

## 1 ☐ EECS 122, Lecture 26

Today's Topics:

Introduction to Quality of Service

Traffic Regulation and Scheduling

Kevin Fall, [kfall@cs.berkeley.edu](mailto:kfall@cs.berkeley.edu)

## 2 ☐ Another Look at Sharing

- Computer network are shared resources, leading to contention
- By specific allocation of resources to entities (individuals, traffic classes), can adjust the quality of service provided
- QoS important in an economically-driven environment with limited resources
- Upshot: more \$ will probably get you better QoS

## 3 ☐ Problem Set #5 (the last)

- Peterson & Davie
  - Ch 8: 16, 19
  - Ch 9: 15, 16
  - Reading: p.368-378, 382-386
  - Ch 7: 21, 22
  - Due May 6

## 4 ☐ Why QoS Anyhow?

- Assumption of 2 types of applications:
  - elastic (adapt to available resources)
  - guaranteed/in-elastic (non-adaptable)

- Service provided
  - elastic apps are usually ok with best-effort service model
  - in-elastic apps may be unusable if not given enough bandwidth or too high delay (audio)

## 5 ☐ Basic Service Model

- Assume that a server provides access to a resource, and incoming requests are queued in a service queue
- Requests may contain traffic descriptors describing desired Quality of Service
- Servers use a scheduling discipline to determine which request to provide service to next

## 6 ☐ Expressing Reservations

- For applications wishing to reserve network resources, need two items:
  - traffic descriptor (TD): QoS parameters
  - signaling protocol: a way to indicate the TD to the network/service
- Traffic descriptor used in 3 ways:
  - traffic contract (used for \$ and legal)
  - input to regulator
  - input to policer

## 7 ☐ Traffic Regulation and Policing

- Policer and regulator basically identical except for

their location, and that a regulator usually only delays traffic (rather than dropping it)

## 8 ☐ Common Traffic Descriptors

- Peak Rate: maximum source rate
  - for fixed size pkts: inverse of inter-pkt time (time between start of each packet)
  - for variable size: max rate over some period
  - easy to compute and police, but very sensitive to outliers, so useful by itself only for networks with smooth traffic
- Average rate: average over some time
  - $[t, a]$ :  $a$  bits may be sent over a window of time  $t$

## 9 ☐ Average Rate Descriptor Window

- For  $[t, a]$  average rate descriptor, how should the window work?
- Jumping window
  - source will not inject more than  $a$  bits in hops of time  $t$
  - sensitive to start time
- Moving window
  - source will not inject more than  $a$  bits in any window of time  $t$  (not sensitive to start time)

## 10 ☐ Linear Bounded Arrival Processes (LBAPs)

- An LBAP-constrained source bounds the number of bits it transmits in any interval of length  $t$  by a linear function of  $t$ . The number of bits transmitted in any interval of length
- is roughly the long-term average rate allocated by the net to the source and the longest burst a source may send, given the choice of

## 11 Leaky Bucket Regulators

- Regulates an LBAP descriptor

## 12 Leaky Bucket Operation

- Leaky bucket accumulates fixed-size *tokens* in a *token bucket*
- Transmits a packet (from buffer, if any are there) or arriving packet only if the sum of the token sizes in the bucket add up to the packet size
- More tokens are periodically added to the bucket (at rate  $\rho$ ). If tokens are to be added when the bucket is full, they are discarded.

## 13 Properties of Leaky Bucket

- Can think of this as a single-server queuing system with constant service rate
- Can act as peak rate or moving-window average rate regulator
  - peak-rate reg if  $\rho = \text{peak rate}$ ,  $\tau = 1$
  - ave-rate reg if  $\rho = \text{avg rate}$ ,  $\tau = 1$
- Can act as policer (no packet buffer)

## 14 Properties of Leaky Bucket

- Does not bound the peak rate of small bursts (sometimes augmented for this)
  - because bucket may contain enough tokens to cover a complete burst size
- Performance [loss rate] depends only on the sum of the packet buffer size and the token bucket size (one trades off the other exactly)

## 15 Example

- 2 tokens of size 100 bytes added to bucket of capacity 500 each second
  - avg rate=200 bytes/s, largest burst size is 500 bytes, peak rate is unbounded (500 bytes can be transmitted arbitrarily fast)
  - can never send packets bigger than 500 bytes
  - if a 400 byte packet arrives when bucket contains 200 bytes of tokens, will have to wait between just over 0.5 and 1 sec to send

## 16 ☐ Constructing End-to-End QoS

- In combination with packet scheduling, regulation/policing can be used to achieve end-to-end Quality of Service...
- How to do the scheduling?

## 17 ☐ What Can be Scheduled?

- Given an output link scheduler, can allocate:
  - mean delay (using service order)
  - bandwidths (using service ratios)
  - loss rates (adjusting buffering)
- For individual application requirements, need a per-flow reservation/allocation capability

## 18 ☐ Scheduling Disciplines

- Really two parts:
  - service ordering (who goes when)
  - loss rate allocation (who to delete/drop)
- For networking, want to allocate:
  - link bandwidth
  - buffer space
- Most published results are studies of output queues

to links operating at the network layer

## 19 ☐ Requirements of a Scheduler

- Ease of implementation
  - both best-effort and guaranteed traffic
- Fairness and protection (for best-effort)
- Performance bounds (for guaranteed)
- Ease and efficiency of corresponding admission control procedure

## 20 ☐ Ease of Implementation

- Scheduling choices are made per-packet
  - map packet to flow/connection
  - want  $O(1)$  not  $O(N)$  scaling [ $N$  connections]
- Execution time considerations
  - algorithm itself may run fast
  - issue is amount of scheduling state (variables, records, pointers, etc) and how long it takes to access these...
  - memory bandwidth/latency issues

