Electrical Engineering E6761
Computer Communication Networks
Lecture 5
Routing:
Router Internals, Queueing

Professor Dan Rubenstein
Tues 4:10-6:40, Mudd 1127
Course URL:
http://www.cs.columbia.edu/~danr/EE6761

Overview

- Finish Last time: TCP latency modeling
- Queueing Theory
  - Little's Law
  - Poisson Process / Exponential Distribution
  - A/B/C/D (Kendall) Notation
  - M/M/1 & M/M/1/K properties
- Queueing "styles"
  - scheduling: FIFO, Priority, Round-Robin, WFQ
  - policing: leaky-bucket
- Router Components / Internals
  - ports, switching fabric, crossbar
  - IP lookups via tries

TCP latency modeling

Q: How long does it take to receive an object from a Web server after sending a request?

- TCP connection establishment
- data transfer delay

Notation, assumptions:

- Assume one link between client and server of rate R
- Assume: fixed congestion window, W segments
- S: MSS (bits)
- O: object size (bits)
- no retransmissions (no loss, no corruption)

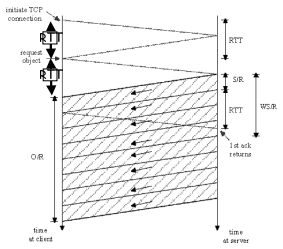
S/R = time to send a packet's bits into the link

Two cases to consider:

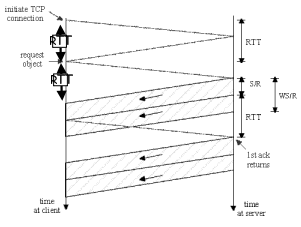
- WS/R > RTT + S/R: ACK for first segment in window returns before window's worth of data sent
- WS/R < RTT + S/R: wait for ACK after sending window's worth of data sent

TCP latency Modeling

K:=⌈O/WS⌉ = # of windows needed to fit object



Case 1: latency = 2RTT + O/R



Case 2: latency = 2RTT + O/R + (K-1)[S/R + RTT - WS/R]

idle time bet. window transmissions

TCP Latency Modeling: Slow Start

- Now suppose window grows according to slow start.
- Will show that the latency of one object of size O is:

$$Latency = 2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}$$

where P is the number of times TCP stalls at server:

$$P = \min\{Q, K-1\}$$

- where Q is the number of times the server would stall if the object were of infinite size.

- and K is the number of windows that cover the object.

TCP Latency Modeling: Slow Start (cont.)

$\frac{S}{R} + RTT$ = time from when server starts to send segment until server receives acknowledgement

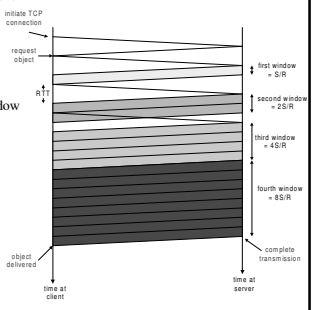
$2^{k-1}\frac{S}{R}$ = time to transmit the kth window

$\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]^+$ = stall time after the kth window

$$\text{latency} = \frac{O}{R} + 2RTT + \sum_{p=1}^{P} stallTime_p$$

$$= \frac{O}{R} + 2RTT + \sum_{k=1}^{P}[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}]$$

$$= \frac{O}{R} + 2RTT + P[RTT + \frac{S}{R}] - (2^P - 1)\frac{S}{R}$$



TCP Latency Modeling: Slow Start (cont.)

Example:

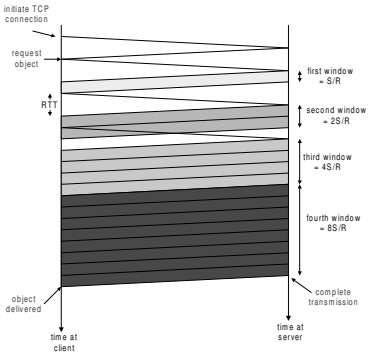
O/S = 15 segments

K = 4 windows

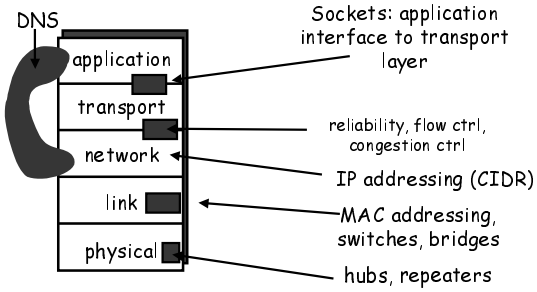
Q = 2

P = min{K-1,Q} = 2

Server stalls P=2 times.



