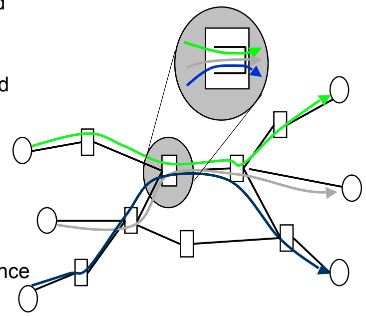
15-441 Computer Networking

Queue Management and
Quality of Service (QOS)

Traffic and Resource Management

- Resources statistically shared
  $\sum \text{Demand}_i(t) > \text{Resource}(t)$
- Overload causes congestion
  - packet delayed or dropped
  - application performance suffer
- Local vs. network wide
- Transient vs. persistent
- Challenge
  - high resource utilization
  - high application performance



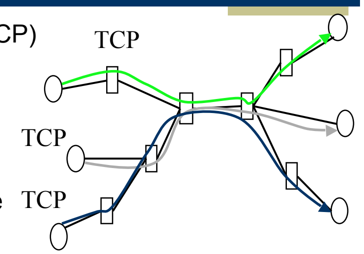
Resource Management Approaches

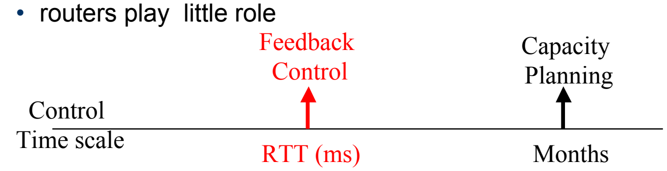
$$\sum \text{Demand}_i(t) > \text{Resource}(t)$$

- Increase resources
  - install new links, faster routers
  - capacity planning, provisioning, traffic engineering
  - happen at longer timescale
- Reduce or delay demand
  - Reactive approach: encourage everyone to reduce or delay demand
  - Reservation approach: some requests will be rejected by the network

Congestion Control in Today's Internet

- End-system-only solution (TCP)
  - dynamically estimates network state
  - packet loss signals congestion
  - reduces transmission rate in presence of congestion
  - routers play little role



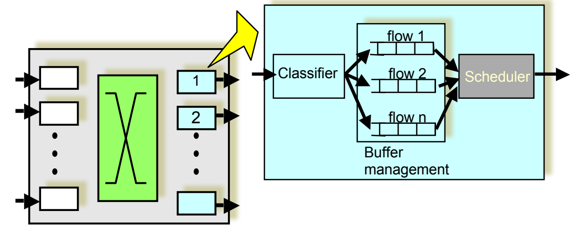


More Ideas on Traffic Management

- Improve TCP
  - Stay with end-point only architecture
- Enhance routers to help TCP
  - Random Early Discard
- Enhance routers to control traffic
  - Rate limiting
  - Fair Queueing
- Provide QoS by limiting congestion

Router Mechanisms

- Buffer management: when and which packet to drop?
- Scheduling: which packet to transmit next?



Overview

- Queue management & RED
- Why QOS?
- QOS Principles
- Introduction to Scheduling Policies
- Integrated Services

Queuing Disciplines

- Each router must implement some queuing discipline
- Queuing allocates both bandwidth and buffer space:
  - Bandwidth: which packet to serve (transmit) next
  - Buffer space: which packet to drop next (when required)
- Queuing also affects latency

Typical Internet Queuing

- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
- FIFO (first-in-first-out)
  - Implies single class of traffic
- Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: drop policy

FIFO + Drop-tail Problems

- Leaves responsibility of congestion control completely to the edges (e.g., TCP)
- Does not separate between different flows
- No policing: send more packets → get more service
- Synchronization: end hosts react to same events

FIFO + Drop-tail Problems

- Full queues
  - Routers are forced to have have large queues to maintain high utilizations
  - TCP detects congestion from loss
    - Forces network to have long standing queues in steady-state
- Lock-out problem
  - Drop-tail routers treat bursty traffic poorly
  - Traffic gets synchronized easily → allows a few flows to monopolize the queue space

