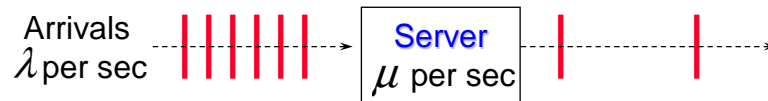


## Networked Multimedia: Priority, Rate control, Flow control

---



The author of these slides is Dr. Mark Pullen. Students registered Dr. Pullen's course may make a single machine-readable copy and print a single copy of each slide for their own reference, so long as each slide contains the copyright statement. Permission for any other use, either in machine-readable or printed form, must be obtained from the author in writing.

## Lecture Overview

---

- Queueing model
- Why is priority needed
- Rate control
- Flow control
- Fair queueing

## Queueing Theory

---

Queueing theory provides a mathematical basis for understanding and predicting the behavior of systems with discrete processes.

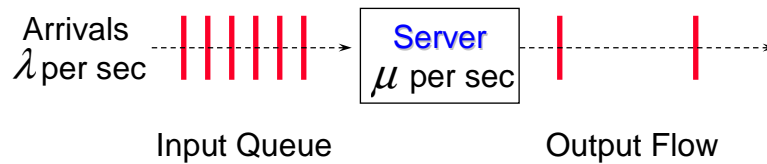
References:

1. Gross & Harris, *Fundamentals of Queueing Theory*, 3rd Ed., Wiley, 1998
2. Kleinrock, *Queueing Systems*, Wiley, 1975
3. Bertsekas & Gallager, *Data Networks*, 2nd Ed., Prentice-Hall, 1992

## Queueing Theory

Basic Model: Single Server M/M/1

---



Common assumptions (M/M/1 single-server queue):

Exponentially distributed inter-arrival time (Poisson arrivals)

Exponentially distributed service time, independent of arrivals

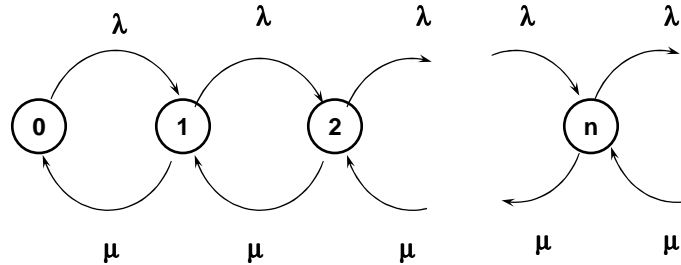
Mean arrival rate must be less than mean service rate!

These assumptions are often close to reality, generally make the analysis more tractable.

## Queueing Theory

### Basic Model: Single Server M/M/1

---

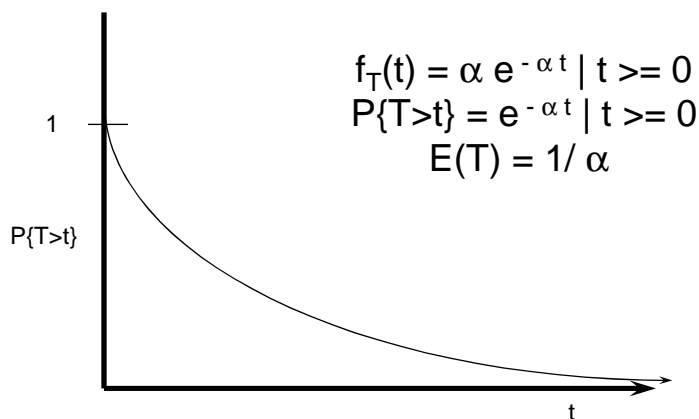


In steady state (requires  $\lambda < \mu$ ):

$$p_1 = (\lambda / \mu) p_0 \quad p_n = (\lambda / \mu)^n p_0 \quad p_0 = 1 - \lambda / \mu$$

