the discussion mentioning ergodicity.

- Little's Law (Optional)
  - I presented an intuitive "proof". A more formal proof of this well known Law is in Bertsekas and Gallager, "Data Networks", Section 3.2

- Tutorial exercises based on this week's lecture are available from course web site
  - We will discuss the questions in next week's tutorial time