CS 6323
Complex Networks and Systems:
Modeling and Inference

Lecture 5:
Performance Analysis I

Prof. Gregory Provan
Department of Computer Science
University College Cork

Slides: Based on M. Yin (Performability Analysis)

Overview

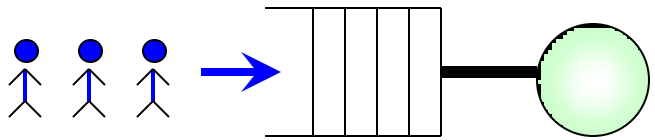
- Queuing Models
- Birth-Death Process
- Poisson Process
- Task Graphs

Network Modelling

- We model the transmission of packets through a network using *Queueing Theory*
- Enables us to determine several QoS metrics
  - Throughput times, delays, etc.
  - *Predictive* Model
- Properties of queueing network depends on
  - Network topology
  - Performance for each “path” in network
  - Network interactions

Queuing Networks

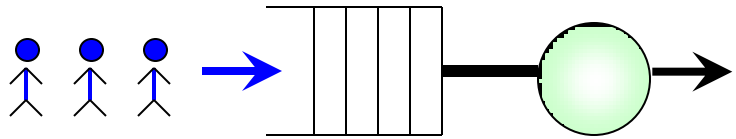
- A queuing network consists of service centers and customers (often called jobs)
- A service center consists of one or more servers and one or more queues to hold customers waiting for service.



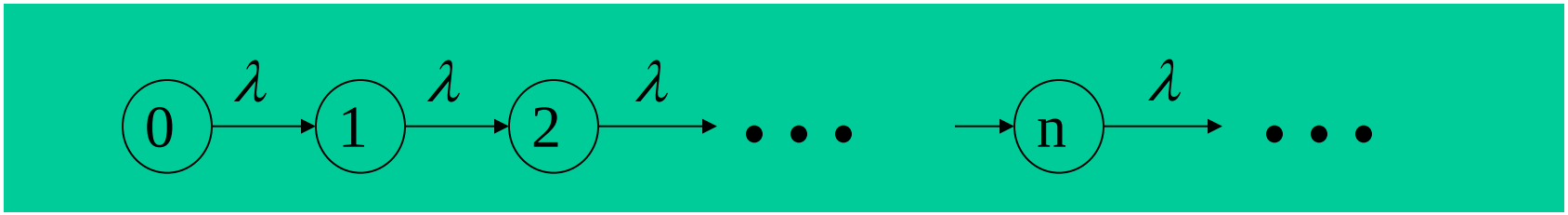
λ: customers arrival rate μ: service rate

Interarrival

- Interarrival time: the time between successive customer arrivals
- Arrival process: the process that determines the interarrival times
- It is common to assume that interarrival times are exponentially distributed random variables. In this case, the arrival process is a Poisson process



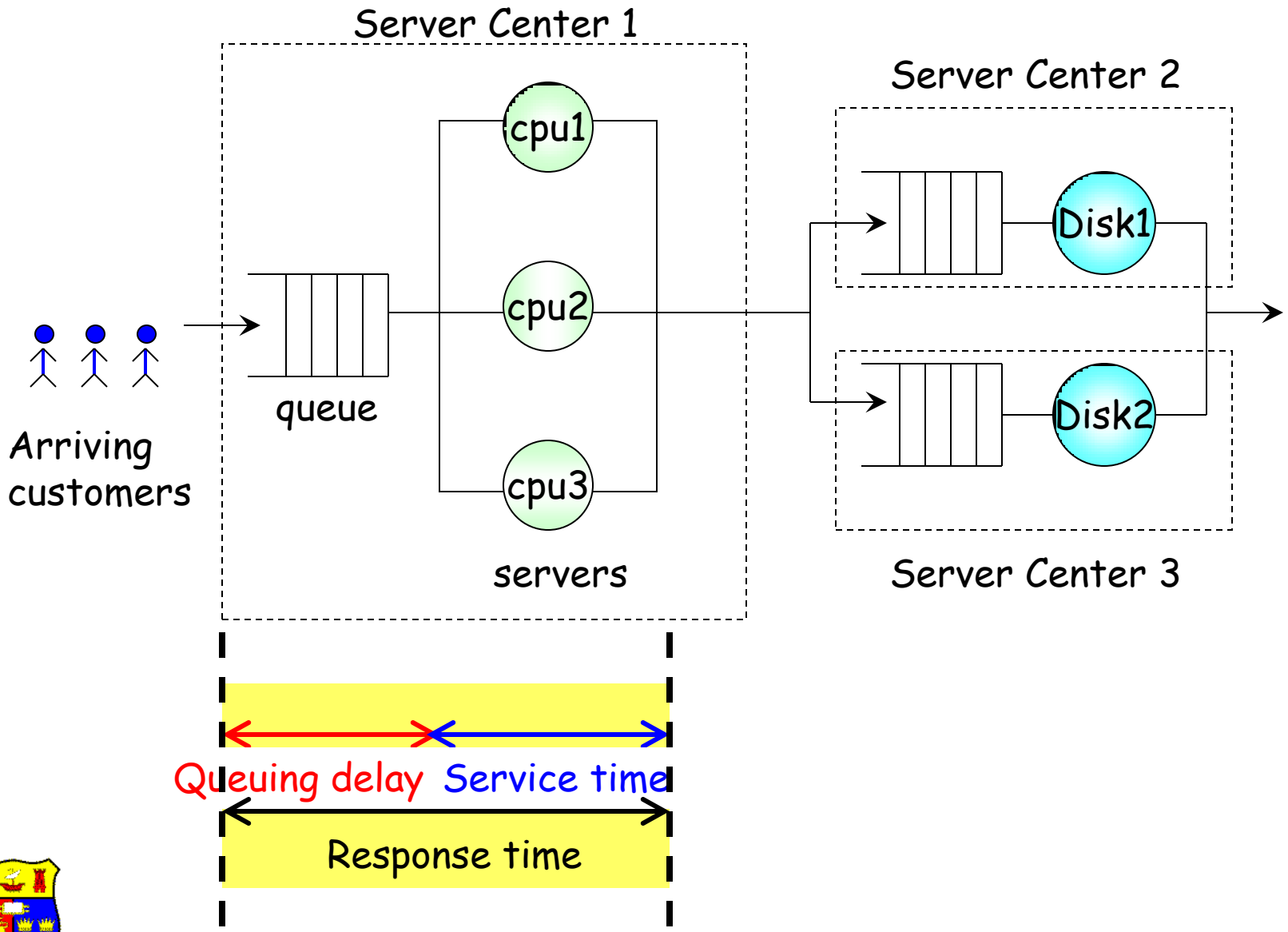
λ: customers arrival rate μ: service rate



Service Time

- The service time depends on how much work the customer needs and how fast the server is able to perform the work
- Service times are commonly assumed to be **independent, identically distributed** (***iid***) random variables.

A Queuing Model Example



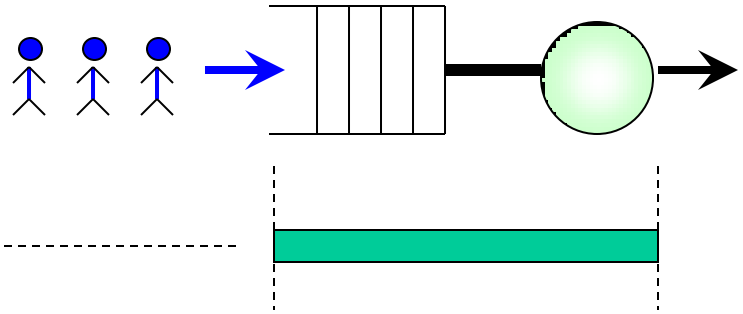
Terminology

Term	Explanations
Buffer size	The total number of customers allowed at a service center. The buffer size usually includes the ones waiting for service and receiving service
Population size	The total number of potential customers that may want to enter the service center. When it is large enough, it is usually assumed to be infinite.
Queuing discipline	The algorithm that determines the order in which customers are served. Some common recognized queuing disciplines: FCFS, priority, round robin (time sharing), etc.
Preemptive	A queuing discipline is called "preemptive" when the service being provided to a customer can be suspended if some higher priority customer arrives.

Open vs. Closed Queues

- Open:
  - the customers arrive from an external source
- Closed:
  - no external source of customers and no departures

Measures of Interest



- queue length
- response time
- throughput: the number of customers served per unit of time
- utilization: the fraction of time that a service center is busy

A relationship between **throughput** and **response time** – **Little's law.**

Little's Law

- The mean number of jobs in a queuing system in the steady state is equal to the product of the arrival rate and the mean response time

$$\overline{L} = \lambda \bullet \overline{W}$$

$\overline{L}$: The average number of customers in a queuing system

λ : the average arrival rate of customers admitted to the system

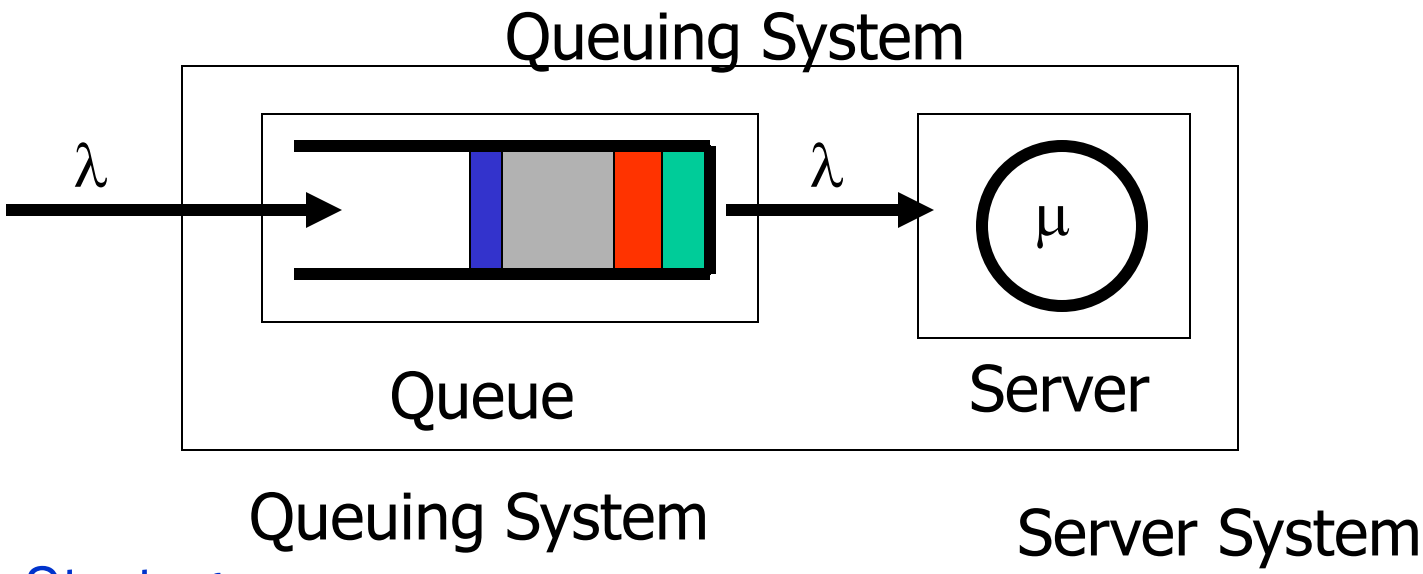
$\overline{W}$: The average time spent by a customer in the queuing system