## Exponential Distribution

---



$$f_T(t) = \alpha e^{-\alpha t} \mid t \geq 0$$

$$P\{T > t\} = e^{-\alpha t} \mid t \geq 0$$

$$E(T) = 1 / \alpha$$

## M/M/1 Queueing Formulas

---

Exponential distribution

$$f_T(t) = \alpha e^{-\alpha t} \mid t \geq 0$$

Utilization factor

$$\rho = \lambda / \mu \mid \lambda \leq \mu$$

Number in system

$$N = \rho / (1 - \rho) = \lambda / (\mu - \lambda)$$

this formula is only true for M/M/1

Time in system  
(Little's formula)

$$T = N / \lambda$$

Waiting time in queue

$$W = T - 1 / \mu$$

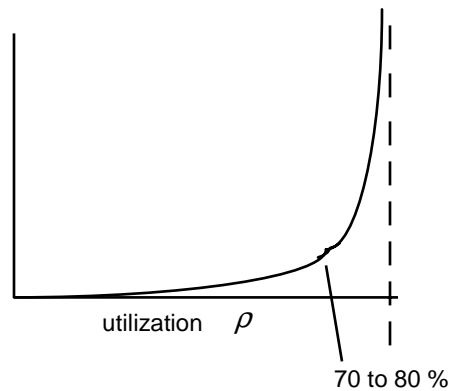
Number in queue

$$N_q = \lambda W$$

## M/M/1 Queueing System Performance

---

Number  
in  
system



$$\rho = \frac{\lambda}{\mu} = \frac{\text{average arrival rate}}{\text{average service rate}}$$

## M/M/1 Queueing Theory Example

---

An IP router is sending packets through a 64 kbps link.

- The length of the packets, when framed, is an exponential random variable with mean 400 bytes.
- The interval between arrival of packets (from the router's other inputs) is an exponential random variable with mean 15 packets per second.
- When a new packet is routed through this link and another packet is already being sent, the new packet is stored in a first-come first-served queue.

Calculate:

- average number of packets in the router in bytes
- average time from availability of a packet to be transmitted, until its last bit has been transmitted.

## M/M/1 Queueing Theory Example

---

Average number of packets in the router in bytes:

$$\lambda = 15 \text{ pps (given)}$$

$$\mu = 64000 / (400 * 8) = 20 \text{ pps}$$

$$\rho = \lambda / \mu = 15 / 20 = .75$$

$$N = \rho / (1 - \rho) = .75 / .25 = 3 \text{ packets}$$

Average time from availability of a packet to be transmitted, until its last bit has been transmitted:

$$T = N / \lambda = 3 / 15 = .2 \text{ s}$$

## M/M/1 Queueing Theory Example

continued

---

Repeat for average arrival rate 19 pps:

$$\rho = \lambda / \mu = 19 / 20 = .95$$

$$N = \rho / (1 - \rho) = .95 / .05 = 19 \text{ packets}$$

$$T = N / \lambda = 19 / 19 = 1 \text{ s}$$

Repeat for arrival rate 25:

The M/M/1 model is not useful, it only works when:  
arrival rate < service rate.

The queue buffer will fill up and the router will start dropping packets at a rate of 25-20=5 pps.

## A More General Result

### The Pollaczek- Khinchin (P- K) Formula

---

(M/G/1: any single server with Poisson arrivals)

Total time in a queueing node:

$$T = \bar{x} + \frac{\lambda \bar{x}^2}{2(1 - \rho)}$$

Where  $\bar{x}$  is average service time

$\lambda$  is average arrival rate

$\rho$  is utilization =  $\frac{\text{average arrival rate}}{\text{average service rate}}$

$\bar{x}^2$  is second moment of service (grows with variability of packet size)

## P-K Formula Example

---

Two packet systems are used to transmit real-time data.

- Both systems have the same Poisson-distributed input, and same processor and link capacity, limited by the outgoing link speed of 1.0 Mbps.
- Both systems have average packet length 1000 bits and average arrival rate 900 pps.
- System A does not constrain the packets, their length is exponentially distributed and service therefore has second moment  $2/\mu^2$ .
- System B has fixed-length packets and therefore has second moment  $1/\mu^2$ .

