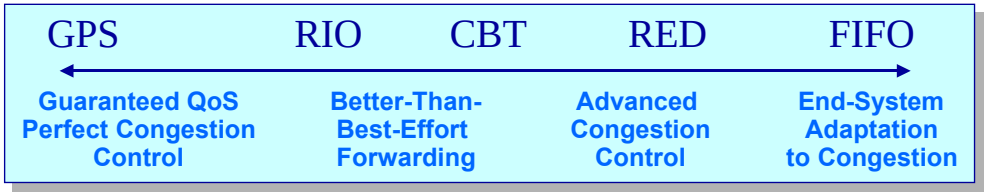
Trends in Congestion Control and Quality-of-Service

Active Queue Management on the Internet of the Future

Kevin Jeffay
University of North Carolina at Chapel Hill
Department of Computer Science
jeffay@cs.unc.edu
February 21, 2001

http://www.cs.unc.edu/Research/dirt

The Evolution of Quality-of-Service on the Internet



- The Internet is evolving to support quality-of-service
  - » Capacity allocation & inter-flow protection are required for QoS
- The current mechanisms for realizing QoS are more about router queue management than virtual circuits
  - » Active Queue Management can provide performance comparable to packet scheduling with lower state requirements and algorithmic complexity
- The Internet of tomorrow will provide router "forwarding behaviors" rather than end-to-end "services"

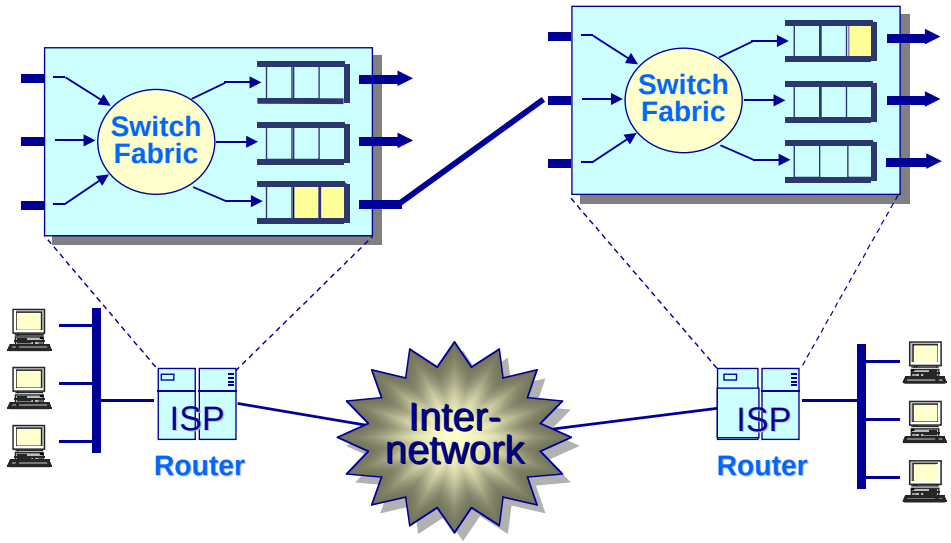
Trends in Congestion Control and Quality-of-Service

Outline

- Background
  - » Congestion control and quality-of-service on the Internet today
- Active Queue Management for advanced congestion control
  - » Random Early Detection (RED)
  - » Explicit Congestion Notification (ECN)
  - » Dealing with non-congestion-responsive sources
- Active Queue Management for service allocation
  - » "Controlled load" service
  - » "Expected capacity" service
  - » "Premium" service
  - » "Expedited forwarding" service
  - » Differentiated services (*diffserv*) architecture for the Internet

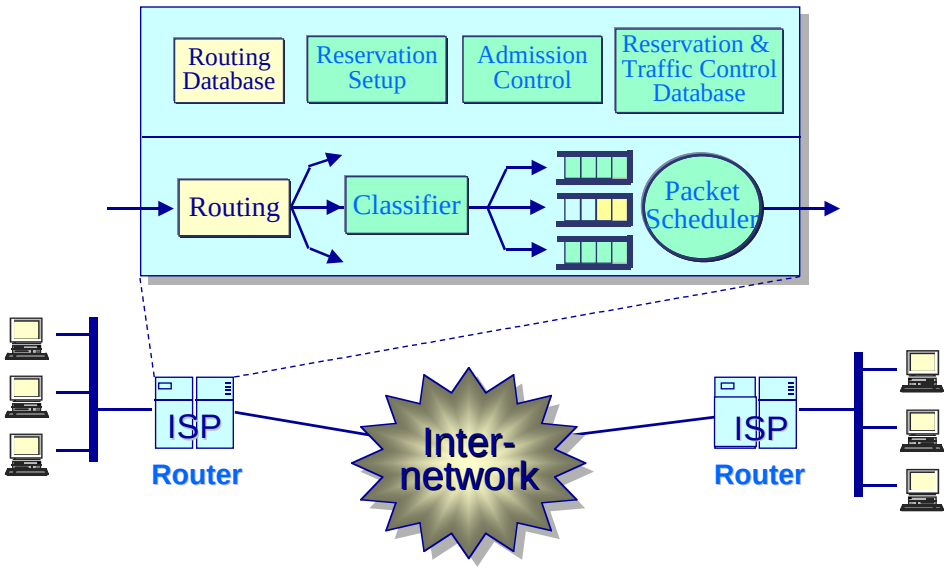
The Nature of Congestion

Queueing delays in routers



Router-Based Congestion Control

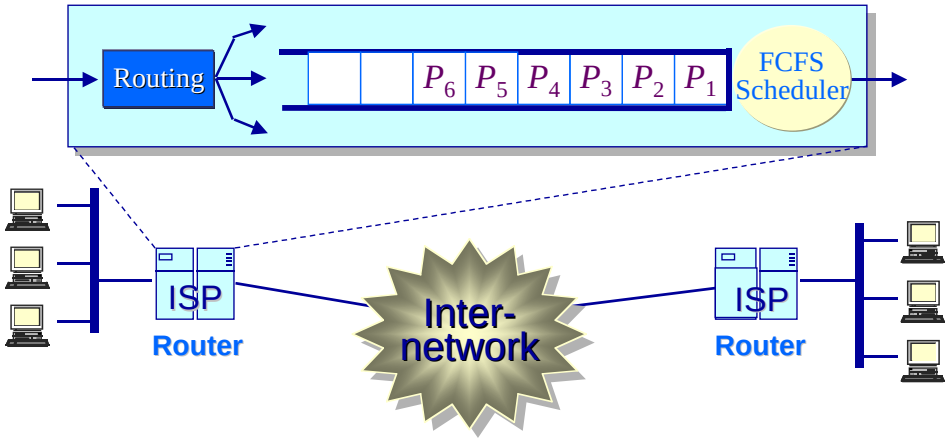
Solution 1: Open-loop congestion control