Example using Little's law

- Cork Tunnel
  - Observe 120 cars queued in the Cork Tunnel
  - Observe 32 cars/minute arrive at the tunnel
- What is average waiting time before and in the tunnel?

$$W = \frac{L}{\lambda} = \left(\frac{120}{32}\right) = 3.75\text{min}$$

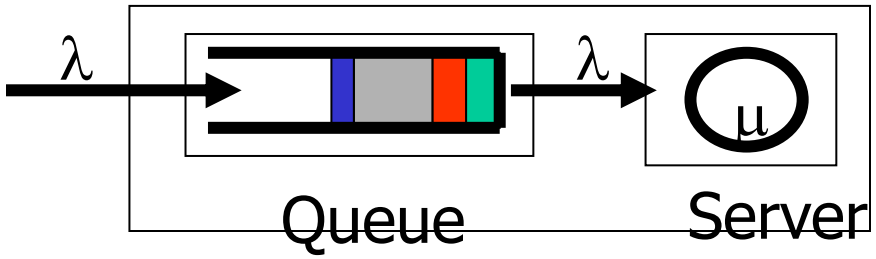
Model Queuing System



Strategy:

Use Little's law on both the complete system and its parts to reason about average time in the queue

Queuing Notation



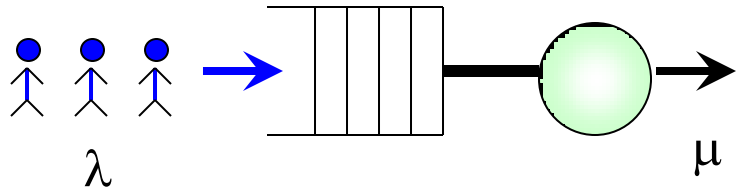
- Called an M/M/1 queue

Arrival Process

Service Process

Servers

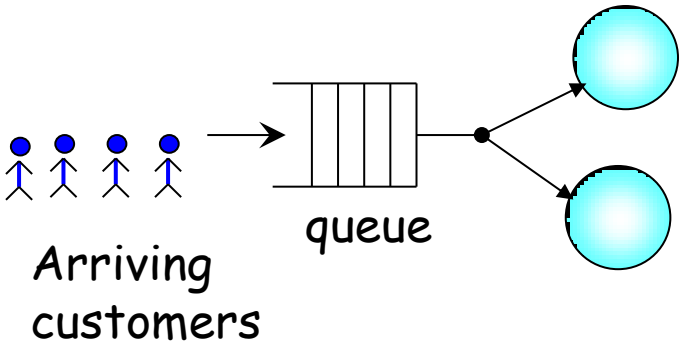
M/M/1 Queue



M/M/1 queue

- The first ‘M’ means the arrival process is exponential distributed
- The second ‘M’ means the service process is exponential distributed
- ‘1’ means the number of servers is 1
- Assuming buffer size and population size are infinity
- First-Come First-Served discipline is applied

M/M/2 Queue



M/M/2

Exponential Arrival Process

Exponential Service Process

2 Servers

General (Kendal) Notation

Queuing Model is usually described as

X/Y/Z/K/L/D

(M denotes the exponential distribution, E for Erlang, H for hyperexponential, D for deterministic and G for general)

X: Arrival process
Y: Service process
Z: Number of servers at the service center
K: Buffer size
L: Population size
D: The queuing discipline

K, L and D are often omitted, which means **K, L are ∞ and D is FCFS**

Distributions

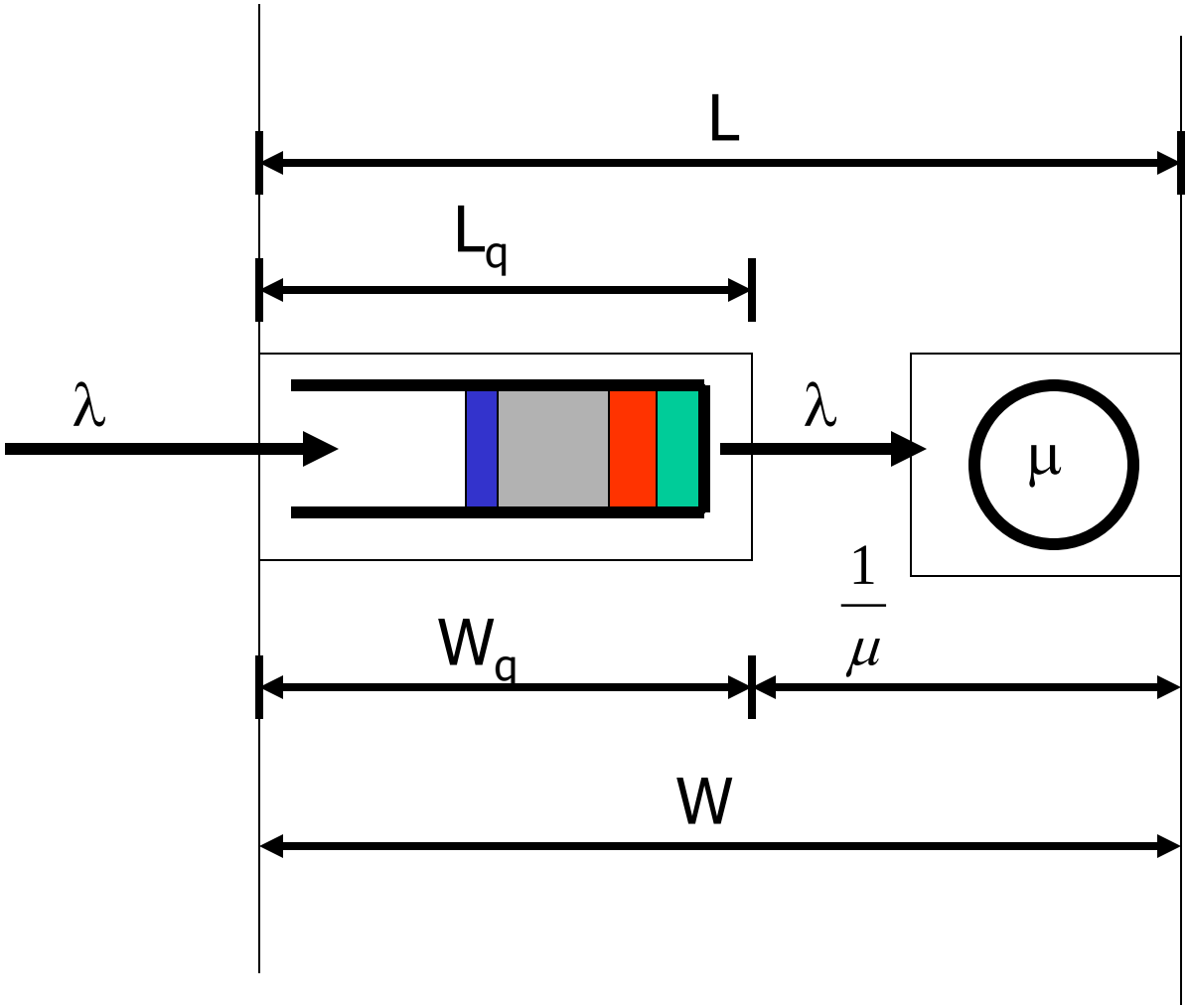
- M: Exponential
- D: Deterministic (e.g. fixed constant)
- E_k: Erlang with parameter k
- H_k: Hyperexponential with param. k
- G: General (anything)

- M/M/1 is the simplest ‘realistic’ queue

Kendal Notation Examples

- M/M/1:
  - Exponential arrivals and service, 1 server, infinite capacity and population, FCFS (FIFO)
- M/M/m
  - Same, but M servers
- G/G/3/20/1500/SPF
  - General arrival and service distributions, 3 servers, 17 queue slots (20-3), 1500 total jobs, Shortest Packet First

M/M/1 queue model



Analysis of M/M/1 queue

- Goal: A closed form expression of the probability of the number of jobs in the queue (P_i) given only λ and μ

Solving queuing systems

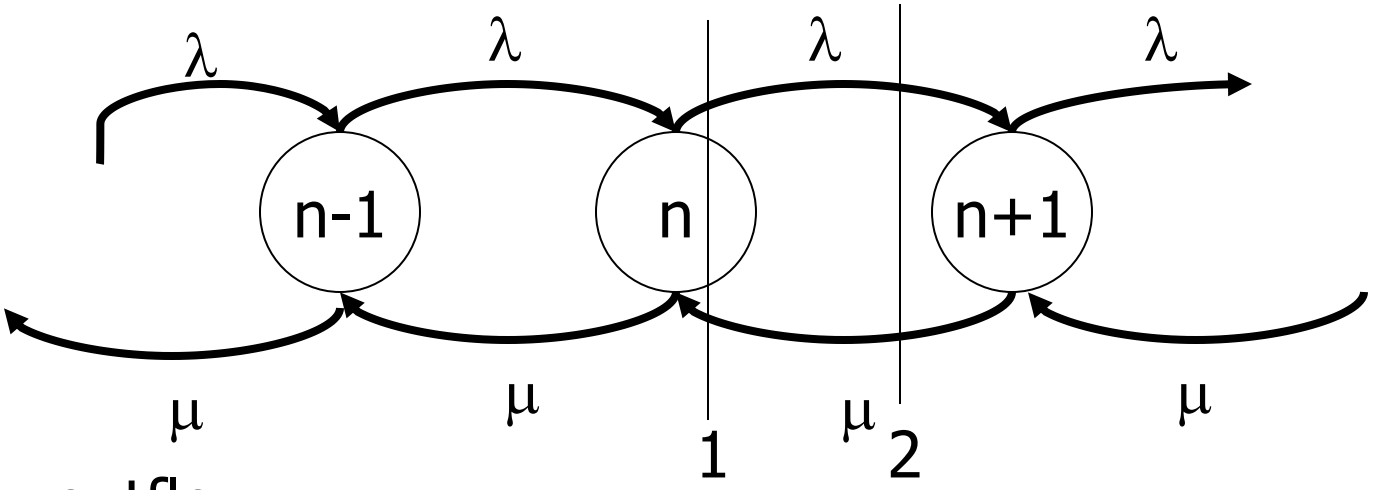
- Given:
  - λ: Arrival rate of jobs (packets on input link)
  - μ: Service rate of the server (output link)
- Solve:
  - L: average number in queuing system
  - L_q average number in the queue
  - W: average waiting time in whole system
  - W_q average waiting time in the queue
- 4 unknowns: need 4 equations