Active Queue Management

- Design active router queue management to aid congestion control
- Why?
  - Router has unified view of queuing behavior
  - Routers see actual queue occupancy (distinguish queue delay and propagation delay)
  - Routers can decide on transient congestion, based on workload

Design Objectives

- Keep throughput high and delay low
  - High power (throughput/delay)
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes

Lock-out Problem

- Random drop
  - Packet arriving when queue is full causes some random packet to be dropped
- Drop front
  - On full queue, drop packet at head of queue
- Random drop and drop front solve the lock-out problem but not the full-queues problem

Full Queues Problem

- Drop packets before queue becomes full (early drop)
- Intuition: notify senders of incipient congestion
  - Example: early random drop (ERD):
    - If qlen > drop level, drop each new packet with fixed probability *p*
    - Does not control misbehaving users

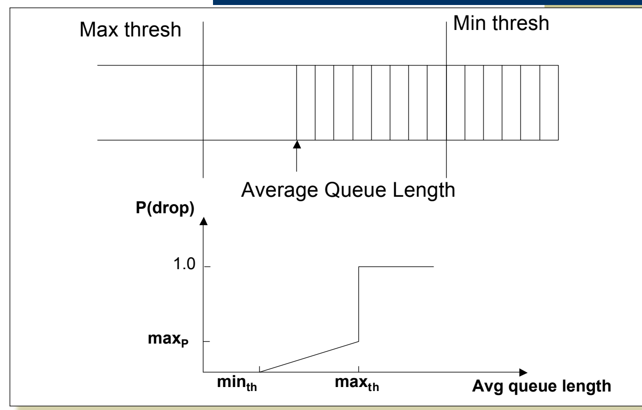
Random Early Detection (RED)

- Detect incipient congestion
- Assume hosts respond to lost packets
- Avoid window synchronization
  - Randomly mark packets
- Avoid bias against bursty traffic

RED Algorithm

- Maintain running average of queue length
- If avg < min_{th} do nothing
  - Low queuing, send packets through
- If avg > max_{th}, drop packet
  - Protection from misbehaving sources
- Else mark packet in a manner proportional to queue length
  - Notify sources of incipient congestion

RED Operation



Explicit Congestion Notification (ECN) [Floyd and Ramakrishnan 98]

- Traditional mechanism
  - packet drop as implicit congestion signal to end systems
  - TCP will slow down
- Works well for bulk data transfer
- Does not work well for delay sensitive applications
  - audio, WEB, telnet
- Explicit Congestion Notification (ECN)
  - borrow ideas from DECBit
  - use two bits in IP header
    - ECN-Capable Transport (ECT) bit set by sender
    - Congestion Experienced (CE) bit set by router

Congestion Control Summary

- Architecture: end system detects congestion and slow down
- Starting point:
  - slow start/congestion avoidance
    - packet drop detected by retransmission timeout (RTO) as congestion signal
  - fast retransmission/fast recovery
    - packet drop detected by three duplicate acks
- Router support
  - RED: early signaling
  - ECN: explicit signaling

Overview

- Queue management & RED
- Why QOS?
- QOS Principles
- Introduction to Scheduling Policies
- Integrated Services

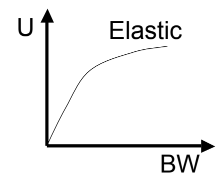
Motivation

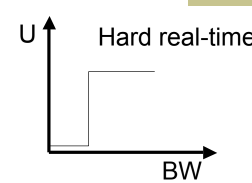
- Internet currently provides one single class of **"best-effort" service**
  - No assurances about delivery
- At internet design most applications are ***elastic***
  - Tolerate delays and losses
  - Can adapt to congestion
- Today, many "real-time" applications are ***inelastic***

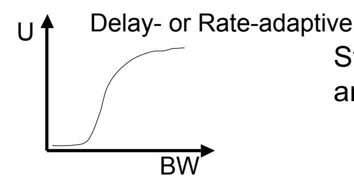
Why a New Service Model?