Router-Based Congestion Control

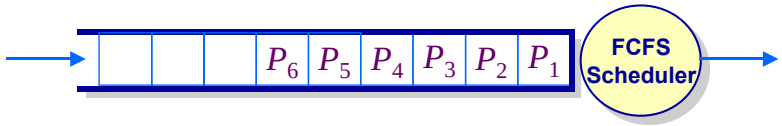
Solution 2: Closed-loop congestion control

- Normally, packets are only dropped when the queue overflows
  - » "Drop-tail" queueing



Buffer Management & Congestion Avoidance

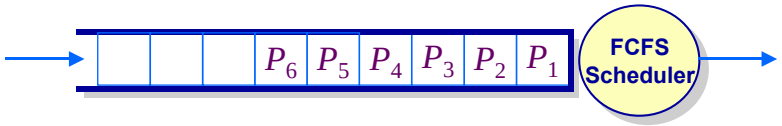
The case against drop-tail



- Large queues in routers are a bad thing
  - » End-to-end latency is dominated by the length of queues at switches in the network
- Allowing queues to overflow is a bad thing
  - » Connections that transmit at high rates can starve connections that transmit at low rates
  - » Causes connections to synchronize their response to congestion and become unnecessarily bursty

Buffer Management & Congestion Avoidance

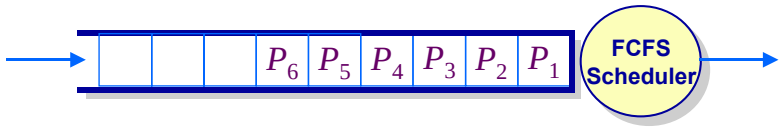
Early random packet drop



- When the queue length exceeds a threshold, packets are dropped with a fixed probability
- Claims:
  - » This should penalize connections that are transmitting at high rates
- Problems:
  - » Doesn't accommodate bursty traffic well
  - » Doesn't provide protection from misbehaving flows

Buffer Management & Congestion Avoidance

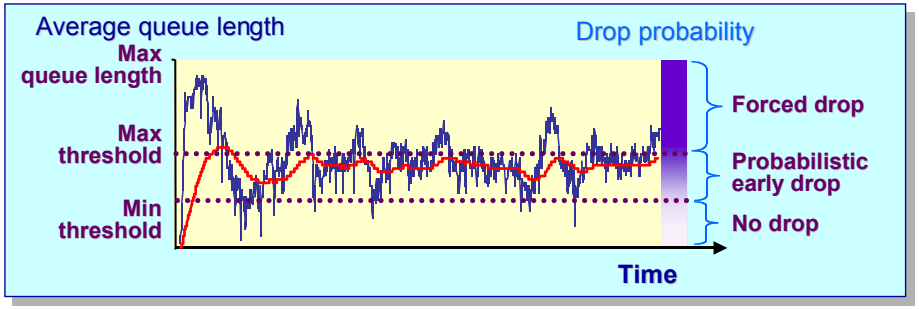
Early random packet drop



- Problems
  - » Doesn't accommodate bursty traffic well
  - » Doesn't provide protection from misbehaving flows
- Fundamental issues
  - » When to drop packets
  - » Which packets to drop
- Claim: Unless these issues are separated, you risk being biased against bursty traffic

Buffer Management & Congestion Avoidance

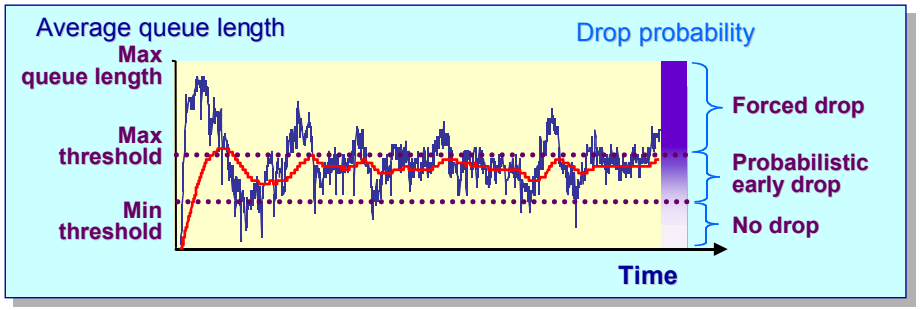
Random early detection (RED) packet drop

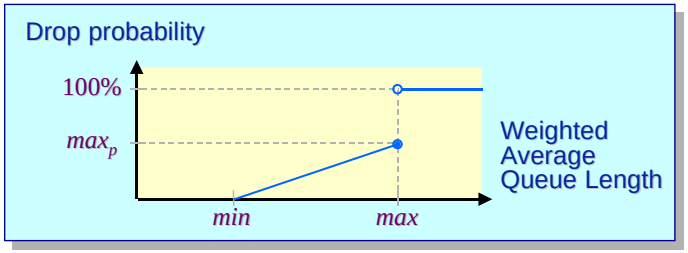


- Use an exponential average of the queue length to determine when to drop
  - » Accommodates short-term bursts
- Tie the drop probability to the weighted average queue length
  - » Avoids over-reaction to mild overload conditions

Buffer Management & Congestion Avoidance

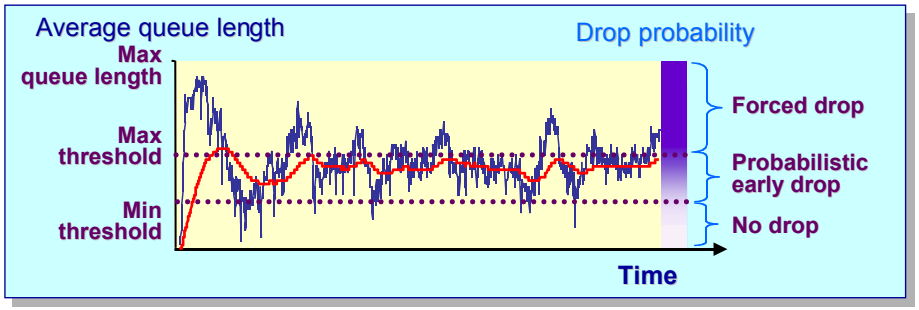
Random early detection (RED) packet drop