Solving queuing systems

- 4 unknowns: L, L_q W, W_q
- Relationships using Little's law:
  - $L=\lambda W$
  - $L_q=\lambda W_q$ (steady-state argument)
  - $W = W_q + (1/\mu)$
- If we know any 1, can find the others
- Finding L is hard or easy depending on the type of system. In general:

$$L = \sum_{n=0}^{\infty} nP_n$$

Equilibrium conditions



inflow = outflow

1: $(\lambda+\mu)P_n = \lambda P_{n-1} + \mu P_{n+1}$

2: $\lambda P_n = \mu P_{n+1}$

stability: 3: $\lambda \le \mu, \rho = \frac{\lambda}{\mu}, \rho \le 1$

Solving for P_0 and P_n

1: $P_1 = \rho P_0 \ , \ P_2 = (\rho)^2 P_0 \ , \ P_n = (\rho)^n P_0$

2: $\sum_{n=0}^{\infty} P_n = 1 \ , \ P_0 \sum_{n=0}^{\infty} \rho^n = 1 \ , \ P_0 = \frac{1}{\sum_{n=0}^{\infty} \rho^n}$,

3: $\sum_{n=0}^{\infty} \rho^n = \frac{1}{1-\rho}, \rho < 1$ (geometric series)

4: $P_0 = \frac{1}{\sum_{n=0}^{\infty} \rho^n} = \frac{1}{\frac{1}{(1-\rho)}} = 1-\rho$ 5: $P_n = (\rho)^n (1-\rho)$

Solving for L

$$L = \sum_{n=0}^{\infty} nP_n = \sum_{n=0}^{\infty} n\rho^n(1-\rho) = (1-\rho)\rho\sum_{n=1}^{\infty} n\rho^{n-1}$$

$$(1-\rho)\rho\frac{d}{d\rho}\left(\sum_{n=0}^{1}\rho^n\right) = (1-\rho)\rho\frac{d}{d\rho}\left(\frac{1}{1-\rho}\right)$$

$$(1-\rho)\rho\left(\frac{1}{(1-\rho)^2}\right) = \frac{\rho}{(1-\rho)} = \frac{\lambda}{\mu-\lambda}$$

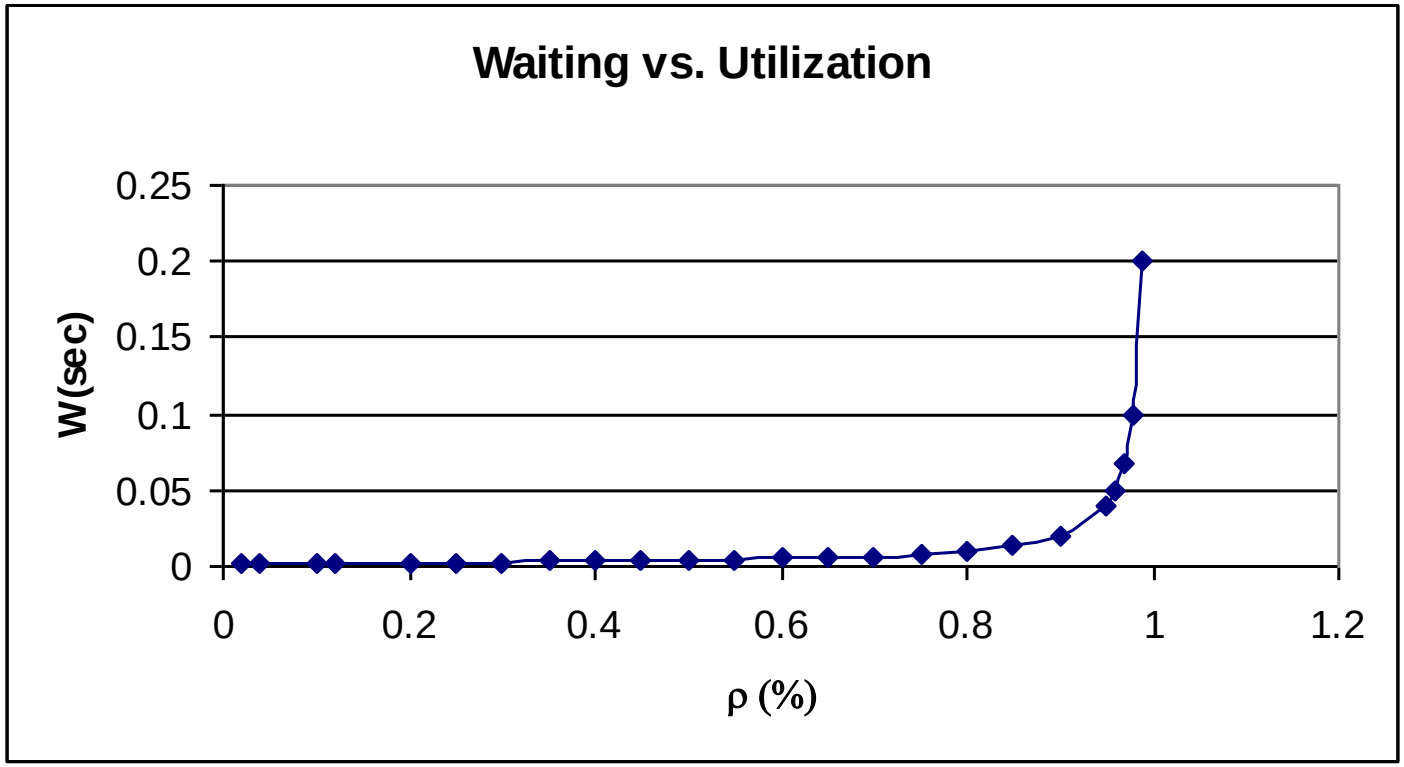
Solving W, W_q and L_q

$$W = \frac{L}{\lambda} = \left(\frac{\lambda}{\mu-\lambda}\right)\left(\frac{1}{\lambda}\right) = \frac{1}{\mu-\lambda}$$

$$W_q = W - \frac{1}{\mu} = \left(\frac{\lambda}{\mu-\lambda}\right) - \left(\frac{1}{\mu}\right) = \frac{\lambda}{\mu(\mu-\lambda)}$$

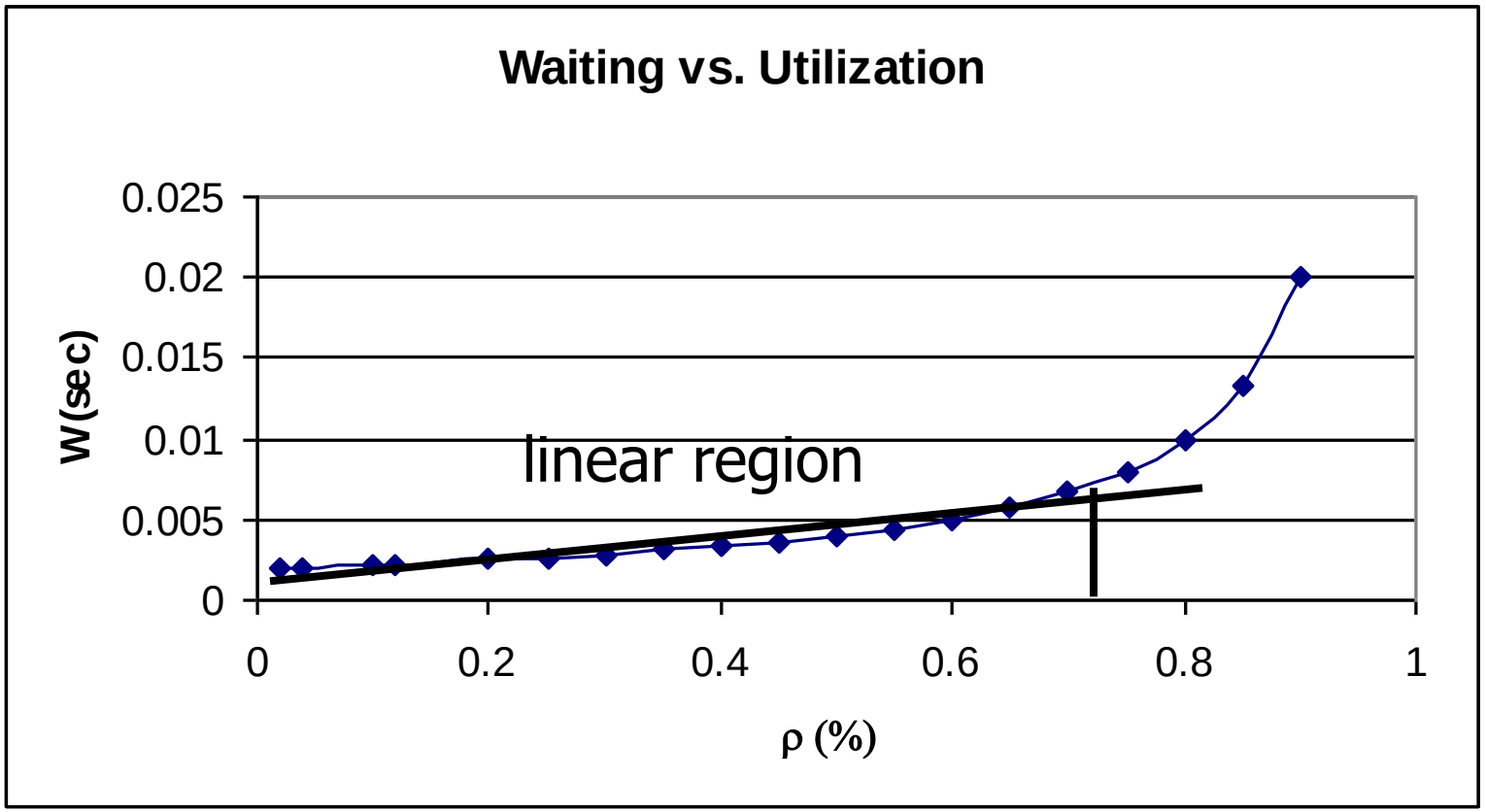
$$L_q = \lambda W_q = \lambda \frac{\lambda}{\mu(\mu-\lambda)} = \frac{\lambda^2}{\mu(\mu-\lambda)}$$