What we've seen so far (layered perspective)...



Today: part 1 of network layer: inside a router

Queueing, switching, lookups

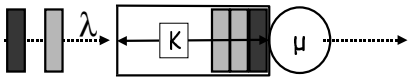
Queueing

- 3 aspects of queueing in a router:
  - arrival rate and service time distributions for traffic
  - scheduling: order of servicing pkts in queue(s)
  - policing: admission policy into queue(s)

9

Model of a queue

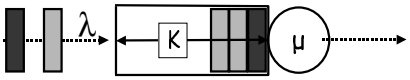
Queuing model (a single router or link)



- Buffer of size K (# of customers in system)
- Packets (customers) arrive at rate λ
- Packets are processed at rate μ
- λ and μ are **average** rates

10

Queues: General Observations



- Increase in λ leads to more packets in queue (on average), leads to longer delays to get through queue
- Decrease in μ leads to longer delays to get processed, leads to more packets in queue
- Decrease in K:
  - packet drops more likely
  - less delay for the "average" packet accepted into the queue

11

Little's Law (a.k.a. Little's Theorem)

- Let p_i be the i^{th} packet into the queue
- Let N_i = # of pkts already in the queue when p_i arrives
- Let T_i = time spent by p_i in the system: includes
  - time sitting in queue
  - time it takes processor to process p_i
- If K = ∞ (unlimited queue size) then

$$\lim_{i\to\infty} E[N_i] = \lambda \lim_{i\to\infty} E[T_i]$$

Holds for any distribution of λ, μ (which means for any distribution of T_i as well)!!

12

Little's Law: examples

- People arrive at a bank at an avg. rate of 5/min. They spend an average of 20 min in the bank. What is the average # of people in the bank at any time?

  λ=5, T=20, E[N] = λ E[T] = 5(20) = 100

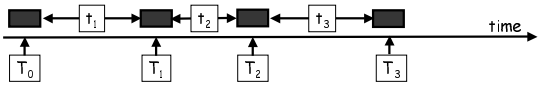
- To keep the average # of people under 50, how much time should be spent by customers on average in the bank?

  λ=5, E[N] < 50, E[T] = E[N] / λ < 50 / 5 = 10

13

Poisson Process / Exponential Distribution

- Two ways of looking at the same set of events



- $\{T_i\}$ = times of packet arrivals are "described by a Poisson process"
- $\{t_i\}$ = time between arrivals are "exponentially distributed"
- The process / distribution is special because it's memoryless:
  - observing that an event hasn't yet occurred doesn't increase the likelihood of it occurring any sooner
  - observing "resets" the state of the system

14

Memorylessness

- An example of a memoryless R.V., T
  - Let T be the time of arrival of a memoryless event, E
  - Choose any constant, D
  - P(T > D) = P(T > x+D | T > x) for any x
- We "checked" to see if E occurred before y and found out that it didn't
  - Given that it did not occur before time x, the likelihood that it now occurs by time D+x is the same as if the timer just started and we're only waiting for time D

15

Which are memoryless?

- The time of the first head for a fair coin
  - tossed every second — Yes (for discrete time units)!

    P(T > D) = P(T > D+x | T>x) for x an integer
  - tossed 1/n seconds after the n^{th} toss

    No (e.g., P(T>1) = .5, P(T>2 | T>1) = .25)
- The on-time arrival of a bus
  - arriving uniformly between 2 and 3pm

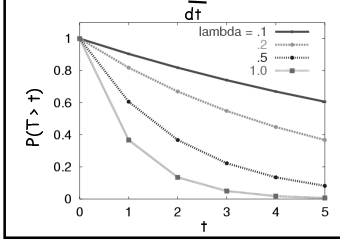
    No (e.g., P(T>2:30 = .5), P(T>3:00 | T>2:30) = 0
  - if $P(T > D) = 1 / 2^D$