Find the mean time from arrival in queue to end of transmit for each system.

## P-K Formula Example

---

- Find the mean time from arrival in queue to end of transmit for each system:

$$\lambda = 900 \text{ pps} \quad \mu = 1000000 / 1000 = 1000 \text{ pps}$$

$$\bar{X} = 1 / \mu = .001 \text{ s} \quad \rho = \lambda / \mu = .9$$

$$\overline{X_A^2} = 2 / \mu^2 = .000002 \quad \overline{X_B^2} = 1 / \mu^2 = .000001$$

$$T_A = \bar{X} + (\lambda \overline{X^2}) / (2(1 - \rho)) = .001 + .009 = .010 \text{ s}$$

$$T_B = .001 + .0045 = .0055 \text{ s}$$

*A's delay is nearly twice as large!*

## Why Give Some Processes Priority?

---

- Some categories of data are more important to sustaining the virtual environment
  - if short term demand for processing or communication outstrips capacity, assign it to time-critical data first
  - for example, audio delivery in the virtual classroom
    - graphics are a second or two late, it will not be noticed
    - if there is a gap in the audio, it is noticed instantly and distracts from the learning process

## Why Priority? (2)

---

- Some processes can function reasonably well with open-loop estimators
  - if short term demand for processing or communication outstrips capacity, assign it to these processes last
  - for example, position of an object in a dead-reckoning system
    - strictly speaking, update needed only when the sender knows the dead-reckoned position will be wrong
    - visual anomaly is likely to be small if one or two updates are not sent
    - as number of skipped messages grows, the object's priority should grow

## Techniques for Defeating the Queueing Curve

---

- Processing
  - Operating system priority allocation
- Communication
  - Buffering: store data until capacity is available to send it
  - Jitter compensation: time-stamp data and delay all of it artificially at receiver for smooth flow
  - Traffic shaping
    - leaky bucket / token bucket
    - priority / weighted queueing

## Leaky Bucket

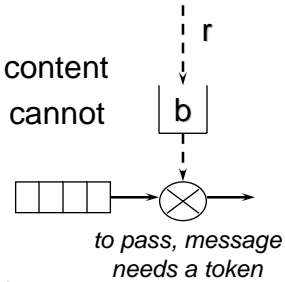
---

- Imagine a bucket that leaks at a certain rate
- For every unit volume leaked, one message may be sent
- This is a *fixed rate control* which allows traffic to pass at a given rate (if any is present)
  - deterministic flow minimizes  $\frac{x^2}{2}$  in P-K formula
  - system calculations are easy: add up the flows

## Token Bucket

- Improvement on on leaky bucket rate control\*
  - bucket starts out with  $b$  permits
  - permits arrive at rate  $r$  per second
  - every packet sent decrements bucket content
  - if bucket is empty, and arriving packet cannot be transmitted

- IntServ Token Bucket (RFC2215) implements this scheme
  - results in “rate-controlled” transmission



- \* see Bertsekas & Gallager *Data Networks 2nd ed*  
pp 511-512

## Closed Loop Flow Control

- Rate control is open-loop and conservative
- What if there are no competing messages?
  - don't need to limit transmission
- To find out, use a closed-loop control
  - receiver sends permit to sender when it is ready for more data
  - slowdown in round-trip time implies congestion
  - closed-loop system has the effect of reducing traffic, thus limits congestion
  - this technique is used by TCP but also can be used elsewhere in message-based systems

## Fair Queueing

- Depending on the situation, might count messages per unit time or total data per unit time in determining fairness
- When dealing with multiple sources of the same priority:
  - fairness may require that each source is able to send the same number of bytes/s rather than the same number of messages/s
  - if so, count bytes in the queue and allow each sender to transmit enough messages to equal the longest front-of-queue message
- **Weighted Fair Queueing**: variation of this where the sources are allowed to send at different rates
  - apply a weighting multiplier to the number of bytes in each queue