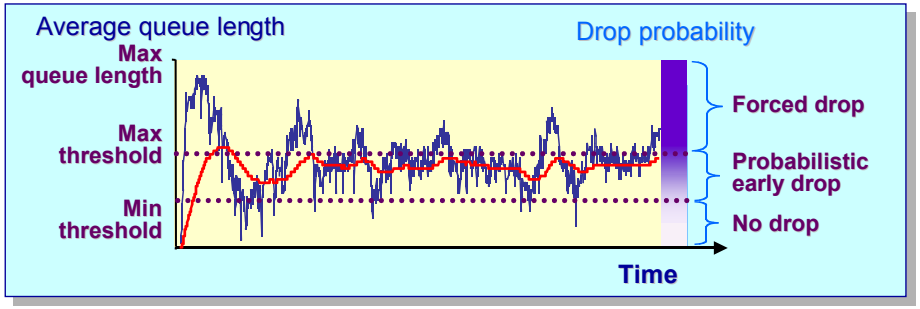
Buffer Management & Congestion Avoidance

Random early detection (RED) packet drop



- Amount of packet loss is roughly proportional to a connection's bandwidth utilization
  - » But there is no a priori bias against bursty sources
- Average connection latency is lower
- Average throughput ("goodput") is higher

Buffer Management & Congestion Avoidance
Random early detection (RED) packet drop



- RED is controlled by 5 parameters
  - » *qlen* — The maximum length of the queue
  - » w_q — Weighting factor for average queue length computation
  - » min_{th} — Minimum queue length for triggering probabilistic drops
  - » max_{th} — Queue length threshold for triggering forced drops
  - » max_p — The maximum drop probability

Random Early Detection
Algorithm

```
for each packet arrival:
   calculate the average queue size ave
   if min_th ≤ ave ≤ max_th
      calculate drop probability p_a
      drop arriving packet with probability p_a
   else if max_th ≤ ave
      drop the arriving packet
```

- The average queue length computation needs to be low pass filtered to smooth out transients due to bursts
  - » $ave = (1 - w_q)ave + w_q q$
- After idle periods, average needs to be adjusted
  - » $nbr = \frac{C}{\text{min packet size}}$ (*current time – time last packet forwarded*)
  - » $ave = (1 - w_q)^{nbr} ave$

Random Early Detection
Algorithm

```
for each packet arrival:
   calculate the average queue size ave
   if min_th ≤ ave ≤ max_th
      calculate drop probability p_a
      drop arriving packet with probability p_a
   else if max_th ≤ ave
      drop the arriving packet
```

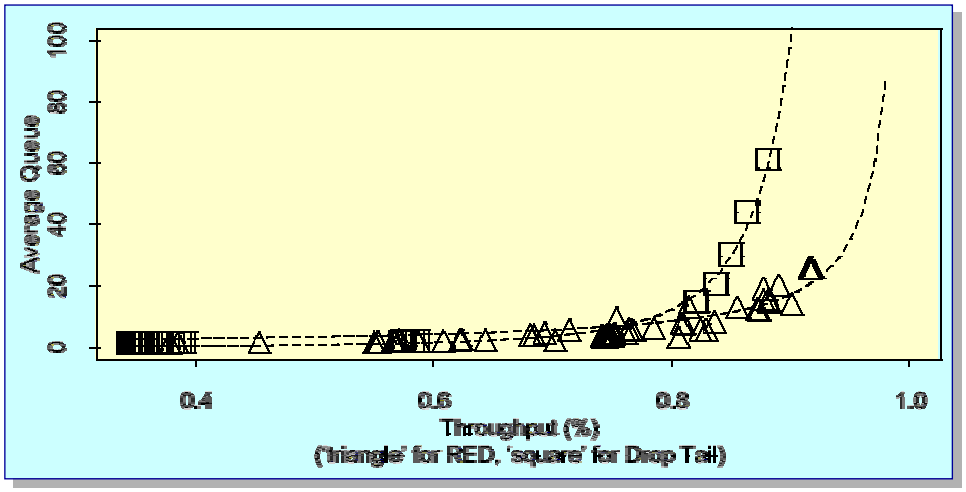
- The drop probability is computed in two steps
  - » $p_b = max_p \frac{ave - min_{th}}{max_{th} - min_{th}}$
  - » $p_a = p_b \frac{p_b}{1 - count \cdot p_b}$

Queue length can be measured in bytes rather than packets

$\Leftarrow \quad p_b = p_b \frac{packet\ size}{max\ packet\ size}$

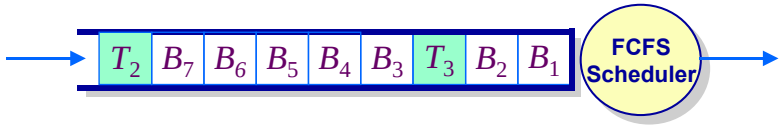
where *count* is the number of packets enqueued since the last drop

Random Early Detection
Performance



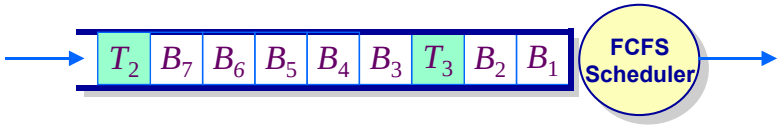
- Floyd/Jacobson simulation of two TCP (*ftp*) flows

Random Early Detection Variants
Explicit Congestion Notification (ECN)



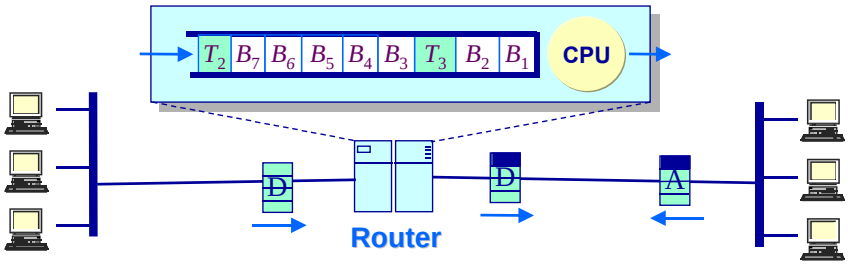
- Dropping packets is simple and effective but may penalize traffic classes unfairly
  - » What if a set of bulk transfers (*ftp*) share a congested link with a set of *telnet* connections?
    - ❖ Both will experience the same loss rate
    - ❖ But is the effect of the loss the same on each class?
- How long does it take for a sender to detect and react to congestion under RED?