    Yes: $P(T>D+x \mid T>x) = (1/2^{D+x}) / (1/2^x) = 1 / 2^D$

16

The exponential distribution

- If T is an exponentially distributed r.v. with rate λ, then $P(T > t) = e^{-\lambda t}$, hence:
  - $P(T < t) = 1 - e^{-\lambda t}$
  - $P(T > t+x \mid T > x) = e^{-\lambda t}$ (memoryless)
  - $\text{dens}(T=t) = \frac{d}{dt} P(T>t) = -\lambda e^{-\lambda t}$



Note bounds:

$P(T > 0) = 1$

$\lim_{t \to \infty} P(T > t) = 0$

Expo Distribution: useful facts

- Let green packets arrive as a Poisson process with rate λ_1, and red packets arrive as a Poisson process with rate λ_2
  - (green + red) packets arrive as a Poisson process with rate $\lambda_1 + \lambda_2$
  - P(next pkt is red) = $\lambda_1 / (\lambda_1 + \lambda_2)$
- Q: is the aggregate of n Poisson processes a Poisson process?
- PASTA (Poisson Arrivals See Time Averages)
  - P(system in state X when Poisson arrival arrives) = E[state of system]
  - Why? due to memorylessness
  - Note: rarely true for other distributions!!

What about 2 Poisson arrivals?

- Let T_i be the time it takes for i Poisson arrivals with rate λ to occur. Let t_i be the time between arrivals i and i-1 (where $t_0 = T_1$)
- $P(T_2>t) = P(t_0>t) + \int_{x=0}^{t} \text{dens}(t_0=x)\, P(t_1 > t-x)\, dx$

  $= e^{-\lambda t} + \int_{x=0}^{t} -\lambda e^{-\lambda t} \cdot e^{-\lambda(t-x)}\, dx$

  $= e^{-\lambda t}(1 + \lambda t)$
- Note: T_2 is <u>not</u> a memoryless R.V.:

  $P(T_2 > t \mid T_2 > s) = e^{-\lambda(t-s)}(1 - \lambda t) / (1 - \lambda s)$

  $P(T_2 > t-s) = e^{-\lambda(t-s)}(1 - \lambda(t-s))$

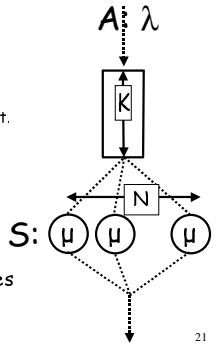
What about n Poisson arrivals?

- Let N(t) be the number of arrivals in time t
- $P(N(t) = 0) = P(T_1 > t) = e^{-\lambda t}$
- $P(N(t) = 1) = P(T_2 > t) - P(T_1 > t) = e^{-\lambda t}(1 + \lambda t) - e^{-\lambda t} = \lambda t e^{-\lambda t}$
- $P(N(t) = n) = \int_{x=0}^{t} \text{dens}(N(x)=n-1) P(N(t-x)=1)\, dx$
- Solving gives $P(N(t) = n) = (\lambda t)^n e^{-\lambda t}/n!$
- So $P(T_n > t) = \sum_{i=0}^{n-1} P(N(t) = i)$

A/S/N/K systems (Kendall's notation)

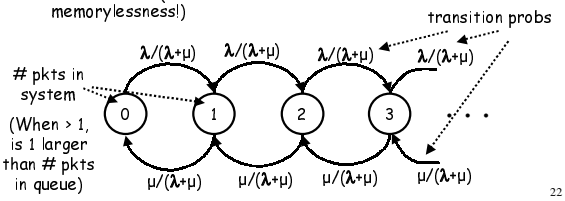
A/S/N/K gives a theoretical description of a system

- A is the arrival process
  - M = Markovian = Poisson Arrivals
  - D = deterministic (constant time bet. arrivals)
  - G = general (anything else)
- S is the service process
  - M,D,G same as above
- N is the number of parallel processors
- K is the buffer size of the queues
  - K term can be dropped when buffer size is infinite



The M/M/1 Queue (a.k.a., birth-death process)

- a.k.a., M/M/1/∞
  - Poisson arrivals
  - Exponential service time
  - 1 processor, infinite length queue
- Can be modeled as a Markov Chain (because of memorylessness!)
- Distribution of time spent in state n the same for all n > 0 (why? why different for state 0?)



