EECS 122, Lecture 27

Today’s Topics:

Scheduling Best-Effort and
Guaranteed Connections

QoS in ATM

Kevin Fall, kfall@cs.berkeley.edu

Where we are…

- The motivation for QoS
  - a desire to provide a better than best effort service
  - motivation by ISPs to charge for new services
- Need scheduling and traffic regulation
  - guaranteed and best-effort handled somewhat differently
- Haven’t talked about scheduling details...

Scheduling of Best-Effort Traffic

- Ideal work-conserving scheduling discipline that achieves max-min fairness is called Generalized Processor Sharing (GPS)
- GPS serves an infinitesimally small amount of data from each queue in round-robin fashion
- GPS is not implementable, but achieves exact weighted max-min fairness, so usually compare real approaches to GPS

Max-Min Fairness using GPS

- A GPS server, *m*, that serves *N* sessions on a link is characterized by *N* positive real numbers $\phi_1^m, \phi_2^m, \ldots, \phi_N^m$. These numbers denote the relative amount of service to each session. More precisely, if $S_i^m(\tau, t)$ is the amount of session *i* traffic served by server *m* during an interval [**t** ,*t*], then

$$\frac{S_i^m(\tau, t)}{S_j^m(\tau, t)} \geq \frac{\phi_i^m}{\phi_j^m} r^m; \; j = 1,2,\ldots, N$$

- (for session *i* continually backlogged)

What this means…

- So, for non-backlogged connections, they already receive what they ask for. The GPS server ensures that backlogged connections share the remaining bandwidth in proportion to the assigned weights (i.e. max-min fairness).
- For every backlogged connection *i*, each receives a service rate at switch *m* of:

$$g_i^m = \frac{\phi_i^m}{\sum_{j=1}^{N} \phi_j^m} r^m$$

Weighted Round Robin

- WRR approximates GPS reasonably well for connections with equal packet sizes
  - if different size packets, need mean packet size to be close to GPS [may be hard to know]; if not known, RR isn’t fair
  - fair only over time scales longer than a round time; with large # connections or small weight, may be unfair over long periods
- Note WRR would be ok in ATM…

Deficit Round Robin (DRR)

- Modification to WRR so knowing mean packet size is not required
- Choose a <u>quantum</u> of bits to serve from each connection in order. For each HOL packet, if its size is <= (quantum+credit) send and save excess, otherwise save entire quantum. If no packet to send, reset counter (to remain fair)
- Easier implementation than WFQ

DRR Example

- Each connection has a deficit counter (to store credits) with initial value zero. Scheduler is config'd with quantum.
  - Assume quantum=1000, 3 connections (A-C) have packets of size 1500, 800, 1200
  - Round 1: A's counter goes to 1000, B's first packet is served (and its counter goes to 200), C's counter goes to 1000
  - Round 2: A's packet is served (and its counter goes to 500), C's packet is served (and its counter goes to 800), B's counter reset to zero

PGPS (Pkt-by-pkt GPS) and WFQ

- WFQ and PGPS independently discovered disciplines which do not require GPS's infinitesimal service assumption
- we have studied this already, and know it approximates max-min fairness
- If we combine with regulation/policing, we can do some interesting things...

Guaranteed Service Connections

- It turns out you can provide bandwidth and delay bounds using WFQ (wow!)
- <u>Assuming a leaky-bucket constrained source</u> *i* , assume its service rate is:

$$g_i^m = \frac{\phi_i^m}{\sum_{j=1}^{N} \phi_j^m} r^m ; g(i) = \min_m \{g_i^m\} \geq \quad (i)$$

- Result (end2end trans+queue delay D):

$$D^*(i) \leq \frac{(i)}{g(i)} + \sum_{m=1}^{M-1} \frac{P_{\max}(i)}{g_i^m} + \sum_{m=1}^{M} \frac{P_{\max}}{r^m}$$

Looking a bit closer...