Random Early Detection Variants
Explicit Congestion Notification (ECN)



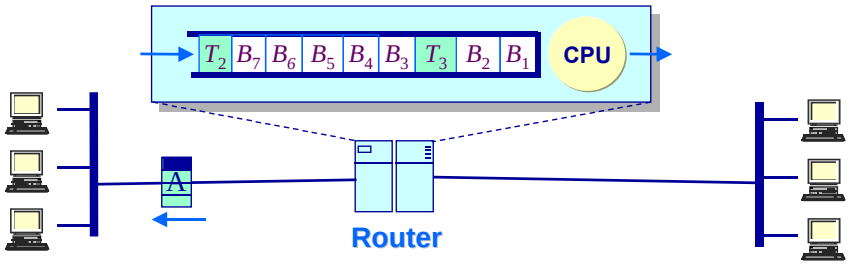
- ECN: Have a router send an explicit signal back to a sender to notify of congestion
  - » ICMP Source Quench
  - » DECbit
  - » Set a bit in a packet's header and require the receiver to inform the sender

Explicit Congestion Notification
Overview



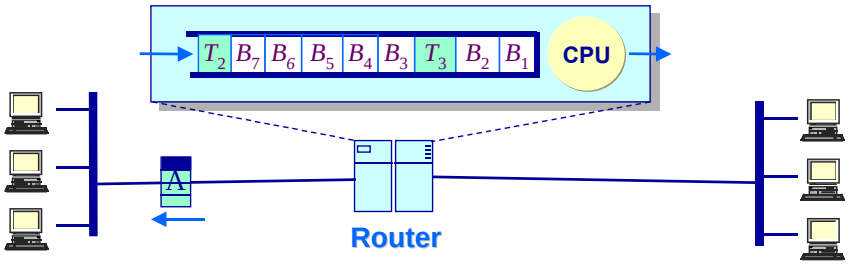
- Assume a RED router that "marks" packets rather than dropping them
- A receiver recognizes the marked packets and sets a corresponding bit in the next outgoing ACK

Explicit Congestion Notification
Overview



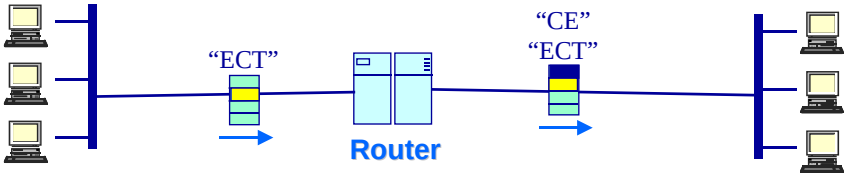
- When a sender receives an ACK with ECN it invokes a response similar to that for packet loss
  - » Halve the congestion window *cwnd* and halve the slow-start threshold *ssthresh*
  - » Continue to use ACK-clocking to pace transmission of data packets

Explicit Congestion Notification
Overview



- When a sender receives an ACK with ECN it invokes a response similar to that for packet loss
- In any given RTT, a sender should react to either ECN or packet loss *but not both*!
  - » Once a response has begun, wait until all outstanding data has been ACKed before beginning a new response

Explicit Congestion Notification
TCP details



- Two bits in the IP header are used to convey ability/willingness to respond to ECN
  - » Bits 6 and 7 in the IP *type-of-service* field
    - ❖ "ECN-Capable Transport" (ECT) bit
    - ❖ "Congestion Experienced" (CE) bit
  - » ECT bit is set by the sender on any/all packets for which ECN is requested
  - » CE bit is set by a router and never reset

Explicit Congestion Notification
TCP details

- Two bits in the TCP header are used to negotiate ECN function between end-systems
  - » ECN-Echo flag indicates that a packet was received with the CE bit set
  - » Congestion Window Reduced (CWR) flag that is used by the sender to signal its response to receipt of an ECN
    - ❖ A receiver continues to set the ECN-Echo flag until it receives a packet with the CWR flag set
- A sender sets ECN-Echo and CWR in the SYN packet to signal ECN capabilities to receiver
  - » Receiver responds with (just) the ECN-Echo set in the SYN-ACK

Trends in Congestion Control and Quality-of-Service

Outline

- Background
  - » Congestion control and quality-of-service on the Internet today
- Active Queue Management for advanced congestion control
  - » Random Early Detection (RED)
  - » Explicit Congestion Notification (ECN)
  - » Dealing with non-congestion-responsive sources
- Active Queue Management for service allocation
  - » "Controlled load" service
  - » "Expected capacity" service
  - » "Premium" service
  - » "Expedited forwarding" service
  - » Differentiated services (*diffserv*) architecture for the Internet

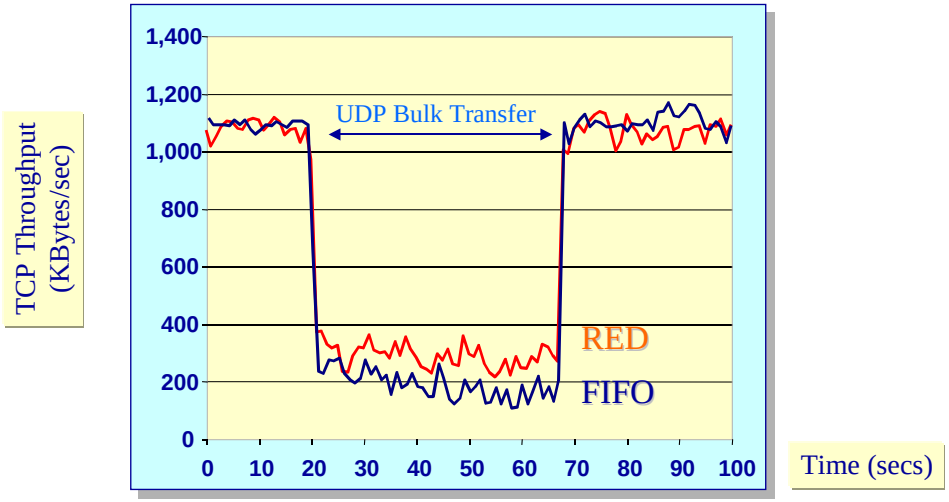
Router-Based Congestion Control

Congestion avoidance *v.* protection/fair-sharing

- RED/ECN works best for *cooperative* sources/protocols
  - » Good for sources that respond to packet loss as an indicator of congestion
- RED protects the router's queue from being persistently full
  - » RED provides a limited form of protection from "non-responsive" flows — enqueued packets experience lower delay
- But RED does not provide protection/isolation between connections
  - » Amount of packet loss is roughly proportional to a connection's bandwidth utilization
  - » But this may not matter!
  - » Non-responsive flows can trigger congestion collapse

Random Early Detection

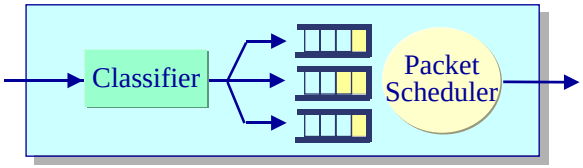
Congestion avoidance *v.* protection/fair-sharing



- TCP performance on a 10 Mbps link under RED in the face of a "UDP" blast

Router-Based Congestion Control

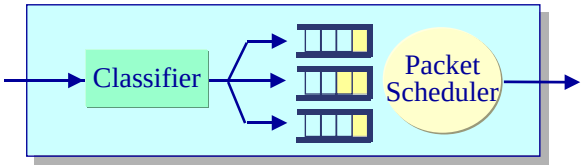
Dealing with heterogeneous/non-responsive flows



- TCP requires protection/isolation from non-responsive flows
- Solutions?
  - » Employ fair-queuing/link scheduling mechanisms
  - » Identify and police non-responsive flows
  - » Employ fair buffer allocation within a RED mechanism
- What about the so-called non-responsive flows?
  - » Are they really evil?