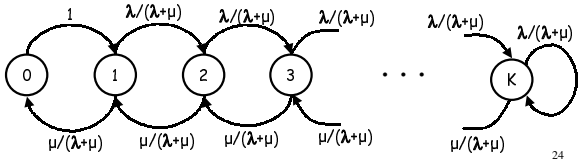
M/M/1 cont'd

As long as $\lambda < \mu$, queue has following steady-state average properties

- Defs:
  - $\rho = \lambda/\mu$
  - N = # pkts in system
  - T = packet time in system
  - N_Q = # pkts in queue
  - W = waiting time in queue
- $P(N=n) = \rho^n(1-\rho)$
  - (indicates fraction of time spent w/ n pkts in queue)
  - Utilization factor = 1 - P(N=0) = ρ
- $E[N] \equiv \sum_{n=0}^{\infty} n\,P(N=n) = \rho/(1-\rho)$
- $E[T] = E[N] / \lambda$ (Little's Law) $= \rho/(\lambda(1-\rho)) = 1/(\mu - \lambda)$
- $E[N_Q] = \sum_{n=1}^{\infty} (n-1)\,P(N=n) = \rho^2/(1-\rho)$
- $E[W] = E[T] - 1/\mu$ (or $= E[N_Q]/\lambda$ by Little's Law) $= \rho/(\mu - \lambda)$

M/M/1/K queue

- Also can be modeled as a Markov Model
  - requires K+1 states for a system (queue + processor) that holds K packets (why?)
  - Stay in state K upon a packet arrival
  - Note: $\rho \geq 1$ permitted (why?)



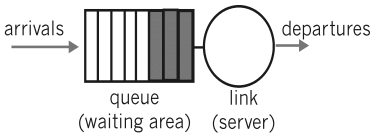
M/M/1/K properties

- $P(N=n) = \begin{cases} \rho^n(1-\rho) / (1 - \rho^{K+1}), & \rho \neq 1 \\ 1 / (K+1), & \rho = 1 \end{cases}$
- $E[N] = \begin{cases} \rho/((1-\rho)(1 - \rho^{K+1})), & \rho \neq 1 \\ 1 / (K+1), & \rho = 1 \end{cases}$
- i.e., divide M/M/1 values by $(1 - \rho^{K+1})$

Scheduling And Policing Mechanisms

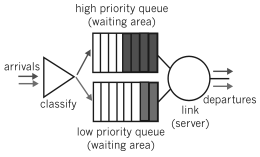
Scheduling: choosing the next packet for transmission on a link can be done following a number of policies;

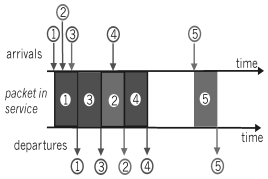
- FIFO (First In First Out) a.k.a. FCFS (First Come First Serve): in order of arrival to the queue
  - packets that arrive to a full buffer are discarded
  - another option: discard policy determines which packet to discard (new arrival or something already queued)



Scheduling Policies

- Priority Queuing:
  - Classes have different priorities
  - May depend on explicit marking or other header info, eg IP source or destination, TCP Port numbers, etc.
  - Transmit a packet from the highest priority class with a non-empty queue

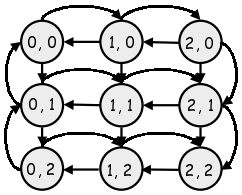




Scheduling Policies

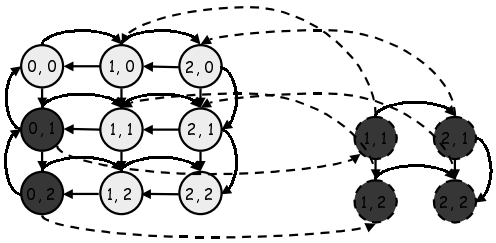
- Priority Queueing cont'd:
  - 2 versions:
    - Preemptive: (postpone low-priority processing if high-priority pkt arrives)
    - non-preemptive: any packet that starts getting processed finishes before moving on

Modeling priority queues as M/M/1/K



- preemptive version (K=2): assuming preempted packet placed back into queue
  - state w/ x,y indicates x priority queued, y non-priority queued
  - what are the transition probabilities?
  - what if preempted is discarded?

