Internet Technology

13. Network Quality of Service

Paul Krzyzanowski

Rutgers University

Spring 2016

Internet gives us "best effort"

- The Internet was designed to provide best effort delivery
  - No guarantees on when or if packet will get delivered
- Software tries to make up for this
  - Buffering, sequence numbers, retransmission, timestamps
- Can we enhance the network to support multimedia needs?
  - Control quality of service (QoS) with resource allocation & prioritization on the network

What factors make up QoS?

- Bandwidth (bit rate)
  - Average number of bits per second through the network
- Delay (latency)
  - Average time for data to get from one endpoint to its destination
- Jitter
  - Variation in end-to-end delay
- Loss (packet errors and dropped packets)
  - Percentage of packets that don’t reach their destination

Service Models for QoS

- No QoS (best effort)
  - Default behavior for IP with no QoS
  - No preferential treatment
  - Host is not involved in specifying service quality needs

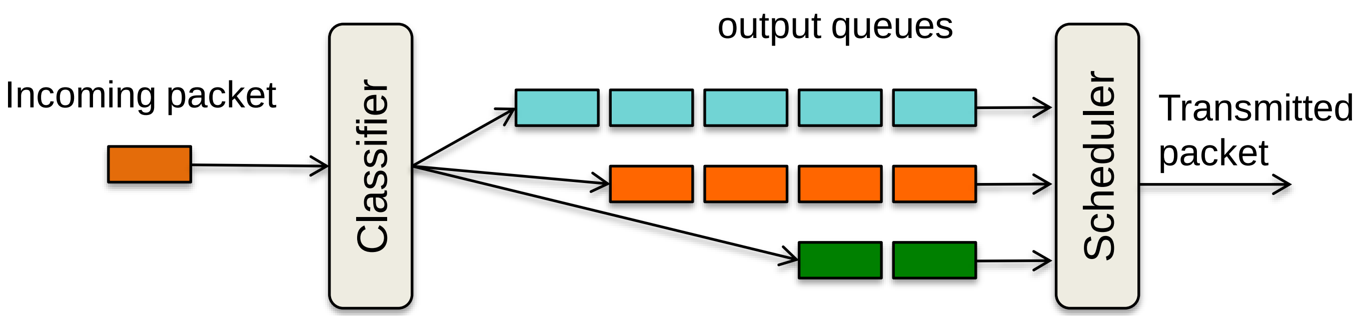
- Soft QoS (Differentiated Services)
  - No explicit setup
  - Identify one type of service (data flow) vs. another
  - Certain classes get preferential treatment over others

- Hard QoS (Integrated Services)
  - Network makes commitment to deliver the required quality of service
  - Host makes an end-to-end reservation
    - Traffic flows are reserved

Link scheduling at a router

Packets usually get lost or delayed at link output queues on a router

– Link scheduling discipline:
  Defines how packets are scheduled at the output queue