Dealing With Non-Responsive Flows

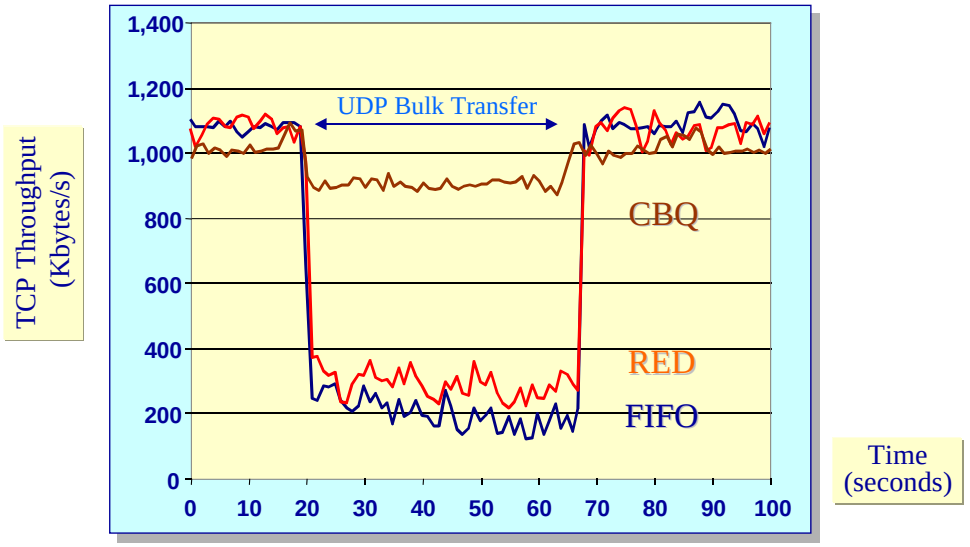
Isolating responsive and non-responsive flows



- Class-based Queuing (CBQ) (Floyd/Jacobson) provides fair allocation of bandwidth to traffic classes
  - » Separate queues are provided for each traffic class and serviced in round robin order (or weighted round robin)
  - » *n* classes each receive exactly 1/*n* of the capacity of the link
- Separate queues ensure perfect isolation between classes
  - » Class performance is only a function of the number of classes and the behavior of intra-class flows

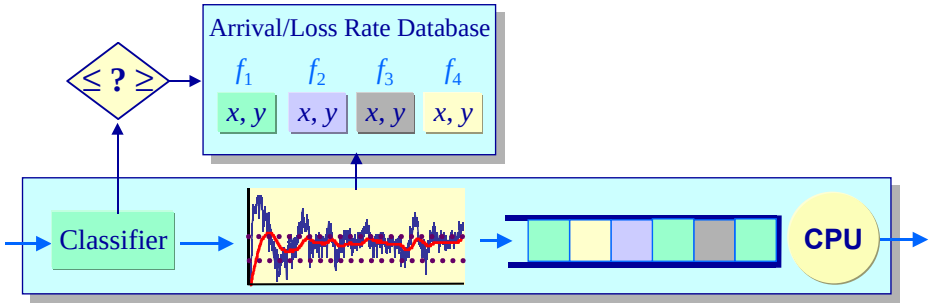
Dealing With Non-Responsive Flows

CBQ performance



Dealing With Non-Responsive Flows

Policing non-responsive flows



- Floyd/Fall: Routers should test flows for responsiveness and police those deemed to be sufficiently unresponsive
- 3 potential ways to classify non-responsive connections:
  - » TCP friendly (the "good")
  - » Unresponsive (the "bad")
  - » Disproportionate users of bandwidth (the "ugly")

Policing Non-Responsive Flows

Classifying non-responsiveness as non-TCP friendliness

- A conformant TCP implementation with an RTT R that transmits B byte packets, should transmit x bytes/sec:

$$x \leq \frac{1.5(2/3)^{0.5} \times B}{R \times p^{0.5}}$$

where p is the packet drop rate

- Rule of thumb: Police any flow whose arrival rate is greater than
  - » 1.2 x *max packet size*/(2 x *link propagation delay* x $p^{0.5}$)

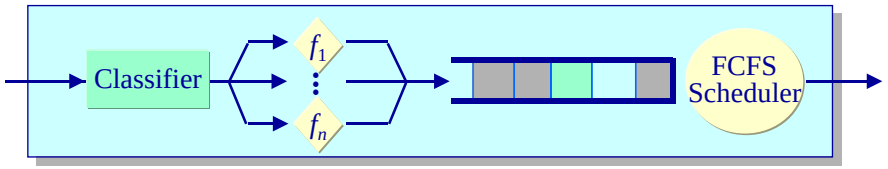
Policing Non-Responsive Flows

Classifying non-responsiveness as greedy

- A responsive flow should decrease its transmission rate in responsive to an increasing packet drop rate
  - » If the drop rate increases by a factor of x then a flow should reduce its transmission rate by a factor of at least $x^{0.5}$
- A flow is a disproportionate user of bandwidth if it consumes more than $ln(3n)/n$ of available bandwidth and has a transmission rate of greater than $c/p^{0.5}$ for a constant c
  - » Floyd/Fall: c = 12,000

Dealing With Non-Responsive Flows

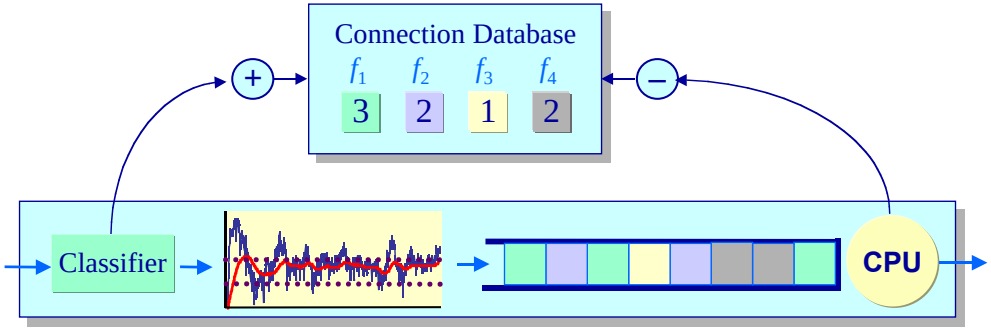
Fair buffer allocation



- Isolation can be achieved by reserving capacity for flows within a single FIFO queue
  - » Rather than maintain separate queues, keep counts of packets in a single queue
- Lin/Morris: Modify RED to perform fair buffer allocation between active flows
  - » Independent of protection issues, fair buffer allocation between TCP connections is also desirable