Response Time vs. Arrivals

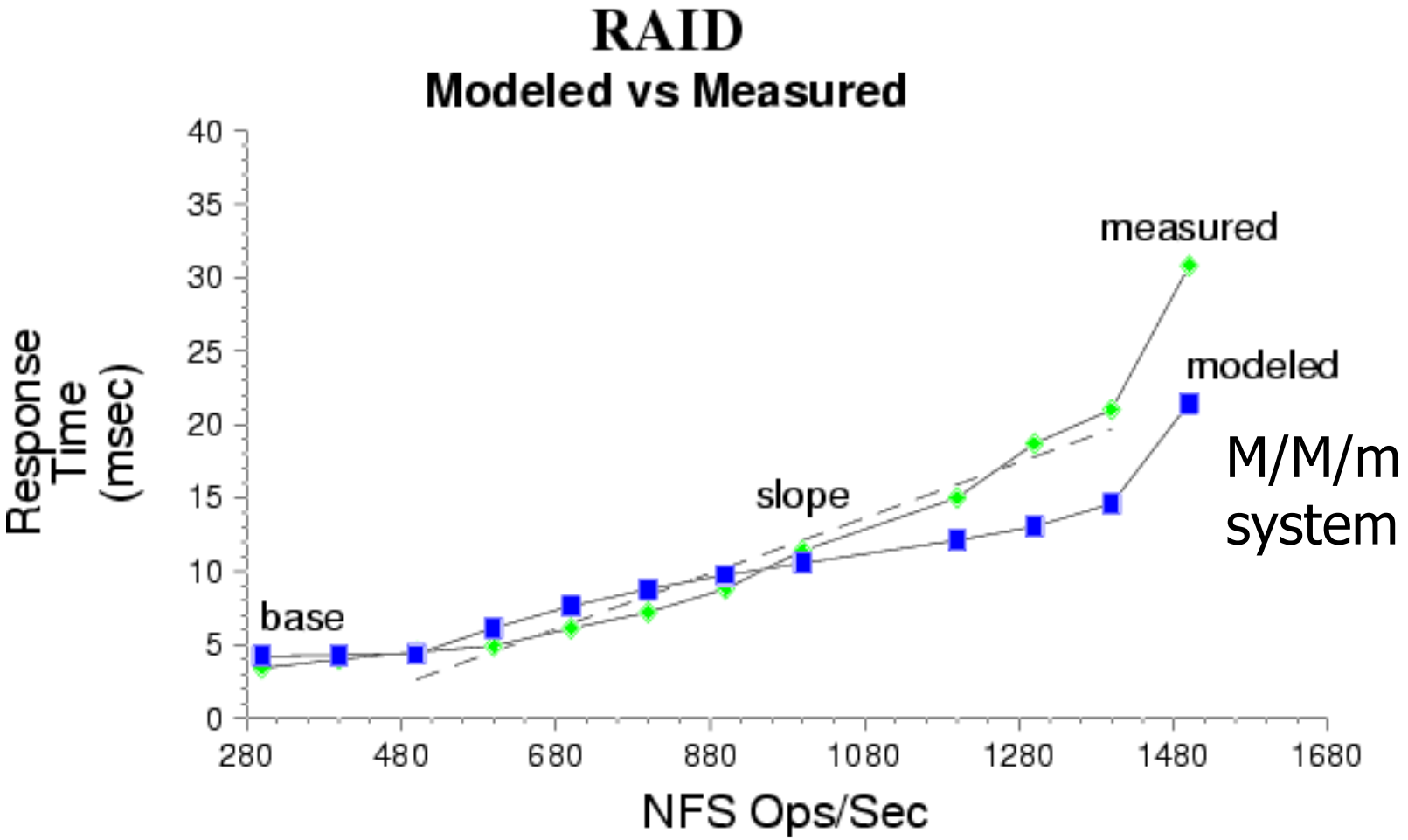


$$W = \frac{1}{\mu - \lambda}$$

Stable Region



Empirical Example



Key Parameters of Interest

The service rate = μ

utilization = $\rho = (\frac{\lambda}{\mu})$

P(n) packets in the gateway = $P_0 P_n = (1-\rho)(\rho^n)$

Mean # in gateway (L) = $\frac{\rho}{1-\rho}$

Example

- Measurement of a network gateway:
  - mean arrival rate (λ): 125 Packets/s
  - mean response time per packet: 2 ms
- Assuming exponential arrivals & departures:
  - What is the service rate, μ ?
  - What is the gateway's utilization?
  - What is the probability of n packets in the gateway?
  - mean number of packets in the gateway?
  - The number of buffers so P(overflow) is $<10^{-6}$?

Example (cont)

The service rate, $\mu = \frac{1}{0.002} = 500\,pps$

utilization = $\rho = (\frac{\lambda}{\mu}) = 0.25\%$

P(n) packets in the gateway =

$$P_0 P_n = (1-\rho)(\rho^n) = (0.75)(0.25^n)$$

Example (cont)

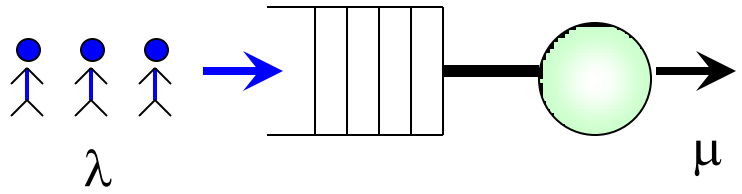
Mean # in gateway (L) =

$$\frac{\rho}{1-\rho} = \frac{0.25}{1-0.25} = 0.33$$

to limit loss probability to less than 1 in a million:

$$\rho^n \leq 10^{-6}$$

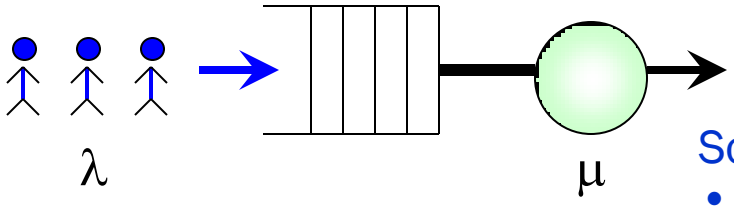
M/M/1 Queue



M/M/1 queue

- The first ‘M’ means the arrival process is exponential distributed
- The second ‘M’ means the service process is exponential distributed
- ‘1’ means the number of servers is 1
- Assuming buffer size and population size are infinity
- First-Come First-Served discipline is applied

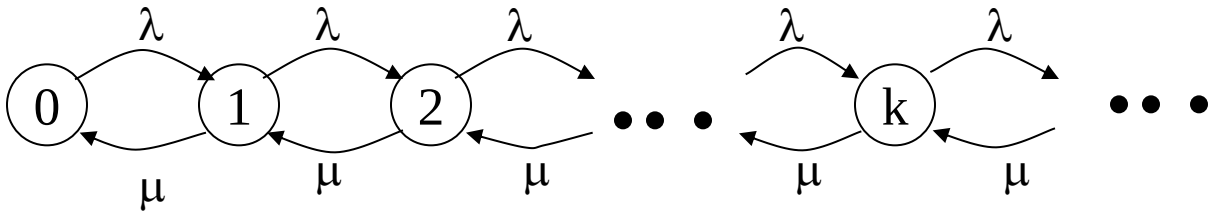
Solving M/M/1 queue



Solved for:

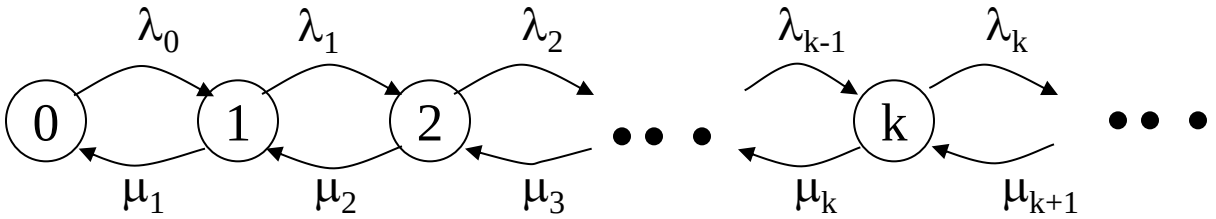
- The steady-state probability in each state
- Server Utilization
- The expected number of customers in the system
- The average response time

Construct the corresponding Markov Chain



A typical birth-death process

Birth-Death Process



A Birth-Death Process is a Markov chain where the transitions can occur only between adjacent states.

You can solve the Markov chain analytically by solving the set of 'balance equations'.

$$P_0 = \frac{1}{1 + \sum_{k \geq 1} \prod_{i=0}^{k-1} \left(\frac{\lambda_i}{\mu_{i+1}} \right)}$$

$$P_k = P_0 \prod_{i=0}^{k-1} \left(\frac{\lambda_i}{\mu_{i+1}} \right)$$

Solving M/M/1 Queue

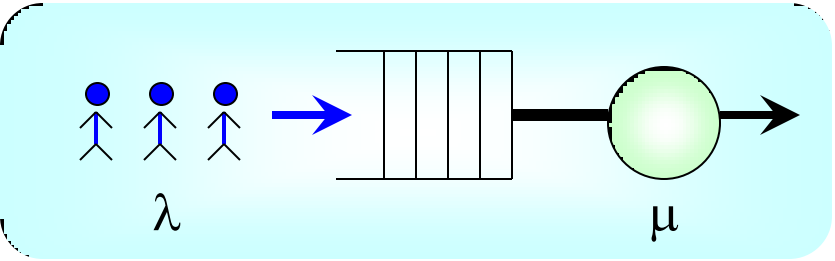
Define the ratio $\rho = \frac{\lambda}{\mu}$ called the utilisation rate, or **traffic intensity**

When $\rho < 1$ (meaning $\lambda<\mu$), the system is called "stable", and the steady state probabilities can be determined by

$$P_0 = \frac{1}{1+\sum_{k\geq 1}\rho^k} = \frac{1}{\sum_{k\geq 0}\rho^k} = 1-\rho$$

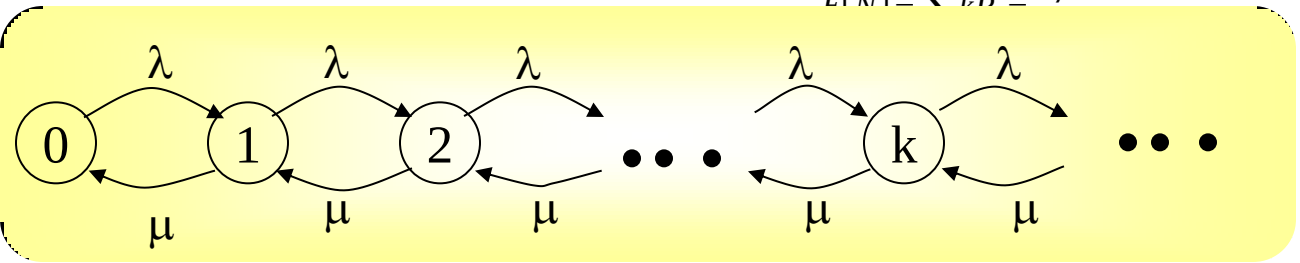
$$P_k = \left(\frac{\lambda}{\mu}\right)^k P_0 = \rho^k P_0 = \rho^k(1-\rho)$$

M/M/1 Queue Property



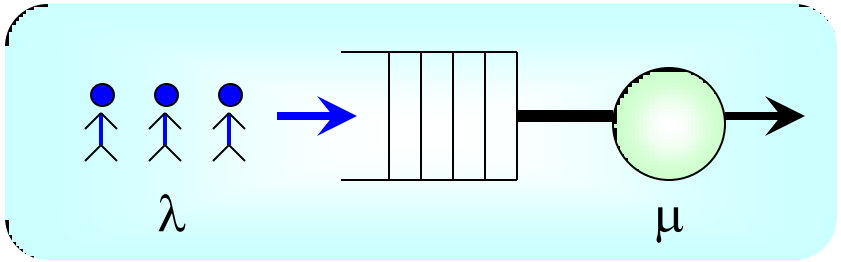
Steady State Probability for state k (k≥0): $(1-\rho)\rho^k$

Server Utilization: $1-P_0=1-(1-\rho)=\rho$



The mean of number of customers in the system E[N]