- What is the **basic objective** of network design?
  - Maximize total bandwidth? Minimize latency?
  - **Maximize user satisfaction** – the total **utility** given to users
- What does utility vs. bandwidth look like?
  - Shape depends on application
  - Must be non-decreasing function

Utility Curve Shapes

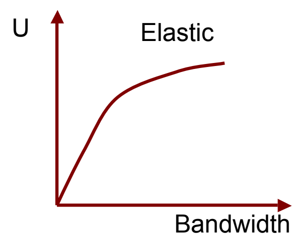






Stay to the right and you are fine for all curves

Utility curve – Elastic traffic

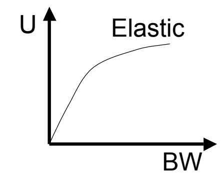


Does equal allocation of bandwidth maximize total utility?

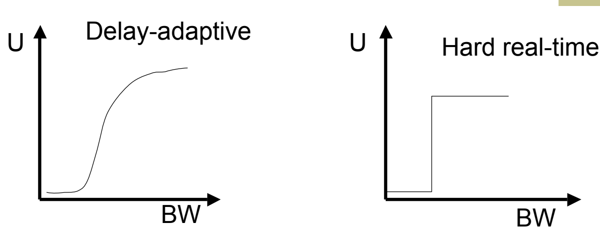
Admission Control

- If U(bandwidth) is concave → elastic applications
  - Incremental utility is decreasing with increasing bandwidth
  - Is always advantageous to have more flows with lower bandwidth
    - No need of admission control;

  This is why the Internet works!



Utility Curves – Inelastic traffic



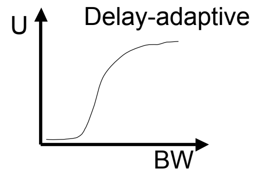
Does equal allocation of bandwidth maximize total utility?

Inelastic Applications

- Continuous media applications
  - **Lower and upper limit** on acceptable performance.
  - BW below which video and audio are not intelligible
  - Internet telephones, teleconferencing with high delay (200 - 300ms) impair human interaction
  - Sometimes called "tolerant real-time" since they can adapt to the performance of the network
- Hard real-time applications
  - Require **hard limits on performance**
  - E.g. control applications

Admission Control

- If U is convex → inelastic applications
  - U(number of flows) is no longer monotonically increasing
  - Need admission control to maximize total utility
- **Admission control** → deciding when adding more people would reduce overall utility
  - Basically avoids overload

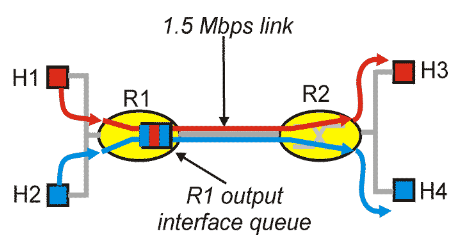


Overview

- Queue management & RED
- Why QOS?
- QOS Principles
- Introduction to Scheduling Policies
- Integrated Services

Improving QoS in IP Networks

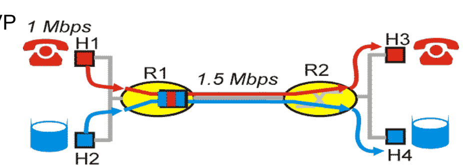
- IETF groups are working on proposals to provide better QoS control in IP networks, i.e., going beyond best effort to provide some assurance for QoS
- Work in Progress includes RSVP, Differentiated Services, and Integrated Services
- Simple model for sharing and congestion studies:



Principles for QoS Guarantees

- Consider a phone application at 1Mbps and an FTP application sharing a 1.5 Mbps link.
  - bursts of FTP can congest the router and cause audio packets to be dropped.
  - want to give priority to audio over FTP
- **PRINCIPLE 1: Marking of packets is needed for router to distinguish between different classes; and new router policy to treat packets accordingly**