Dealing With Non-Responsive Flows

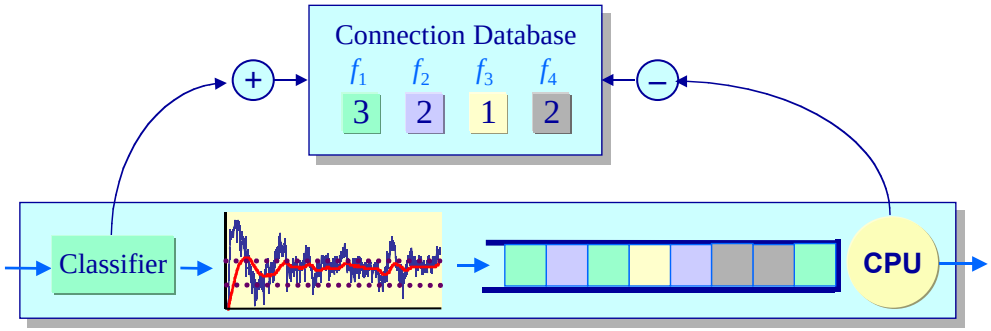
Flow Random Early Detection (FRED)



- Maintain a single FIFO queue but track the number of packets in the queue from each connection
- Subject packets from a connection to RED+ when the connection exceeds its share of the queue's capacity
  - » Drops are proportional to bandwidth used
  - » Unresponsive flows are identified and penalized

Flow Random Early Detection

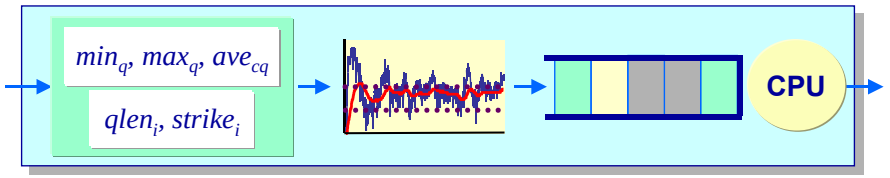
Algorithm



- New state requirements:
  - » min_q, max_q — Min and max per connection thresholds
  - » $qlen_i$ — Number of packets enqueued for connection *i*
  - » ave_{cq} — Current fair share of queue (in queue elements)
  - » $strike_i$ — Number of times a connection has attempted to enqueue more than its fair share of packets

Flow Random Early Detection

Algorithm



- Every connection can always have at least MAX(min_q, ave_{cq}) packets enqueued
- A connection can never have more than max_q packets enqueued
- If the router is congested then a connection can have at up to $2/n$ buffer locations but only if it has never attempted to exceed max_q in the past
  - » Well-behaved connections can burst to twice their fair share
- If the router is congested the actual number of packets enqueued depends on RED

Flow Random Early Detection

Algorithm

```
for each packet arrival from connection i:
    calculate the average queue size ave
    if ave ≥ max_th then max_q = 2  else max_q = min_th

    if qlen_i ≥ max_q or (ave ≥ max_th and qlen_i > 2*ave_cq)
       or (qlen_i > ave_cq and strike_i > 1)
       strike_i++
       drop arriving packet and return

    if min_th ≤ ave < max_th
       if qlen_i ≥ MAX(min_q, ave_cq)
          calculate drop probability & apply to arriving packet
    else if max_th ≤ ave
       drop the arriving packet and return

    if nbr_flows > 0 then ave_cq = ave/nbr_flows
                     else ave_cq = ave
```

Set the max share

Police aggressive flows

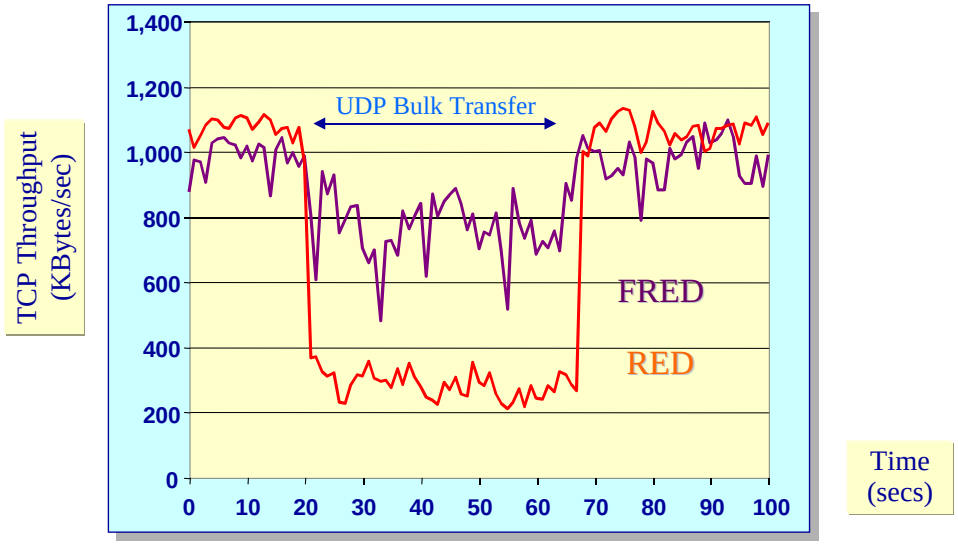
Apply RED test

Set the fair share

- Only new configuration parameter is min_q
  - » Sensible value is min_q = 2-4 packets

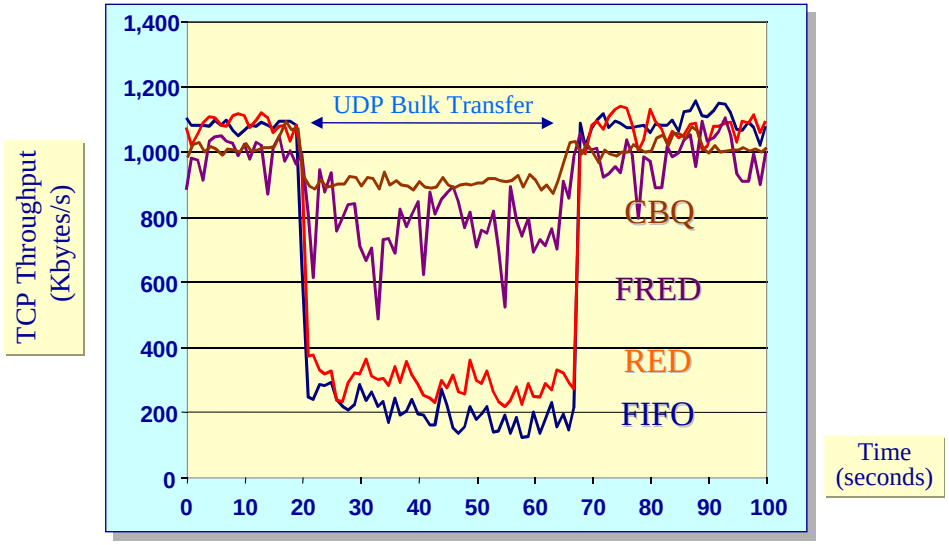
Flow Random Early Detection

Performance in the face of non-responsive flows



Congestion Avoidance *v.* Fair-Sharing

TCP throughput under different queue management schemes



- TCP performance as a function of the state required to ensure/approximate fairness