M/M/1 Queue Property: Average Response Time



Little's Formula

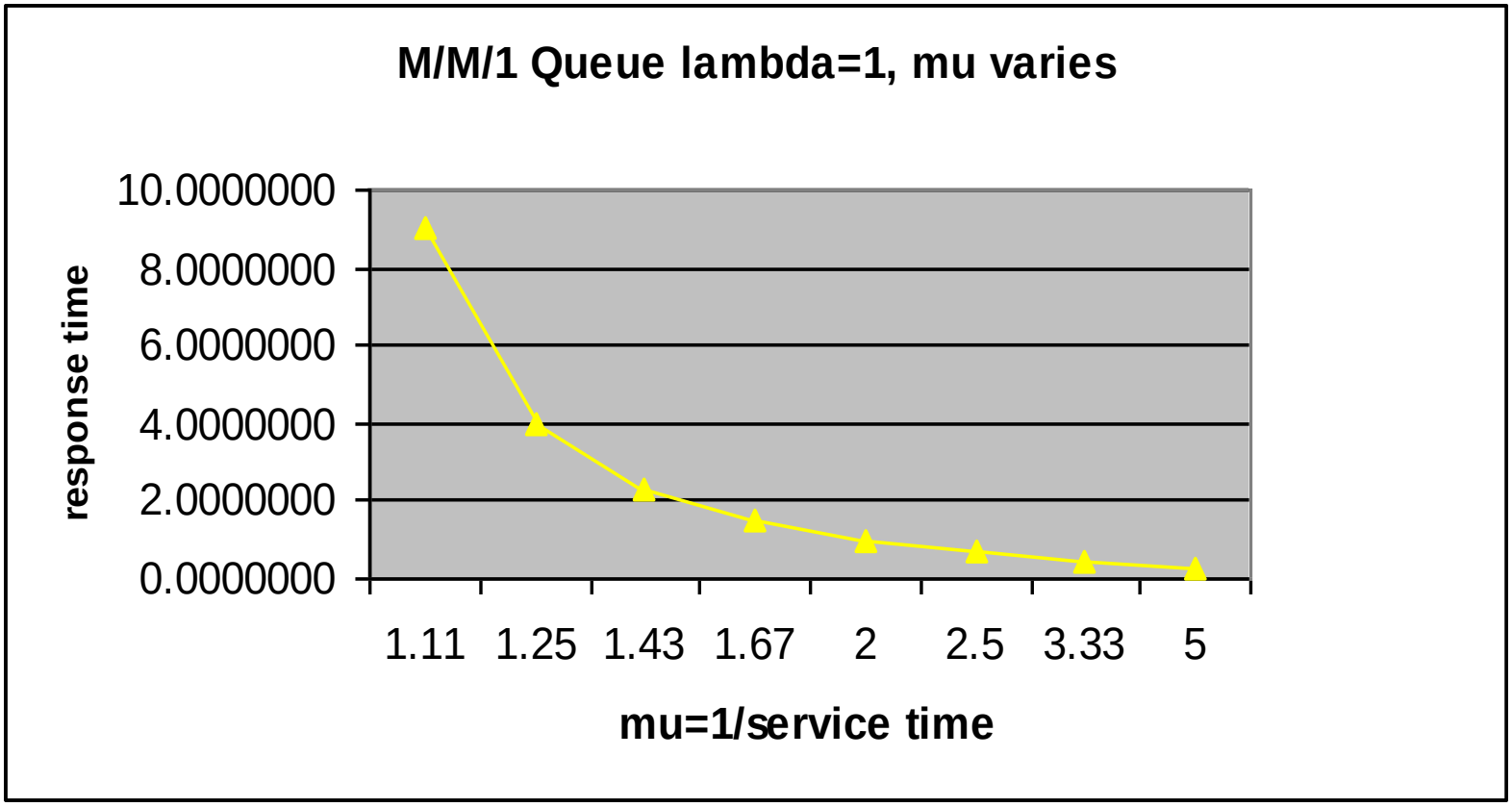
$$\overline{L} = \lambda \bullet \overline{W}$$

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

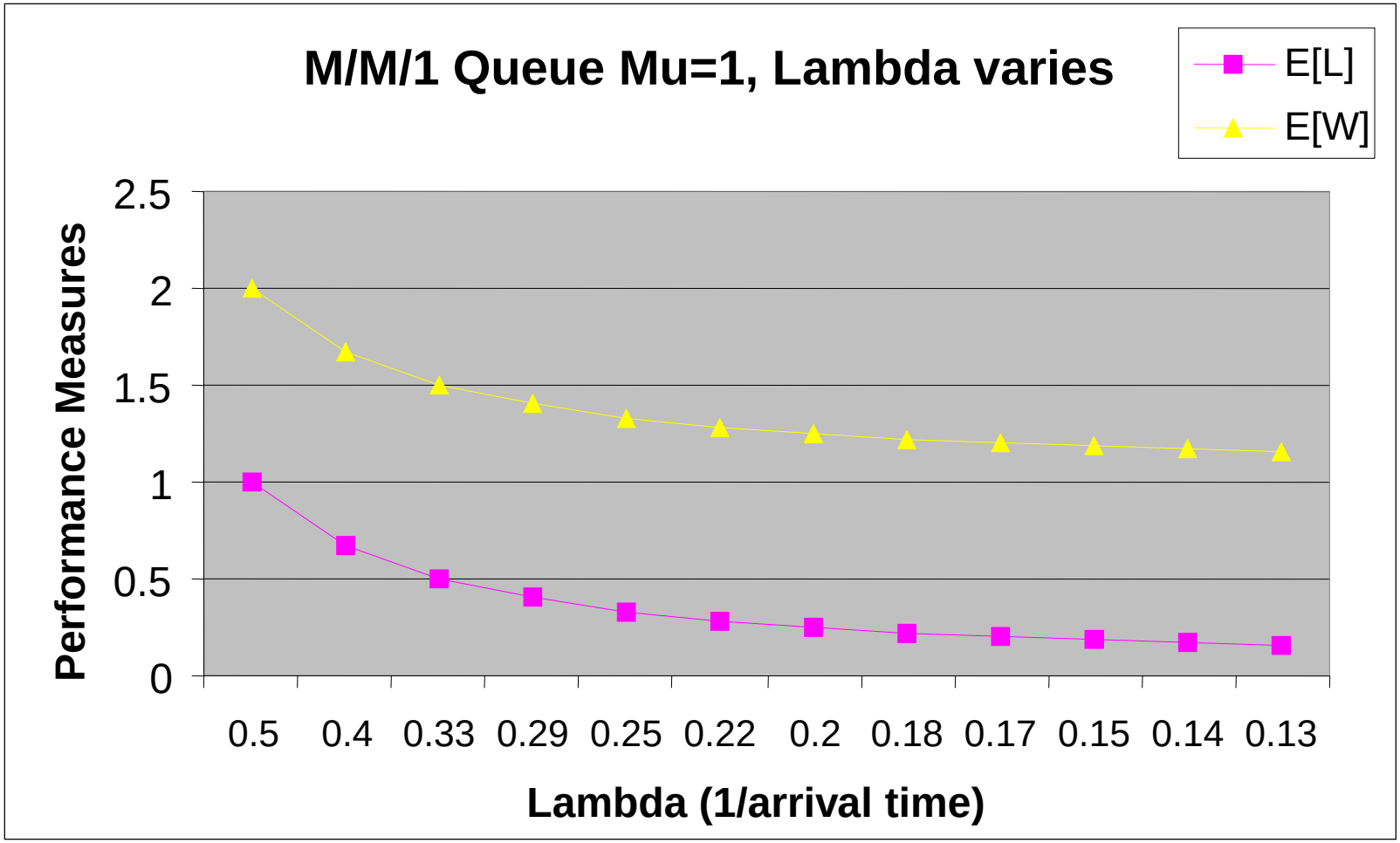
From previous discussion

Average Response Time: $$\frac{\rho}{1-\rho} \times \frac{1}{\lambda} = \frac{1}{\mu(1-\rho)}$$

M/M/1 Performance Figure 1

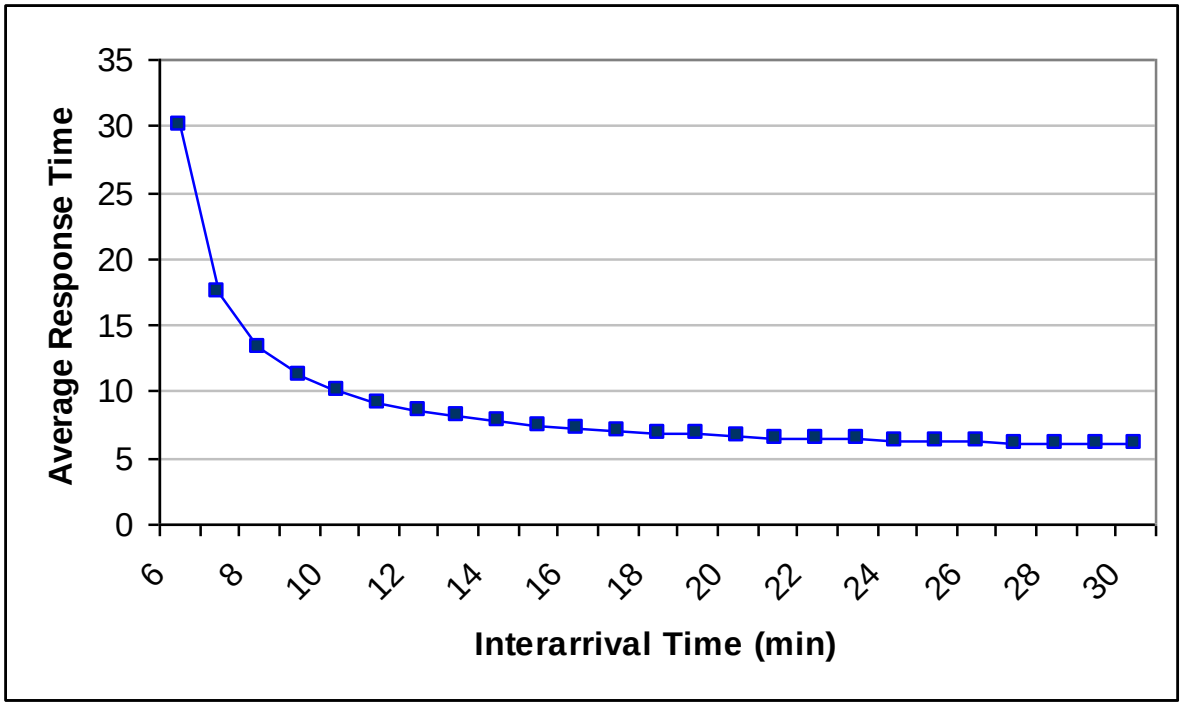


M/M/1 Performance Figure 2



Example: How does a “Fast Lane” work in Disneyland?

Construct an M/M/1 queue
Solve for the average response time, as a function of the inter-arrival time
Control the “people flow” to assure the response time



Properties of a Poisson processes

- Poisson process = exponential distribution between arrivals/departures/service

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

- Key properties:
  - memoryless
    - Past state does not help predict next arrival
  - Closed under:
    - Addition
    - Subtraction

A Special birth-death process:

Poisson Process

- $\lambda(n)=\lambda$ for all $n \geq 0$
- $\mu(n)=0$ for all $n \geq 0$
- Pure birth process
- Definition of the Poisson process:

  The counting process $\{N(t), t \geq 0\}$ is said to be a Poisson process having rate λ, $\lambda>0$, if
  - $N(0)=0$
  - The process has independent increments
  - The number of events in any interval of length t is Poisson distributed with mean λt.