- Result is from Parekh/Gallager, 1992

$$D^*(i) \leq \frac{(i)}{g(i)} + \sum_{m=1}^{M-1} \frac{P_{\max}(i)}{g_i^m} + \sum_{m=1}^{M} \frac{P_{\max}}{r^m}$$

- Definitions:
  - Pmax(i)=largest pkt on connection
  - Pmax=largest pkt allowed on network
  - D*(i)=end-2-end delay on connection i
- With Pmax=0[infinitesimal], gives bound of just $s(i)/g(i)$ [like one scheduler w/rate g(i)]

An Example

- Assume connection has leaky bucket parameters (16KB, 150Kbps), 10 hops, all link bandwidths 45Mb/s. With largest pkt size of 8KB, what g will guarantee an end-to-end delay of 100ms, assuming a total propagation delay of 30ms?
  - Need max queue delay of 100-30=70ms, so we have

Example (cont'd)

$$D^*(i) \le \frac{(i)}{g(i)} + \sum_{m=1}^{M-1} \frac{P_{\max}(i)}{g_i^m} + \sum_{m=1}^{M} \frac{P_{\max}}{r^m}$$

- Solve this for g:
  - 0.7=(16K*8)/g+(10-1)*8K*8/g+10*8K*8/45*10^6)
  - g is about 13Mb/s [>85x source avg rage!]
  - large packets can lead to substantial delays

Virtual Clock

- Similar to WFQ (scheduler stamps packets with finish time tags and services them in order of tags)
- Instead of GPS, emulates TDM; easier to compute finish number than WFQ
- If all connections are backlogged, WFQ and VC behave identically (ok for guar.)
- Read about it in Chapter 8...

Delay-Earliest Due Date (EDD)

- Delay-EDD
  - assign scheduling deadlines so that even with all connections at peak rate, worst-case delay in traffic descriptor is met
  - <u>bandwidth reservation independent of delay bound</u>, but must use peak rate regulator thereby giving up stat muxing gain
  - deadline is time at which is should be sent had it been received according to traffic contract (slower than peak rate)

Jitter-EDD

- Delay-jitter regulator precedes the EDD scheduler
  - packets receive same delay at each hop (except the last one), so total jitter is reduced to that of last hop
  - can provide end-to-end bw, delay, & jitter bounds
- Incorporates Delay-EDD for delay guarantees, so in that case must reserve at peak rate

Rate Controlled Scheduling

- Provide bw, delay, and jitter bounds
- Two components: regulator & scheduler
  - regulator determines eligibility time for pkts
  - scheduler selects among eligible packets
- By selecting which regulator and scheduler, can implement a wide range of overall service disciplines
  - e.g.: rate-controlled static priority
    - rate-jitter regulator & multi-level FCFS prio scheduling

RC Scheduling Implementation Issues

- Implement a regulator
  - if we can tolerate some granularity in time stamps, can use a calendar queue
  - for delay-jitter regulation, also need clock synchronization and timestamps in each pkt
- Implement a scheduler
  - if FCFS or multi-FCFS, just queues
  - if sorted/deadlines, etc need priority queue

Packet Drop Strategies

- Assumption is that guaranteed connections rarely drop any packets (due to admission control), but best-effort flows must deal with this
- Similar characterizations as schedulers:
  - degree of aggregation
  - choice of drop priorities
  - early or overloaded drop
  - drop position

Degree of Aggregation

- Essentially a choice of per-flow or per-class state maintenance
  - per-flow: more protection on overload
  - per-class: less protection, easier to implement
- Can achieve min-max buffer allocation if always drop from the largest queue
- If using WFQ-like scheduler, drop packet with largest finish number, even w/out per-flow queuing

Drop Priorities

- Mark some packets as higher priority
  - on overload, drop lower priority
  - (maybe even do this before overload)
- Loss bits may be set by source or policer (or both)
- What to drop (note cell versus frame in ATM)… several switches uses PPD/EPD [basically doing frame dropping]

Drop Early or on Overload

- Early drop works for responsive sources
  - early random drop and RED
  - ERD not as effective as RED in controlling misbehaving users
- RED substantially improves performance of network of cooperating TCP sources
  - probability of drop is roughly proportional to its throughput share
  - RED has no bias against bursty sources

Drop Position

- Which packet to drop when dropping?
  - head, tail, random, [entire queue]
- Tail drop
  - most straightforward to implement
  - no modification to queue head/tail pointers
- Head drop
  - better for dupack detection (because ‘hole” will be served earlier; don't have wait for whole queue)

QoS Summary

- Guaranteed and best-effort service
  - guaranteed is set up by traffic descriptor, should rarely or never drop packets
  - best-effort may drop packets, but scheduling should be fair
- Leaky-bucket traffic model
- Many scheduling disciplines
  - GPS, WFQ, Delay/Jitter-EDD, etc
- What happens in the real world?