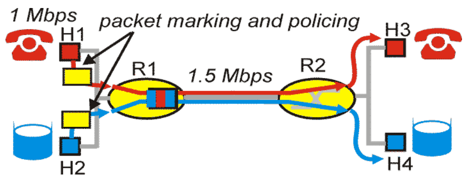
e.g. MPLS, Diffserv,RSVP



Principles for QoS Guarantees (more)

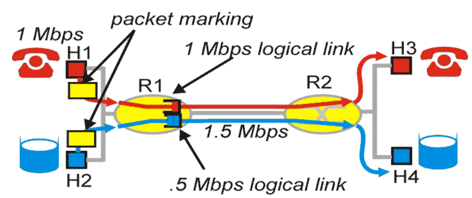
- Applications misbehave (audio sends packets at a rate higher than 1Mbps assumed above);
- **PRINCIPLE 2: provide protection (isolation) for one class from other classes**
- Require Policing Mechanisms to ensure sources adhere to bandwidth requirements; Marking and Policing need to be done at the edges:

e.g. WFQ



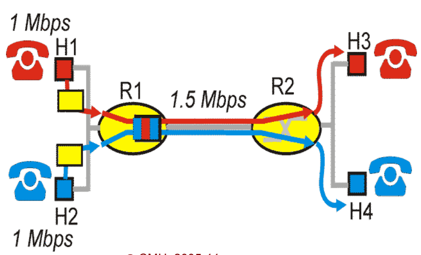
Principles for QoS Guarantees (more)

- Alternative to Marking and Policing: allocate a set portion of bandwidth to each application flow; can lead to inefficient use of bandwidth if one of the flows does not use its allocation
- **PRINCIPLE 3: While providing isolation, it is desirable to use resources as efficiently as possible**

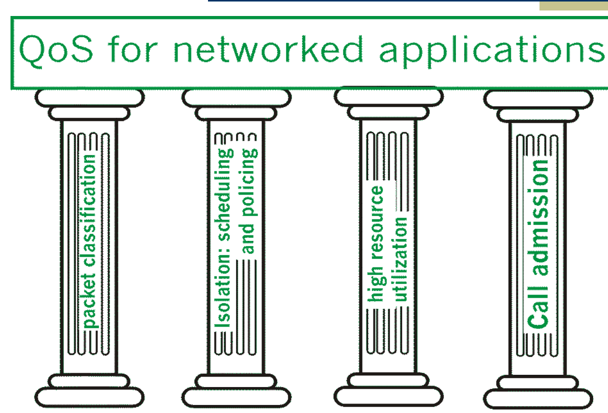


Principles for QoS Guarantees (more)

- Cannot support traffic beyond link capacity
- **PRINCIPLE 4: Need a Call Admission Process; application flow declares its needs, network may block call if it cannot satisfy the needs**



Summary

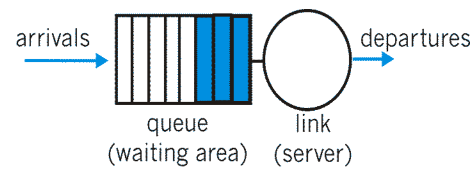


Overview

- Queue management & RED
- Why QOS?
- QOS Principles
- Introduction to Scheduling Policies
- Integrated Services

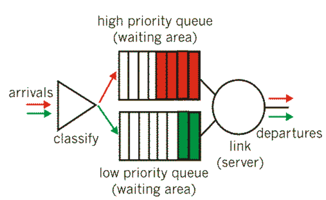
Scheduling And Policing Mechanisms

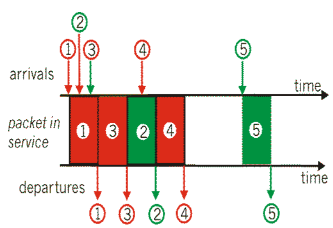
- Scheduling: choosing the next packet for transmission on a link can be done following a number of policies;
- FIFO: in order of arrival to the queue; packets that arrive to a full buffer are either discarded, or a discard policy is used to determine which packet to discard among the arrival and those already queued