Poisson Process (continued)

For all s, t ≥0

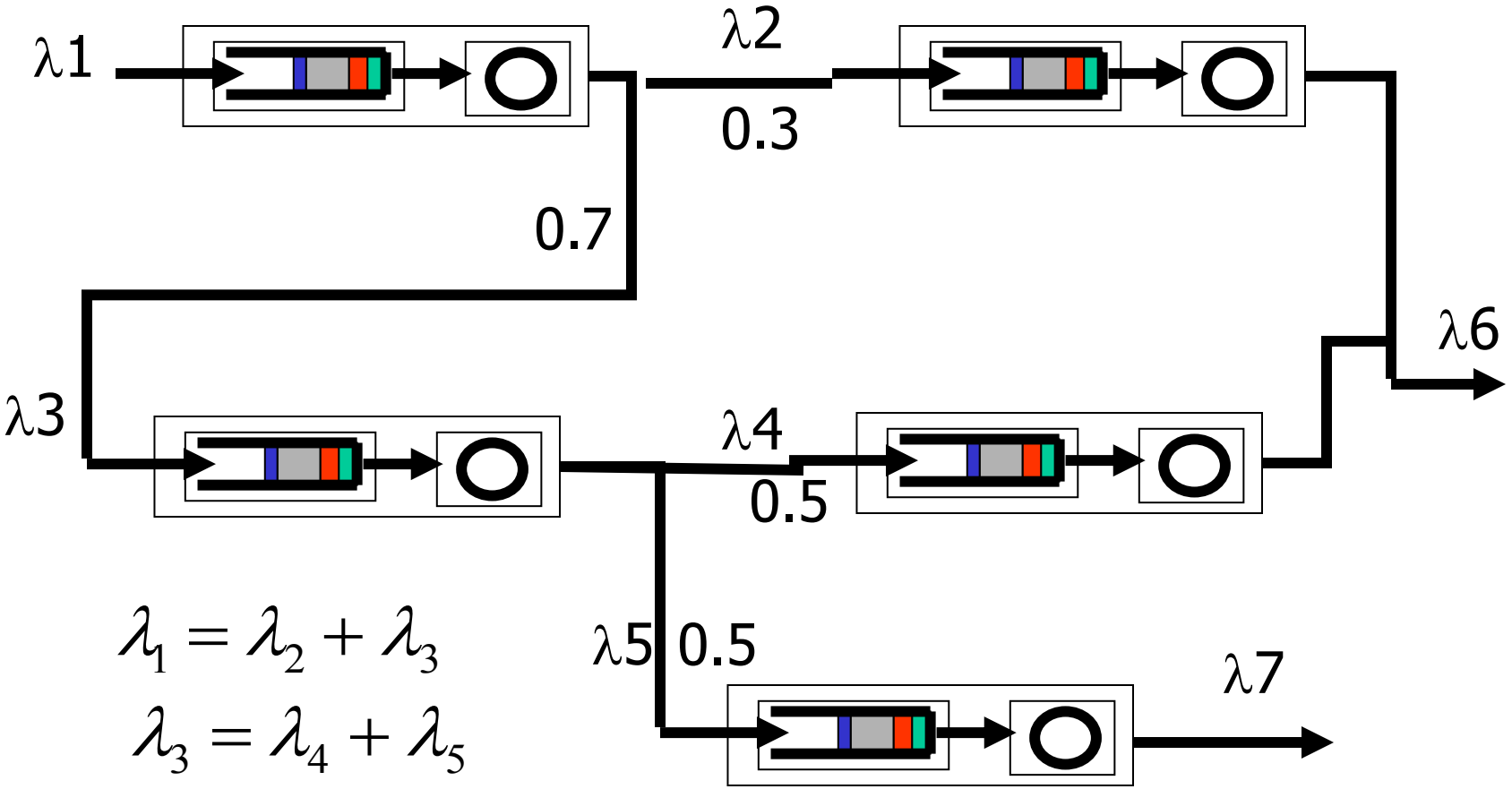
- P{N(t+s)-N(s) =n } = $e^{-\lambda t}\frac{(\lambda t)^n}{n!}$
- E[N(t)] = λ t
- Poisson properties:
  - Inter-arrival times are exponentially distributed
  - Memoryless property: the prob. of occurrence of an event is independent of how many events have occurred in the past and the time since the last event
  - …

Queuing Networks: Rate Addition and Subtraction

- Merge:
  - two Poisson streams with arrival rates λ_1 and λ_2:
    - new Poisson stream: $\lambda_3=\lambda_1+\lambda_2$
- Split :
  - If any given item has a probability P_1 of "leaving" the stream with rate λ_1:

    $\lambda_2=(1-P_1)\lambda_1$

Queuing Networks



$$\lambda_1 = \lambda_2 + \lambda_3$$

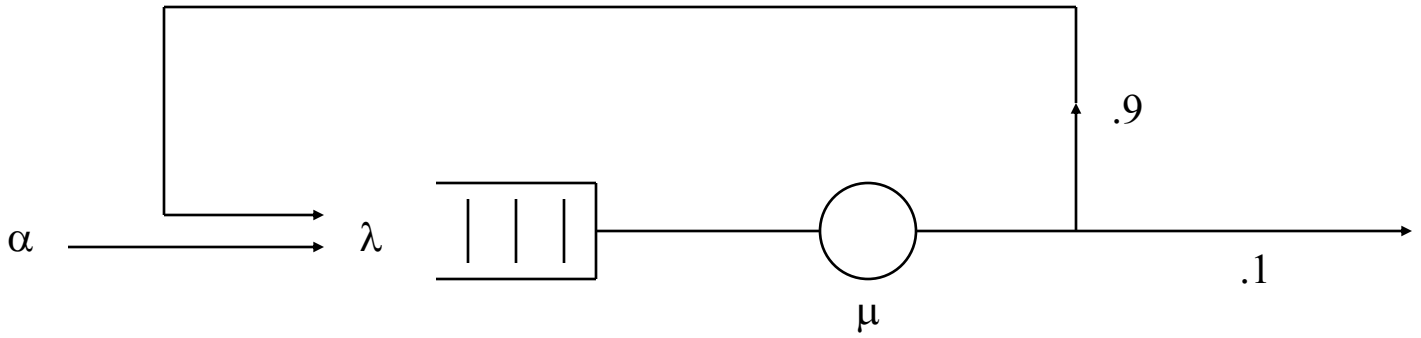
$$\lambda_3 = \lambda_4 + \lambda_5$$

$$\lambda_6 = \lambda_2 + \lambda_4$$

$$\lambda_7 = \lambda_5$$

Open Queuing Networks

- **Example:** Messages arrive at a concentrator according to a Poisson process of rate α. The time required to transmit a message and receive an acknowledgment is exponentially distributed with mean 1/m. Suppose that a message need to be retransmitted with probability p. Find the steady state pdf for the number of messages in the concentrator.
- **Solution:** The overall system can be represented by the simple queue with feedback shown below.



Open Queuing Networks

- The net arrival rate into the queue is $\lambda = \alpha + \lambda p$, that is,

$$\lambda = \frac{\alpha}{1-p}$$

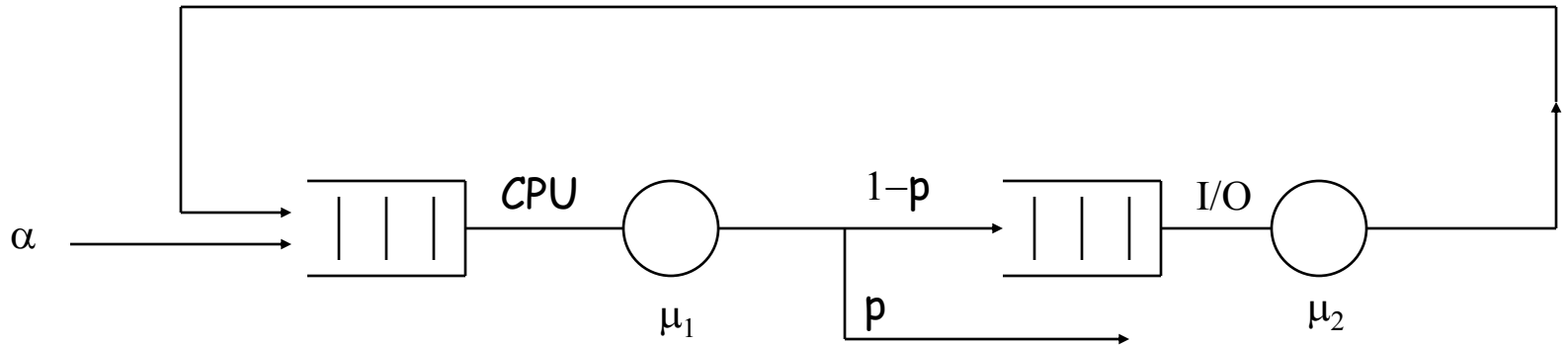
- Thus, the pdf for the number of messages in the concentrator is

$$P[N = n] = (1-\rho)\rho^n \qquad n = 0,1,\ldots$$

$$where \quad \rho = \lambda / \mu = \alpha /(1-\rho)\mu$$

Open Queuing Networks

- **Example:** New programs arrive at a CPU according to a Poisson process of rate α as shown in below fig. A program spends as exponentially distributed execution time of mean $1/\mu_1$ in the CPU. At the end of this service time, the program execution is complete with probability p or it requires retrieving additional information from secondary storage with probability 1-p. Suppose that the retrieval of information from secondary storage requires an exponentially distributed amount of time with mean $1/\mu_2$. Find the mean time that each program spends in the system.