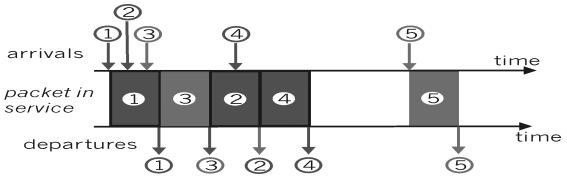
Modeling priority queues as M/M/1/K



- Non-preemptive version (K=2)
  - yellow (solid border) = nothing or high-priority being proc'd
  - red (dashed border) = low-priority being processed
  - what are the transition probabilities?

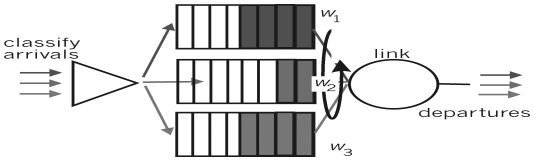
Scheduling Policies (more)

- Round Robin:
  - each flow gets its own queue
  - circulate through queues, process one pkt (if queue non-empty), then move to next queue



Scheduling Policies (more)

- Weighted Fair Queuing: is a generalized Round Robin in which an attempt is made to provide a class with a differentiated amount of service over a given period of time



WFQ details

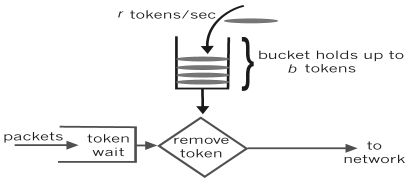
- Each flow, i, has a weight, $W_i > 0$
- A Virtual Clock is maintained: V(t) is the "clock" at time t
- Each packet k in each flow i has
  - virtual start-time: $S_{i,k}$
  - virtual finish-time: $F_{i,k}$
- The Virtual Clock is restarted each time the queue is empty
- When a pkt arrives at (real) time t, it is assigned:
  - $S_{i,k} = \max\{F_{i,k-1}, V(t)\}$
  - $F_{i,k} = S_{i,k} + \text{length}(k) / W_i$
  - $V(t) = V(t') + (t-t') / \sum_{B(t',t)} W_j$
    - t' = last time virtual clock was updated
    - B(t',t) = set of sessions with pkts in queue during (t',t]

Policing Mechanisms

- Three criteria:
  - (Long term) **Average Rate** (100 packets per sec or 6000 packets per min??), crucial aspect is the interval length
  - **Peak Rate**: e.g., 6000 p p minute Avg and 1500 p p sec Peak
  - (Max.) **Burst Size**: Max. number of packets sent consecutively, ie over a short period of time

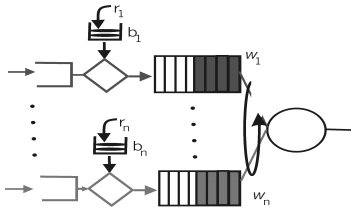
Policing Mechanisms

- Token Bucket mechanism, provides a means for limiting input to specified Burst Size and Average Rate.



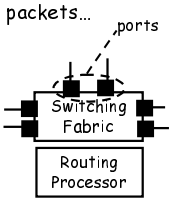
Policing Mechanisms (more)

- Bucket can hold b tokens; token are generated at a rate of *r token/sec* unless bucket is full of tokens.
- Over an interval of length *t*, the number of packets that are admitted is less than or equal to *(r t + b)*.
- Token bucket and WFQ can be combined to provide upper bound on delay.