Scheduling Policies

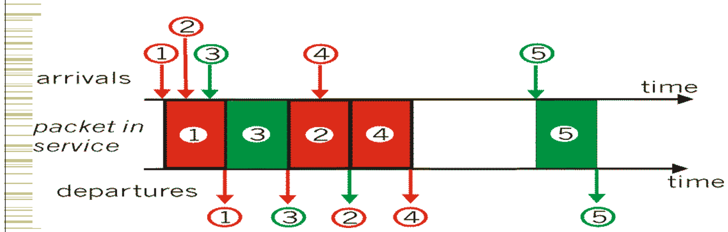
- Priority Queuing: classes have different priorities; class may depend on explicit marking or other header info, eg IP source or destination, TCP Port numbers, etc.
- Transmit a packet from the highest priority class with a non-empty queue
- Preemptive and non-preemptive versions





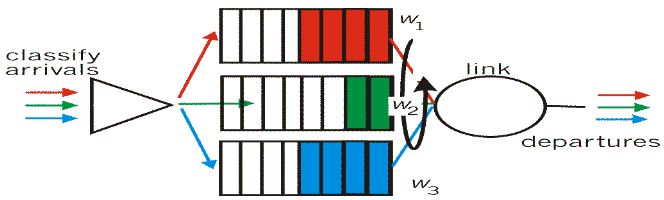
Scheduling Policies (more)

- Round Robin: scan class queues serving one from each class that has a non-empty queue



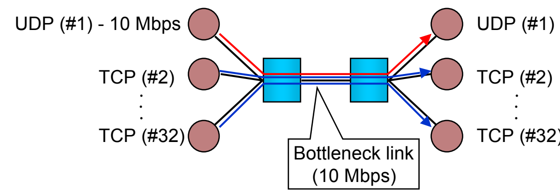
Scheduling Policies (more)

- 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

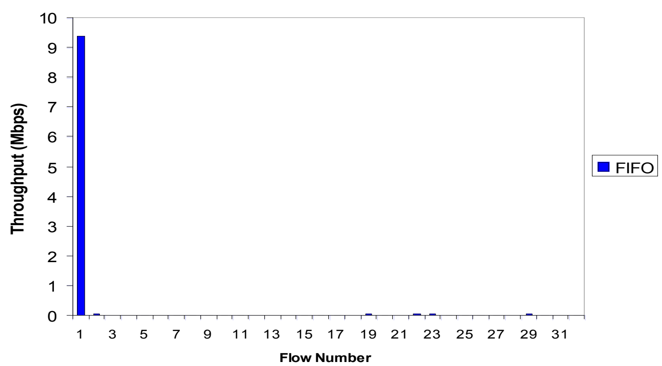


An Example

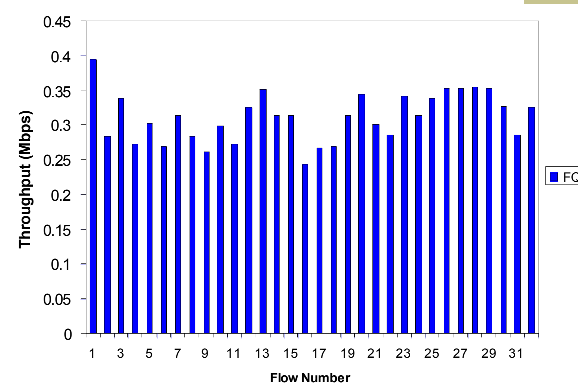
- 1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps line



Throughput of UDP and TCP Flows With FIFO



Example Outcome: Throughput of TCP and UDP Flows With Fair Queueing Router

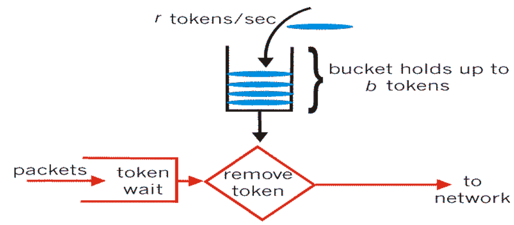


Policing Mechanisms

- 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

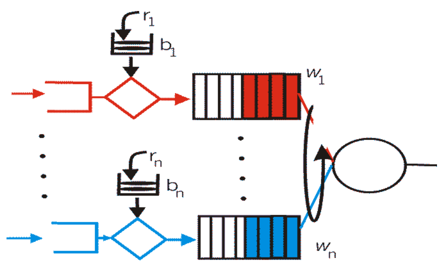
Policing Mechanisms

- 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.