Router-Based Congestion Control

Open issues

- IETF recommends that RED be deployed in routers today

  "All available empirical evidence shows that the deployment of active queue management mechanisms in the Internet would have substantial performance benefits. There are seemingly no disadvantages to using the RED algorithm, and numerous advantages. Consequently, we believe that RED active queue management algorithm should be widely deployed."

- But…
  - » Protection/isolation/fair-sharing issues remain
    - ❖ May lead to the development of more aggressive mechanisms for policing non-TCP conformant traffic
    - ❖ The performance of real-time UDP-based applications may get worse before it gets better
  - » Potential bias against short-lived flows may be present
    - ❖ Is RED bad for the Web?
    - ❖ Is RED/ECN any better?

Trends in Congestion Control and Quality-of-Service

Outline

- Background
  - » Congestion control and quality-of-service on the Internet today
- Active Queue Management for advanced congestion control
  - » Random Early Detection (RED)
  - » Explicit Congestion Notification (ECN)
  - » Dealing with non-congestion-responsive sources
- Active Queue Management for service allocation
  - » "Controlled load" service
  - » "Expected capacity" service
  - » "Premium" service
  - » "Expedited forwarding" service
  - » Differentiated services (*diffserv*) architecture for the Internet

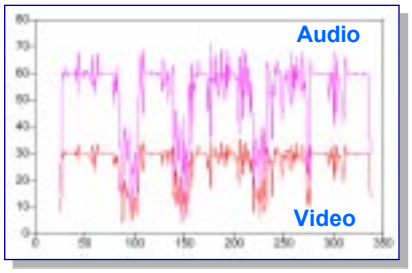
Service Allocation Models for the Internet

Concept

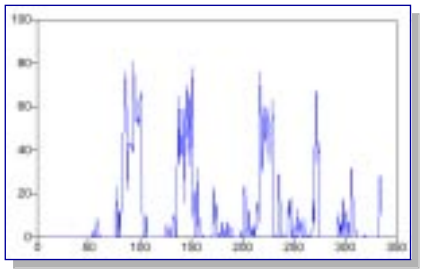
- Congestion control is largely about avoiding congestion collapse
- ISPs need it
  - » Effective congestion control can lead to higher link utilization
- End users need it
  - » Effective congestion control results in higher application-level throughput
- But is it enough?
  - » Service allocation concerns going beyond a fair, best-effort for all, forwarding service
  - » Providing a "better-than-best-effort" service

Service Allocation Models for the Internet

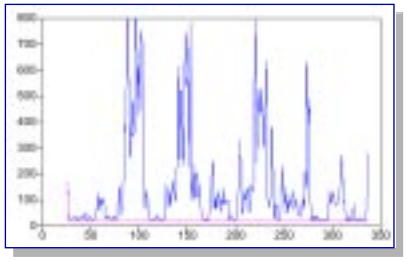
What happens when one's fair share is not enough?



Throughput (frames/sec)



Packet Loss

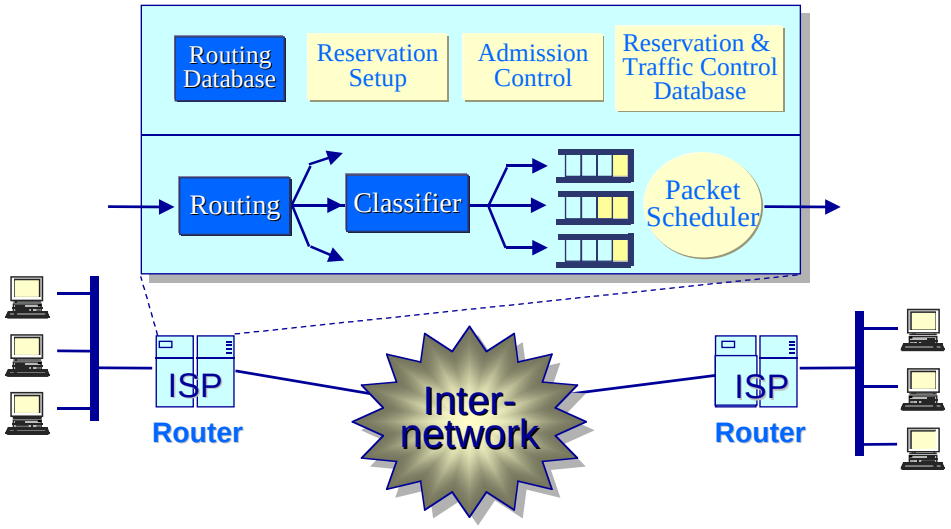


Audio Latency (ms)

- Example: Performance of ProShare™ transmission over the Internet (300 kbps)
  - » Frozen video
  - » Clipped, broken audio

Service Allocation Models for the Internet

The Integrated Services Architecture for the Internet



- INTSERV — *Every* router reserves and maintains state for *every* non-best-effort connection

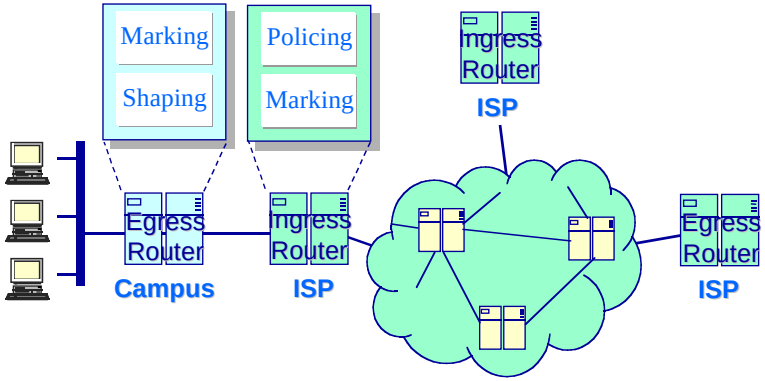
Service Allocation Models for the Internet