Open Queuing Networks

- **Solution:** the net arrival rates into the two queues are

$$\lambda_1 = \alpha + \lambda_2 \quad and \quad \lambda_2 = (1-p)\lambda_1$$

- Thus

$$\lambda_1 = \frac{\alpha}{p} \quad and \quad \lambda_2 = \frac{(1-p)\alpha}{p}$$

- Each queue behaves like an M/M/1 system so,

$$E[L_1] = \frac{\rho_1}{1-\rho_1} \quad \text{and} \quad E[L_2] = \frac{\rho_2}{1-\rho_2}$$

Open Queuing Networks

- Little's formula then gives
  - $E[W_1]=1/\lambda_1 E[L_1]$, $E[W_2]=1/\lambda_2 E[L_2]$,

- Little's formula then gives the mean for total time spent in the system:
  - $E[W] = E[W_1] + E[W_2]$

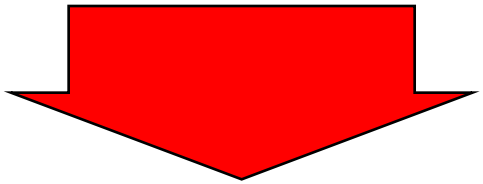
$$E[W] = \frac{(p/a)\rho_1}{1-\rho_1} + \frac{[p/a(1-p)]\rho_2}{1-\rho_2}$$

Applications: Queues and Traffic shaping

- Big Ideas:
  - Traffic shaping:
    - Modify traffic at entrance points in the network
    - Can reason about states of the interior and link properties, loads. (E.g. Leaky, Token bucket)
    - Modify traffic in the routers
      - Enforce policies on “flows”
  - Queuing theory:
    - Analytic analysis of properties of queues as functions of the inputs and service types

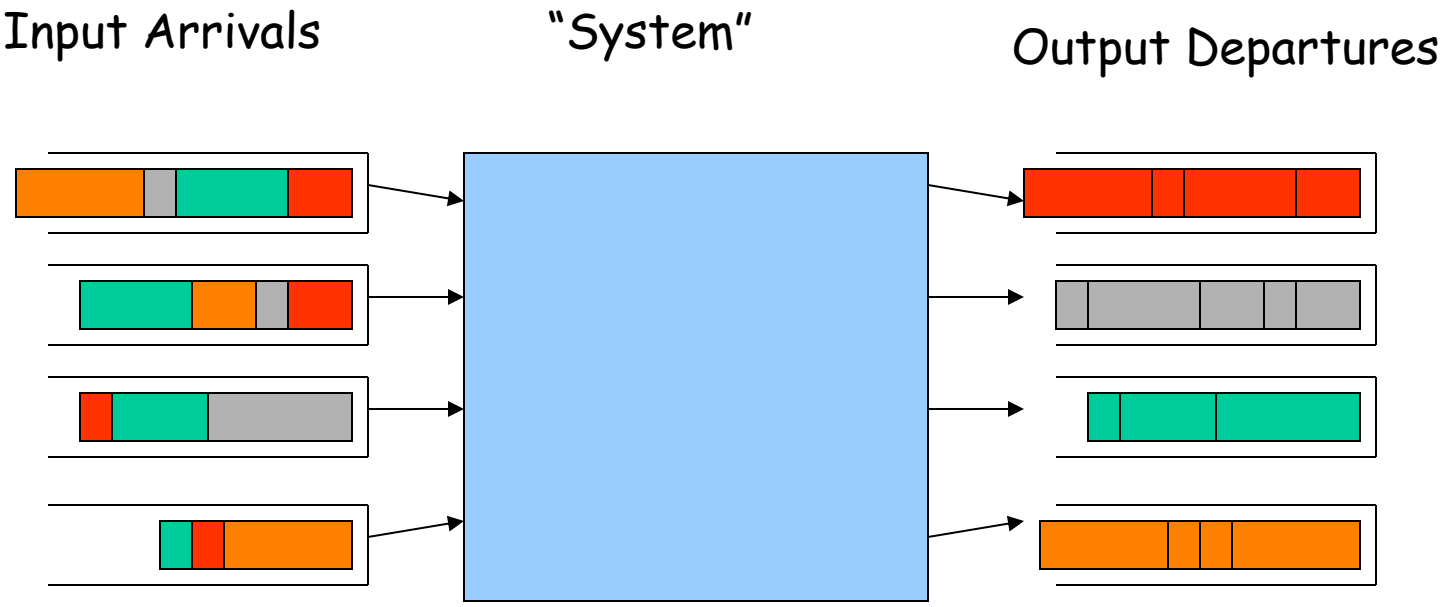
Congestion Control

Too many packets in some part of the system



Congestion

Simplified Network Model



Goal:

Move packets across the system from the inputs to output

System could be a single switch, or entire network

Problems with Congestion

- Performance
  - Low Throughput
  - Long Latency
  - High Variability (jitter)
  - Lost data
- Policy enforcement:
  - User X pays more than user Y, X should get more bandwidth than Y.
  - Fairness
    - One user/flow getting much more throughput or better latency than the other flows

Congestion Control

- Causes of congestion
- Types of congestion control schemes
- Solving congestion

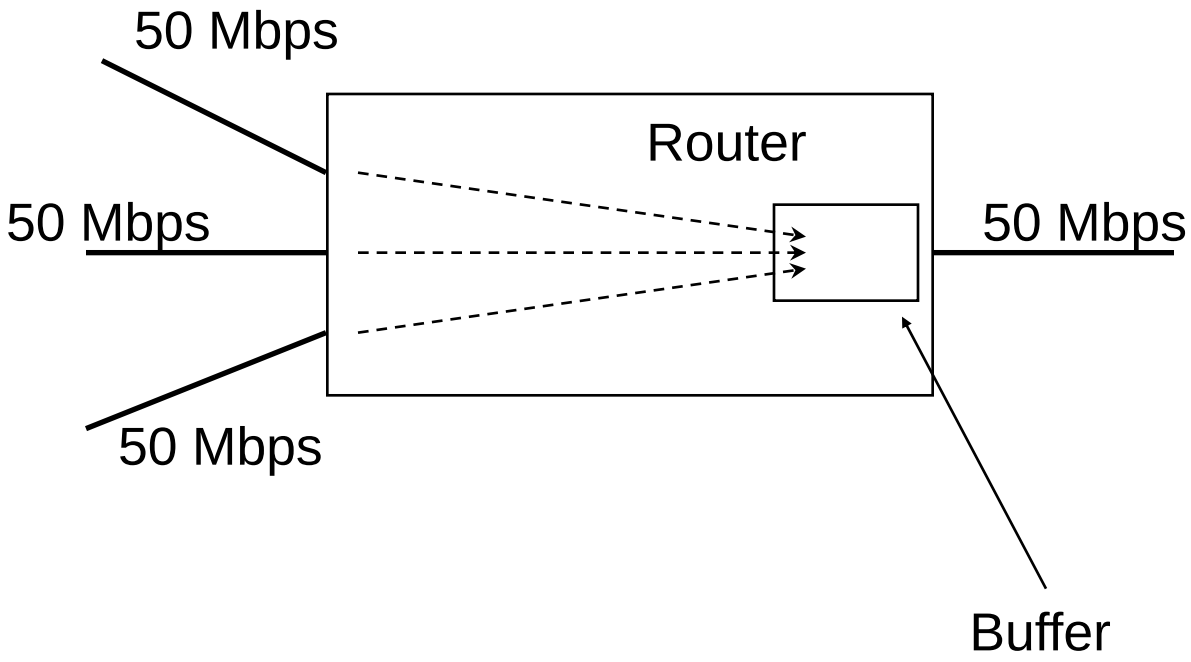
Causes of Congestion

- Many ultimate causes:

  New applications, new users, bugs, faults, viruses, spam, stochastic (time based) variations, unknown randomness.

- Congestion can be long-lived or transient
  - Timescale of the congestion is important
    - Microseconds vs seconds vs hours vs days
    - Different solutions to all the above!

Exhaustion of Buffer Space *(cont'd)*



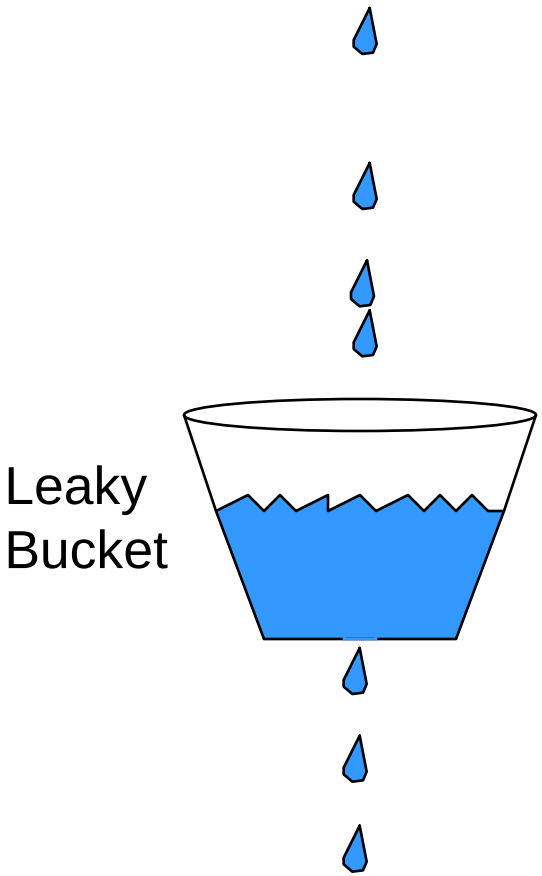
Types of Congestion Control Strategies

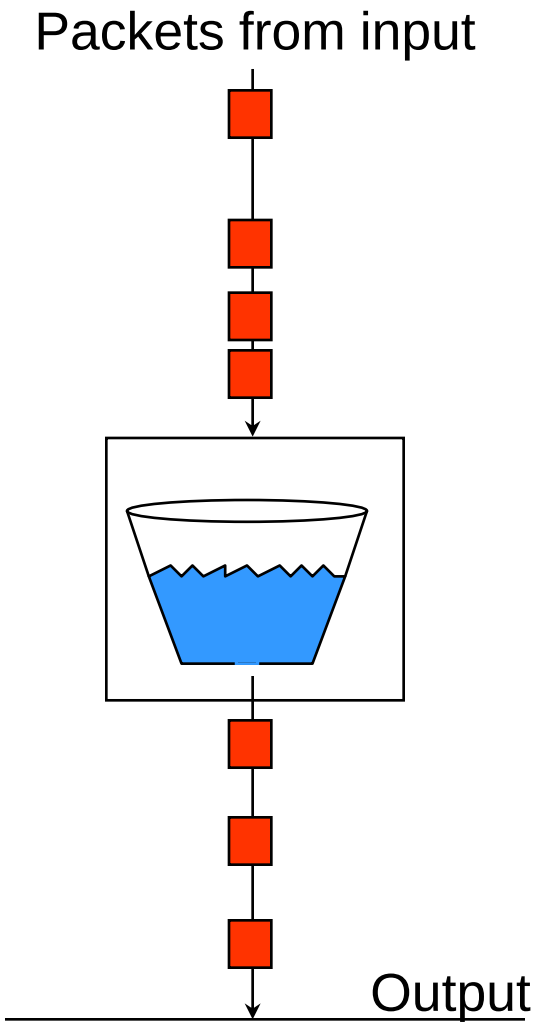
- Terminate existing resources
  - Drop packets
  - Drop circuits
- Limit entry into the system
  - Packet level (layer 3)
    - Leaky Bucket, token bucket, WFQ
  - Flow/conversation level (layer 4)
    - Resource reservation
    - TCP backoff/reduce window
  - Application level (layer 7)
    - Limit types/kinds of applications

Leaky Bucket

- Across a single link, only allow packets across at a constant rate
- Packets may be generated in a bursty manner, but after they pass through the leaky bucket, they enter the network evenly spaced
- If all inputs enforce a leaky bucket, its easy to reason about the total resource demand on the rest of the system

Leaky Bucket: Analogy





Leaky Bucket *(cont'd)*

- The leaky bucket is a "traffic shaper": It changes the characteristics of a packet stream
- Traffic shaping makes the network more manageable and predictable
- Usually the network tells the leaky bucket the rate at which it may send packets when a connection is established

Leaky Bucket: Doesn't allow bursty transmissions

- In some cases, we may want to allow short bursts of packets to enter the network without smoothing them out
- For this purpose we use a token bucket, which is a modified leaky bucket

Summary

- Key assumptions
  - Steady-state conditions
  - Exponential arrival and service rates
- Little's Law
- M/M/1 Queue
  - Fundamental equations for W, L
- Networks of Queues