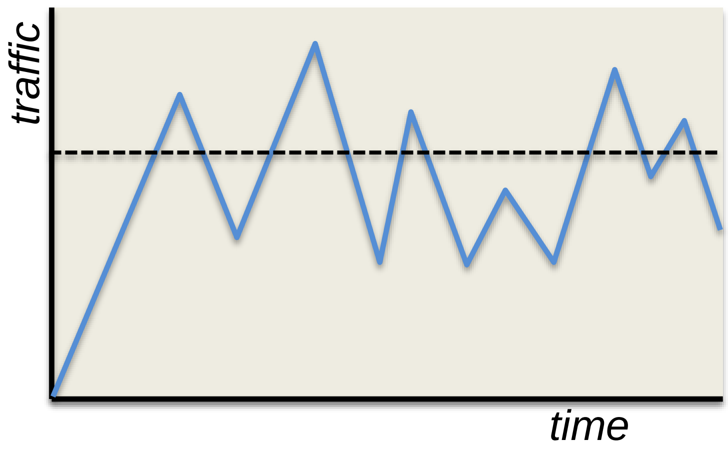
Per-link output queues on a router

Link scheduling disciplines

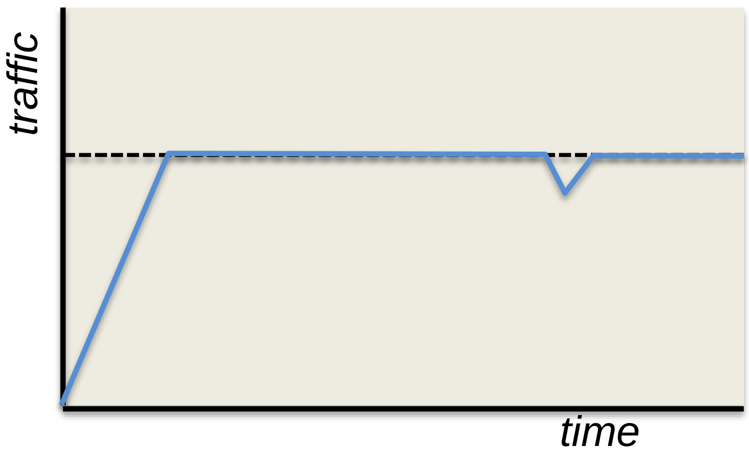
- First-In-First-Out (FIFO)
  - Simplest but no differentiation on service class
- Priority queuing
  - Classify packets based on source/dest address, source/dest port, source link, DS bits, protocol, etc.
  - Each class gets its own queue
  - Transmit packets from the highest class with a non-empty queue
  - Risk of starvation
    - We want traffic isolation: ensure that one class of service cannot adversely affect another class (e.g., consume all bandwidth)
- Round robin
  - Queue per class; each class gets an equal share – not what we want
- Weighted Fair Queuing (WFQ)
  - Each queue gets a priority and a minimum % of link speed

Bandwidth Management

- Traffic Shaping
  – Goal: regulate average rate of data transmission per flow
  – Queue packets during surges and release later: delay traffic
  – Example: high-bandwidth link to low-bandwidth link



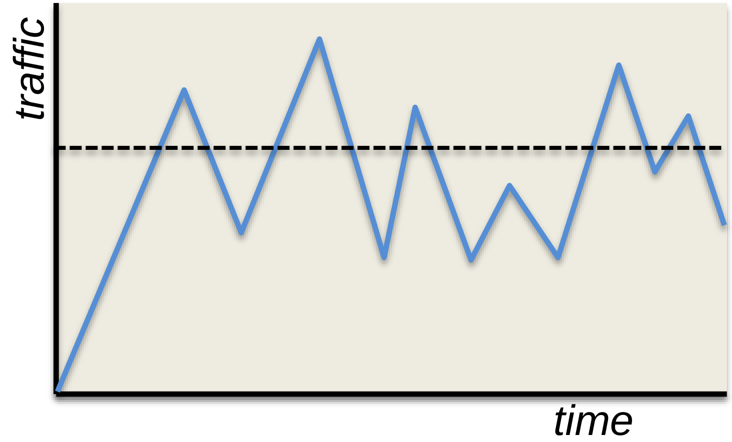
Before shaping



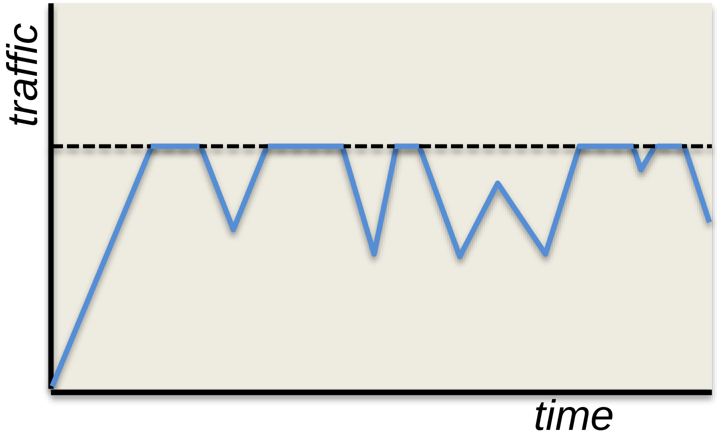
After shaping

Bandwidth Management

- Traffic Policing
  - Goal: Monitor network traffic and discard offenders
  - Discard traffic that exceeds allotted bandwidth



Before shaping



After policing

Traffic Shaping: **Leaky** Bucket

<u>Visualization</u>

- Bucket with a hole
- Filled up at a varying rate
- Water leaks at a constant rate

<u>Implementation</u>

- Add incoming packets to the end of a queue (buffer)
- Transmit packets from the start of the queue at a constant rate

- Bucket = packet queue buffer
- If a packet comes in and bucket is full, discard packet
  - Buffer overrun
- If there is nothing to transmit (bucket is empty)
  - Buffer underrun
- Convert an uneven flow of packets into an even flow
  - Removes jitter

Traffic Shaping/Policing: **Token** Bucket

- Bucket holds *tokens* that are generated at a certain rate
- You need a token to transmit a packet
  - The bucket must hold and destroy a token(s)
- The token bucket allows a host to *save up permission* to send large bursts later
  - Bucket size determines maximum burstiness