The Real World

- Today, there are really two contenders for supporting QoS: Internet and ATM
- ATM QoS has a several-year jump start, but is considerably more complex (and applies only to ATM, of course)
- Internet QoS is a simpler model, but must apply to all sorts of link technologies, and so poses a significant challenge
- Expect to see Internet QoS on ATM...

QoS in ATM: Service Categories

- Constant Bit Rate (CBR)
  - periodic, constant bit rate, like TDM
- Variable Bit Rate (VBR-rt & VBR-nrt)
  - variable periodic sources (real-time and not)
- Available Bit Rate (ABR)
  - like best-effort, but with flow control
- Unspecified Bit Rate (UBR)
  - completely best-effort

ATM Traffic Parameters

- Peak Cell Rate (PCR):
  - maximum cell transport rate
- Sustainable Cell Rate (SCR)
  - average allowable, long-term transfer rate
- Maximum Burst Size (MBS)
  - maximum back-to-back cell burst size
- Minimum Cell Rate (MCR)
  - minimum cell transport rate

ATM QoS Parameters

- Peak-to-peak Cell Delay Variation (CDV)
  - essentially a jitter measurement
  - bound on this is the CDV "tolerance" (CDVT)
- Maximum Cell Transfer Delay (maxCTD)
  - end-to-end delay bound
- Cell Loss Ratio (CLR)
  - bound on fraction of lost cells

Use of Parameters by Category

- CBR: uses PCR as maximum rate and maxCTD as bound on delay of cells
- VBR-rt: uses PCR, SCR, and MBS; cells delayed above maxCTD not valuable
- VCBR-nrt: uses PCR, SCR, and MBS
- ABR: uses PCR and MCR
  - uses feedback (RM cells) for flow control
- UBR: PCR for information only

QoS Components in ATM

- Usage Parameter Control (UPC)
  - regulator used to set CLP bit on overload
- Traffic Shaping
  - leaky bucket control via GCRA algorithm
- Cell Loss Priority Control (CLP)
  - drop policy based on CLP bits
- Connection Admission Control (CAC)
  - per-VC admission control at call setup
- ABR Flow Control via RM cells

Shaping and UPC

- Traffic shaping and UPC are procedures for regulating and policing traffic at the ATM ingress
- Reference algorithm is called GCRA
  - continuous leaky bucket algorithm
  - defines relationship between PCR and CDVT as well as SCR and BT (burst tolerance, derived from PCR, SCR and MBS)
  - notation GCRA(I,L); I=increment, L=limit

Cell Loss Priority

- CLP bit in ATM header may be set by either end station or by shaping/UPC mechanism
- For ATM networks which are sensitive to it, can be used to direct load shedding on network during congested periods

Connection Admission Control

- No algorithmic in the ATM spec, so this is an area for some innovation by switch manufacturers
- Two main approaches:
  - measurement based
  - analytic, based on assumed statistics and traffic parameters

Admission Control

- Many algorithms in the literature
- One important class uses *equivalent capacity*
- Equivalent capacity is the amount of capacity (bandwidth) required to handle a set of statistically multiplexed flows with a bound on the probability of buffer overrun (oversubscription)

Source Models

- Equivalent capacity (and other models) try to estimate statistics of sources by making certain assumptions about their behavior
- One common assumption is that traffic is independent (e.g. independent on/off periods)
- These independence assumptions help to form tractable equations

Equivalent Capacity

- We want to find $\hat{c}_{(s)}$ such that $\Pr(B > \hat{C}_{(S)}) \le$
- So, for assumed two-state sources with exponentially distributed burst and idle periods [memoryless], we can attempt to compute how many flows are likely to be bursting at the same time, but even this is hard, so…
- Assume a Gaussian distribution on the aggregate bit rate (params m, ***s***)

Equivalent Capacity

- With all of these assumptions, we get:

$$\hat{C}_{(S)} \approx m + \ ' \ \ ; \quad ' = \sqrt{-2\ln(\ \ ) - \ln(2\ \ )}$$

$$m = \sum_{i=1}^{N} m_i , \quad {}^{2} = \sum_{i=1}^{N} {}_{i}^{2}$$

- Upshot: admission control not such an easy problem. Furthermore, there is reason to believe these sorts of assumptions may not be valid...