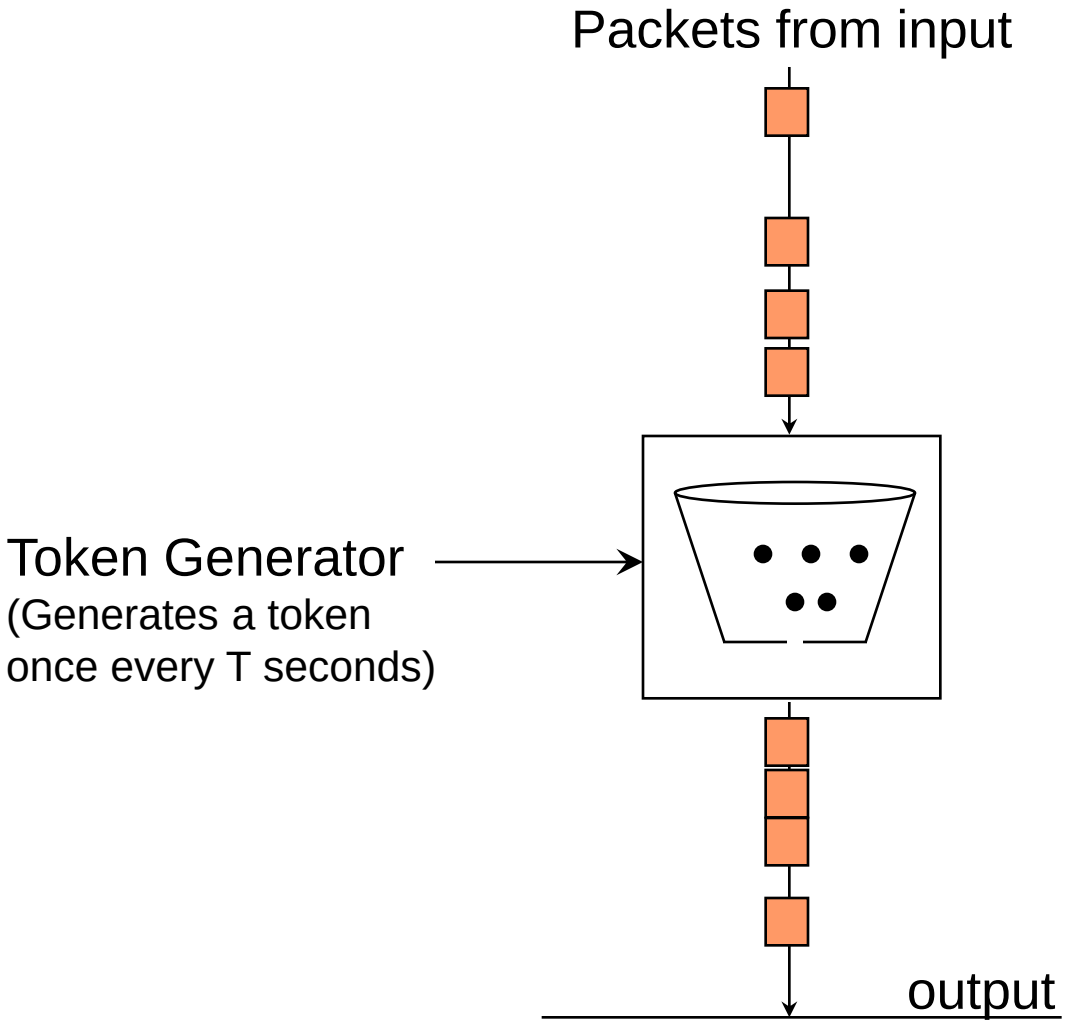
Addition Notes: Traffic Shaping

- Material is for general interest
- Will not be tested on this material

Token Bucket

- The bucket holds logical tokens instead of packets
- Tokens are generated and placed into the token bucket at a constant rate
- When a packet arrives at the token bucket, it is transmitted if there is a token available. Otherwise it is buffered until a token becomes available.
- The token bucket holds a fixed number of tokens, so when it becomes full, subsequently generated tokens are discarded
- Can still reason about total possible demand

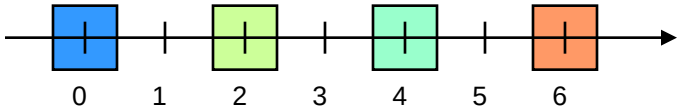
Token Bucket



Token Bucket vs. Leaky Bucket

Case 1: Short burst arrivals

Arrival time at bucket

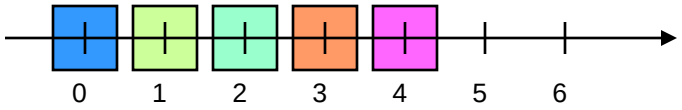


Departure time from a leaky bucket
Leaky bucket rate = 1 packet / 2 time units
Leaky bucket size = 4 packets

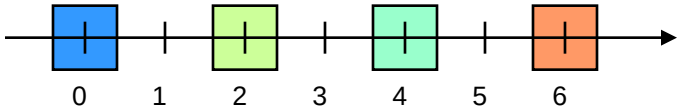
Departure time from a token bucket
Token bucket rate = 1 tokens / 2 time units
Token bucket size = 2 tokens

Token Bucket vs. Leaky Bucket

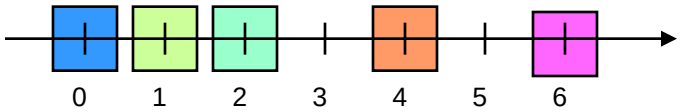
Case 2: Large burst arrivals



Arrival time at bucket



Departure time from a leaky bucket
Leaky bucket rate = 1 packet / 2 time units
Leaky bucket size = 2 packets



Departure time from a token bucket
Token bucket rate = 1 token / 2 time units
Token bucket size = 2 tokens

Multi-link congestion management

- Token bucket and leaky bucket manage traffic across a single link.
- But what if we do not trust the incoming traffic to behave?
- Must manage across multiple links
  - Round Robin
  - Fair Queuing

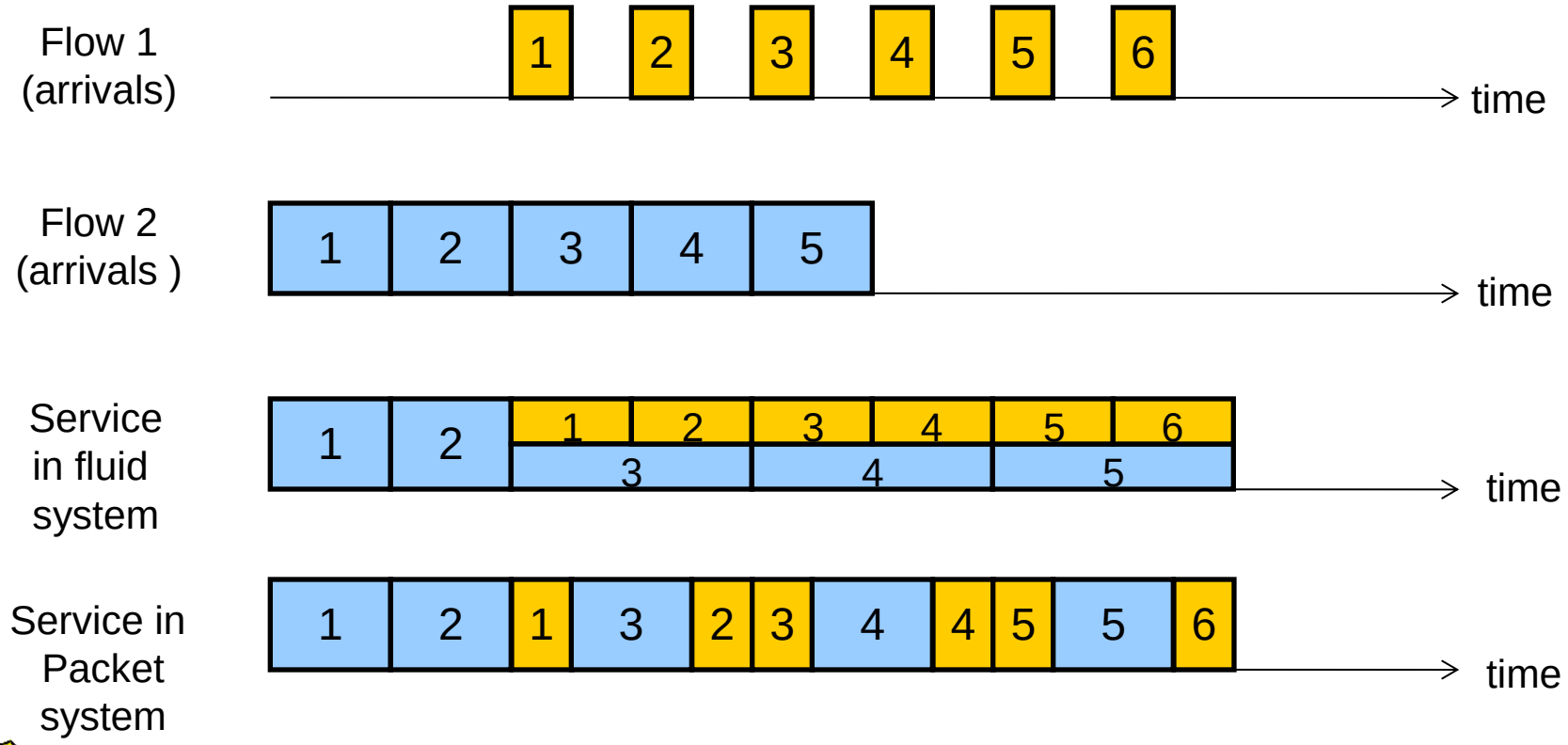
Multi-queue management

- If one source is sending too many packets, can we allow other sources to continue and just drop the "bad" source?
- First cut: round-robin
  - Service input queues in round-robin order
- What if one flow/link has all large packets, another all small packets?
  - Smaller packets get more link bandwidth

Idealized flow model

- For N sources, we would like to give each host or input source 1/N of the link bandwidth
- Image we could squeeze factions of a bits on the link at once
  - ->fluid model
  - E.g. “fluid” would interleave the bits on the link
  - But we must work with packets
- Want to approximate fairness of the fluid flow model, but still have packets

Fluid model vs. Packet Model



Fair Queuing vs. Round Robin

- Advantages: protection among flows
  - Misbehaving flows will not affect the performance of well-behaving flows
    - Misbehaving flow – a flow that does not implement congestion control
- Disadvantages:
  - More complex: must maintain a queue per flow per output instead of a single queue per output
  - Biased toward large packets – a flow receives service proportional to the number of packets

Virtual Time

- How to keep track of service delivered on each queue?
- Virtual Time is the number of rounds of queue service completed by a bit-by-bit Round Robin (RR) scheduler
  - May not be an integer
  - increases/decreases with # of active queues

Approximate bit-bit RR

Virtual time is incremented each time a bit is transmitted for all flows

- If we have 3 active flows, and transmit 3 bits, we increment virtual time by 1.
- If we had 4 flows, and transmit 2 bits, increment Vt by 0.5.

At each packet arrival, compute time packet would have exited the router during virtual time.

This is the packet finish number