Overview

- Queue management & RED
- Why QOS?
- QOS Principles
- Introduction to Scheduling Policies
- Integrated Services

Components of Integrated Services

- Type of commitment
  - What does the network promise?
- Packet scheduling
  - How does the network meet promises?
- Service interface
  - How does the application describe what it wants?
- Establishing the guarantee
  - How is the promise communicated to/from the network
  - How is admission of new applications controlled?

Type of Commitments

- **Guaranteed** service
  - For **hard real-time** applications
  - Fixed guarantee, network meets commitment if clients send at agreed-upon rate
- **Predicted** service
  - For **delay-adaptive** applications
  - Two components
    - If conditions do not change, commit to current service
    - If conditions change, take steps to deliver consistent performance (help apps minimize playback delay)
    - Implicit assumption – network does not change much over time
- **Datagram/best effort service**

Scheduling for Guaranteed Traffic

- Use **token bucket filter** to characterize traffic
  - Described by rate *r* and bucket depth *b*
- Use **Weighted Fair-Queueing** at the routers
- Parekh's bound for worst case queuing delay = b/r

Token Bucket Characteristics

- On the long run, rate is limited to r
- On the short run, a burst of size b can be sent
- Amount of traffic entering at interval T is bounded by:
  - Traffic = b + r*T
- Information useful to admission algorithm

Guarantee Proven by Parekh

- Given:
  - Flow *i* shaped with token bucket and leaky bucket rate control (depth *b* and rate *r*)
  - Network nodes do WFQ
- Cumulative queuing delay D_i suffered by flow *i* has upper bound
  - **D_i < b/r,** (where r may be much larger than average rate)
  - Assumes that Σr < link speed at any router
  - All sources limiting themselves to *r* will result in no network queuing

Sharing versus Isolation

- Impact of queueing mechanisms:
  - Isolation: Isolates well-behaved from misbehaving sources
  - Sharing: Mixing of different sources in a way beneficial to all
- FIFO: sharing
  - each traffic source impacts other connections directly
    - e.g. malicious user can grab extra bandwidth
  - the simplest and most common queueing discipline
  - averages out the delay across all flows
- Priority queues: one-way sharing
  - high-priority traffic sources have impact on lower priority traffic only
  - has to be combined with admission control and traffic enforcement to avoid starvation of low-priority traffic
- WFQ: two-way isolation
  - provides a guaranteed minimum throughput (and maximum delay)

Putting It All Together

- Assume 3 types of traffic: guaranteed, predictive, best-effort
- Scheduling: use WFQ in routers
- Each *guaranteed* flow gets its own queue
- All predicted service flows and best effort are combined into a single separate queue
  - Predictive traffic classes
    - Worst case delay for classes separated by order of magnitude
    - When high priority needs extra bandwidth – steals it from lower class
  - Best effort traffic acts as lowest priority class

Service Interfaces

- Guaranteed Traffic
  - Host specifies rate to network
  - Why not bucket size b?
    - If delay not good, ask for higher rate
- Predicted Traffic
  - Specifies (r, b) token bucket parameters
  - Specifies delay D and loss rate L
  - Network assigns priority class
  - Policing at edges to drop or tag packets
    - Needed to provide isolation – why is this not done for guaranteed traffic?
      - WFQ provides this for guaranteed traffic

Lessons

- TCP can use help from routers
  - RED → eliminate lock-out and full-queues problems
  - FQ → heavy-weight but explicitly fair to all
- QoS
  - What type of applications are there? → Elastic, adaptive real-time , and hard real-time.
  - Why do we need admission control → to maximize utility
  - How do token buckets + WFQ provide QoS guarantees?