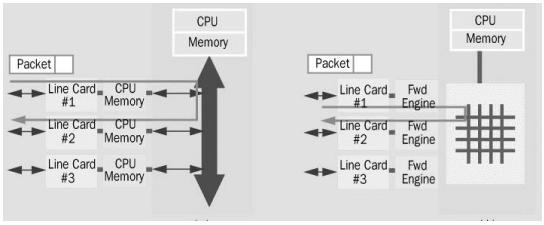


Routing Architectures

- We've seen the queueing policies a router can implement to determine the order in which it services packets
- Now let's look at how routers service packets…
- A router consists of
  - ports: connections to wires to other network entities
  - switching fabric: a "network" inside the router that transfers packets between ports
  - routing processor: brain of the router
    - maintains lookup tables
    - in some cases, does lookups



Router Archs (high end today)



High End:

Each interface has its own CPU

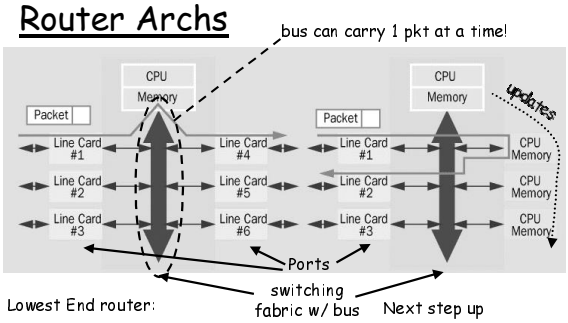
lookup done before using bus → 1 pass on bus

Highest:

Interface's processing done in hardware

Crossbar switch can deliver pkts simultaneously

Router Archs



Lowest End router:

all packets processed by 1 CPU, share the same bus

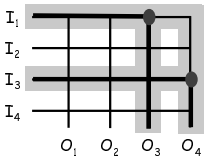
2 passes on the bus per pkt

Next step up

pool of CPUs (still have shared bus, 2 passes per pkt)

main CPU keeps pool up-to-date

Crossbar Architecture

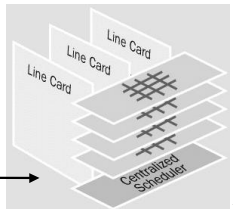


- To complete transfer from I_x to O_y, close crosspoint at (x,y)
- Can simultaneously transfer pairs with differing input and output ports
- multiple crossbars can be used at once

$I_1 \rightarrow O_3$

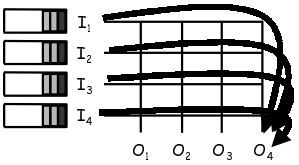
$I_3 \rightarrow O_4$

$I_2 \rightarrow O_3$ WAIT!!



Head-of-line Blocking

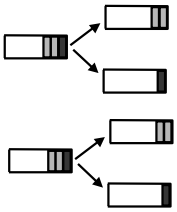
- How to get packets with different input/output port pairings to the cross bar at the same time



- Problem: what if 1st pkt in every input queue wants to go to the same output port?
- Packets at the head of the line are blocking packets deeper in queue from being serviced

41

Virtual Output Queueing



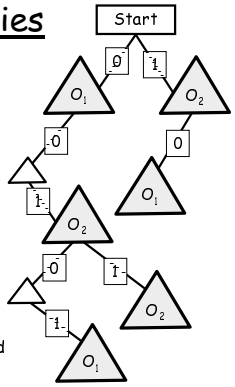
- Each input queue is split into separate virtual queues for each output port
- Central scheduler can choose a pkt to each output port (at most one per input port per round)

Q: how do routers know where to send pkt to?

42

Fast IP Lookups: Tries

- Task: choose the appropriate output port
- Given: router stores longest matching prefixes
- Goal: quickly identify to which outgoing interface packet should be sent
- Data structure: Trie
  - a binary tree
  - some nodes marked by an outgoing interface
  - i^{th} bit is 0 take i^{th} step left
  - i^{th} bit is 1 take i^{th} step right
  - keep track of last interface crossed
  - no link for step, return last interface



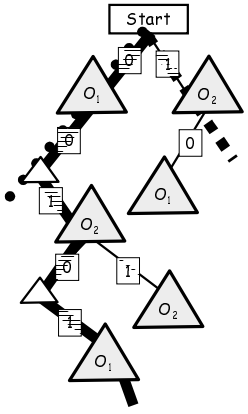
43

Trie example

- Lookup Table:

Prefix	Interface
0	O_1
1	O_2
10	O_1
001	O_2
00101	O_1
0011	O_2

- Examples:
  - 0001010
  - 110101
  - 00101011



44

Next time…

- Routing Algorithms
  - how to determine which prefix is associated with which output port(s)

45