Traffic Shaping/Policing: Token Bucket

Desired average rate: *r* bytes/second

Add a token every *1/r* seconds: assume a token = 1 byte

    If # tokens > b (bucket is full), discard the token

When packet arrives (size = *n* bytes):

    if # tokens is < *n*

        Traffic shaping: queue (delay) the packet until there are enough tokens

        Traffic policing: drop the packet

    else

        transmit the packet and remove *n* tokens

In an implementation, the "tokens" are just one number, not a collection

Token bucket vs. Leaky bucket

- Token bucket: may be bursty
  - Tokens are accumulated when there isn't much data and can be used whenever data arrives
  - Goal: enforce an <u>average rate of traffic</u>

- Leaky bucket: cannot be bursty
  - The bucket is always drained at a fixed rate
  - Goal: enforce a <u>peak rate of traffic</u>

Router support for QoS

- Most routers support two QoS architectures

  - Differentiated Services (DiffServ)
    - Class of a packet is marked in the packet

  - Integrated Services (IntServ)
    - Signaling protocol tells routers that a specific flows needs special treatment
    - IntServ uses the Resource Reservation Protocol (RSVP)

Differentiated Services (soft QoS)

- Treat some traffic as better than other
  - Statistical - no guarantees
- Identify class of service
  - Router can use this data to make scheduling/dropping decisions
- Use on Internet (especially across ISPs) limited due to peering agreement complexities
  - DiffServ only makes sense if *all* routers participate in the same manner

Differentiated Services (DiffServ)

- DSCP field in IPv4 header (top 6 bits of 2nd byte)
  - Differentiated Services Codepoint (DSCP)
  - DS field in an IPv6 header
  - Filled in at the edge (by the host)

- RFC 2597 recommends *codepoints*
  - Four classes of service
  - Grouped into three precedence (priority) levels (low, med, high)

	Class 1	Class 2	Class 3	Class 4
Low	001010	010010	011010	100010
Medium	001100	010100	011100	100100
High	001110	010110	011110	100110

See RFC 3260

Integrated Services: RSVP (Hard QoS)

- IntServ: Integrated Services (RFC 1633)
  - End-to-end reservation of services
- Uses RSVP: ReSerVation Protocol (RFC 2205)
  - Resource reservation & delivery protocol
  - Each unidirectional data stream is a flow
- Every device through which data flows must support RSVP
  - Admission control: determines if a node has sufficient resources for the QoS request
  - Policy control: determines if the user has the permission to make the reservation
  - RSVP is a soft state protocol: reservations expire unless refreshed
    - Typically every 30 seconds

Integrated Services: RSVP

- Sender sends a PATH message requesting bandwidth
  - Traffic specification (TSPEC)
    - Define token bucket: rate & bucket depth, peak rate, min/max packet sizes
  - Establishes a stored route (path) – routers keep state!

- Receiver asks for a reservation
  - Receiver then sends a RESV message to reserve the resources along that path
  - Request specification (RSPEC)
    - Specify levels of assurance
      - Best effort (no reservation)
      - Controlled Load: soft QoS – data rates may increase or packet loss may occur
      - Guaranteed: hard QoS – tight bounds on delay
  - Router (or host) at each hop decides whether to accept the request

RTP & RTCP

Real-time Transport Protocol (RTP)

- Application-level protocol on top of UDP
  - RTP does not define any mechanisms for data delivery or QoS control
  - Delivery is not guaranteed and in-order delivery is not guaranteed

- RTP header:
  - payload type: identifies type of video or audio encoding
    - App can change encoding type mid-stream (e.g., lower bandwidth)
  - sequence number: app can detect missing packets & conceal data loss
  - timestamp: app can play back data at appropriate intervals
  - source ID of stream: uniquely identifies stream; allows demultiplexing

- RTP is widely used for voice and video, particularly for media transport in SIP (Session Initiation Protocol) systems

RTP Control Protocol (RTCP)

- Companion protocol to RTP
- Provides feedback about an RTP flow
  - Out-of-band protocol
- RTP sent on even port X; RTCP on port X+1
- Reports
  - Identifies source name (DNS CNAME)
  - Receiver report: tells sender about received quality of service
    - Lost packet counts, jitter, round-trip delay time
  - Sender report:
    - Absolute timestamp
    - Total packet count in session; total byte count
    - Summary of receiver reports: fraction of packets lost, total lost, jitter estimate

The end