## 21 ☐ Protection and Fairness

- Best effort flows may affect others:
  - especially when congestion-controlled flows (eg TCP) compete against non-congestion-controlled flows (eg UDP)
  - protection provides local isolation
- Fairness
  - local notion of fair meaning equal
  - commonly, the max-min fairness criteria

## 22 ☐ Max-Min Fairness Criteria

- How to share equally with different resource demands

- small users will get all they want
- large users will evenly split the rest
- More formally, perform this procedure:
  - resource allocated to customers in order of increasing demand
  - no customer receives more than requested
  - customers with unsatisfied demands split the remaining resource

### 23 Water-Filling Analogy

### 24 Max-Min Weighted Fair Share

- Can extend basic max-min fairness with a weight vector
- Now can prefer some customers to others:
  - resource allocated to customers in order of increasing demand, normalized by weight
  - no customer receives more than requested
  - customers with unsatisfied demands split the remaining resource in proportion to their weights

### 25 Max-Min Example

- demands: 2, 2.6, 4, 5; capacity: 10
  - $10/4=2.5$  looks good, but 1st customer needs only 2, so excess of 0.5, distribute among 3, so  $0.5/3=0.167$
  - now we have allocs of [2, 2.67, 2.67, 2.67], leaving an excess of 0.07 for cust #2
  - divide that in two, gets [2, 2.6, 2.7, 2.7]
- Maximizes the minimum share to each customer whose demand is not fully satisfied

### 26 Weighted Max-Min Example

- demands: 4, 2, 10, 4; capacity(C): 16; weight vector

- first normalize weights so that minimum weight is 1:
- use sum of normalized weights  $n$  as # custs
- $C/n=16/16=1$  -> alloc
- 7 extra to go to custs #3 and #4 (by weight)
- $7/3=2.3$ , so add (0,0,2.33,4.66)
- with (4,2,3.33,6.66), give #4's excess to #3
- final allocation is (4, 2, 6, 4)

## 27 ☐ Performance Bounds

- For guaranteed service, scheduler should support end-to-end performance bounds as requested by applications and possibly controlled by network admin
- Types of performance bounds
  - deterministic: always holds
  - statistical: holds with some probability over some space.
 Ways of expressing this:
  - numbers (like 98% of green pkts delivered)
  - one-in-N (at most 1 in 100 pkts dropped)

## 28 ☐ Common Performance Bounds

- Bandwidth
  - some minimum bandwidth over some time
  - some systems provide only these types
- Delay
  - bound on some parameter of the delay distribution
    - worst-case delay
    - (measured/sampled) mean delay
    - 99-percentile of delay

## 29 ☐ Common Performance Bounds-2

- Delay-jitter
  - difference between largest and smallest delay seen by

packets on a connection

- End-to-End delay distribution:

### 30 ☐ Common Performance Bounds-3

- Loss bound
  - requires the fraction of packets lost over a connection to be less than some bound
  - can achieve zero loss bound with good admission control

### 31 ☐ Choices in Scheduler Design

- Number of priority levels/classes
- Work conserving or non-work-conserving
- Degree of aggregation within a level
- Service order within a level

### 32 ☐ Priority Scheduling

- each connection gets a priority level
  - n priority levels total
  - service highest (to exhaustion) first, then on to next, etc [simple priority]
  - in worst case, leads to starvation [hence need for admission control]
  - in practice, only a few levels used (say, 3; one for network control, 1 for high priority and 1 for lower priority)

### 33 ☐ Work Conserving or Not

- Work-conserving scheduling is idle only when there is no traffic to send
- Non-work-conserving may be idle at any time, in an effort to smooth out the traffic pattern
  - limits buffer requirements at receiver
  - contributes to longer overall delay

### 34 ☐ Non-Work-Conserving Approach

- Other advantages
  - sum of per-hop bounds can tightly bound the end-to-end delay and delay-jitter
  - if performed at each hop, works on heterogeneous networks
- Idea is to compute packet eligibility time
  - time at which a packet is eligible to be sent out the link to maintain desired traffic characteristics

### 35 ☐ Eligibility Time Computations

- Rate-jitter regulation
  - scheduler guarantees traffic departing switch obeys a rate descriptor [peak rate regulator]
  
- $E(k)$ =eligibility time of  $k$ th packet,  $A(k)$ =arrival time of  $k$ th packet at scheduler,  $X_{min}$ =inverse of peak rate

### 36 ☐ Eligibility Time Computations

- Delay-jitter regulation
  - scheduler guarantees sum of queuing delay in previous switch and regulation delay in current switch are constant. Removes the effect of variability in queuing delay in previous switch
  
- $E(i,k)$ =eligibility time of  $k$ th packet at switch  $i$ ,  $D$ =delay bound at previous switch,  $L$  is largest possible delay on  $[i,i+1]$  link

### 37 ☐ Perspective

- Delay-jitter harder to implement
  - network operator must know bound on prop delay of each link
  - must maintain synchronized clocks at switches
  - provides perfect traffic reconstruction, but may be only of academic interest
- The real-world
  - none of the non-work-conserving disciplines are in use, or are currently really proposed

### 38 ☐ Aggregation

- Degree of “grouping” used for scheduling
  - max aggregation: single state for all
  - min aggregation: state for each connection
  - in between: classes [will be important later]
- Benefits
  - less scheduler state (good for implementation and for advertising info in routing protocols)
  - evenly distributes the jitter induced by bursts due to other members of the class

### 39 ☐ Service Order

- Straightforward: either in-order or not in order. If not in-order, must be tagged and essentially sorted according to some ordering.
- With FCFS, cannot have higher-priority jump ahead of low priority traffic
- FCFS also does not achieve max-min fairness (rewards bandwidth hogs)