Towards a better-than-best-effort service

- So if guarantees are "too much," what's "just enough"?
- IETF proposal: A *controlled load* service
  - » A service that approximates the service a flow would receive under "unloaded conditions" in the network
- In a controlled load service, applications can assume:
  - » A (very) high percentage of transmitted packets will be delivered
  - » A high percentage of transmitted packets will experience a transit delay not significantly greater then the minimum transit delay experienced by any packet

Towards a Better-Than-Best-Effort Service

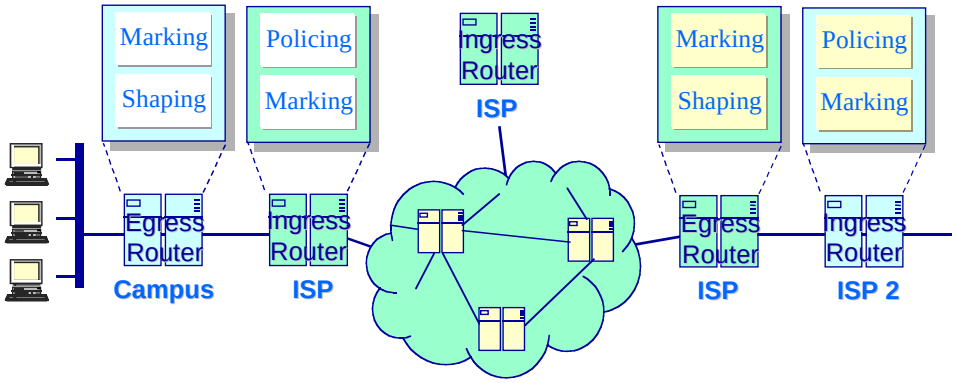
Architectural principles



- Shift in emphasis from per-flow contracts to per-aggregate contracts
  - » All state is maintained at the edges of the network
  - » No new state inside a provider's network
- A campus aggregates traffic that conforms to a "service profile"

Towards a Better-Than-Best-Effort Service

Architectural principles

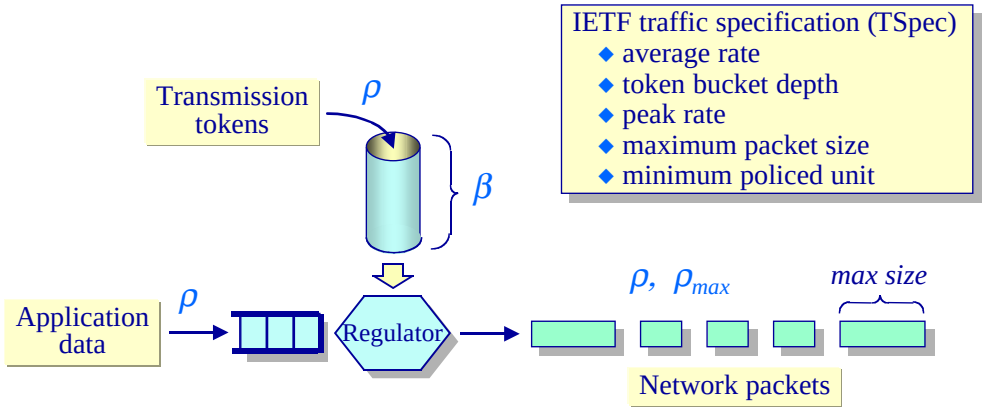


- An ISP policies marked traffic to ensure its compliance with the profile
- An end-user must be able to verify the actual performance it receives
- Service agreements stitched together from bilateral agreements

Towards a Better-Than-Best-Effort Service

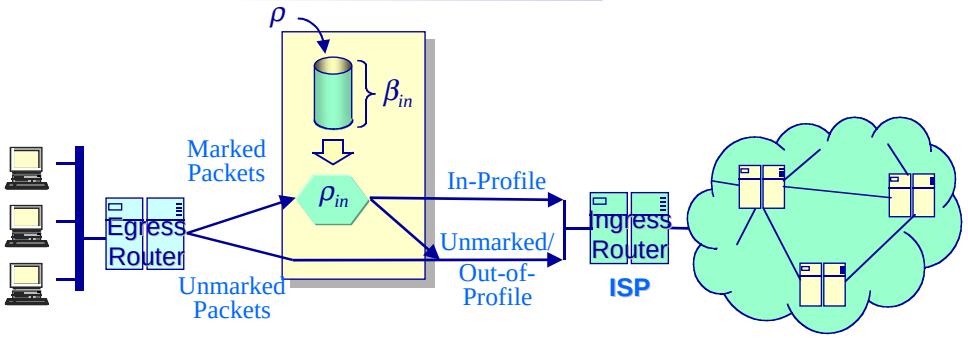
Service profiles

- To receive a service contract an application must specify the service it requires and the traffic it will generate
  - » Canonical flow specification — *the token bucket*



Towards a Better-Than-Best-Effort Service

The Clark *et al.* "expected capacity" service

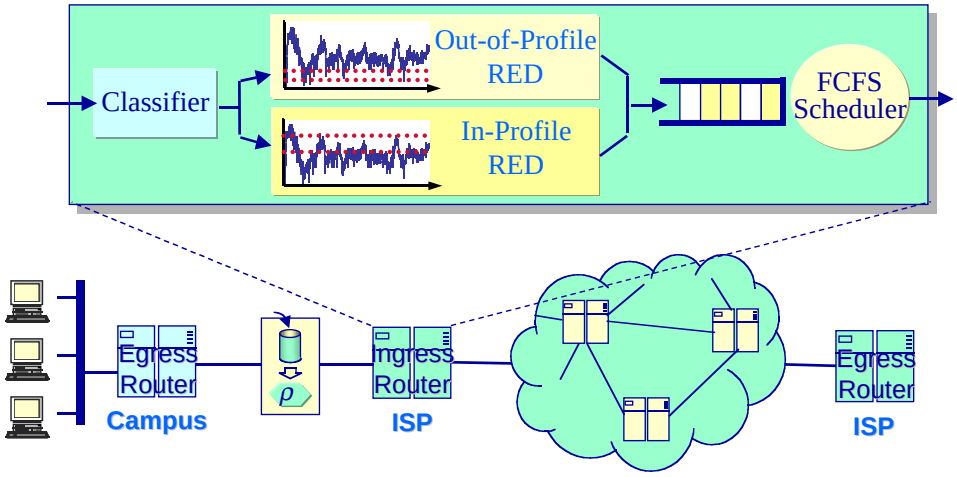


- ISPs allocate capacity for marked flows
- Campus marks packets for "regular" or "assured" service
- A policer checks arriving flows compliance against profile
  - » Conformant "in profile" packets forwarded unchanged
  - » Non-conformant "out of profile" packets demoted to best-effort

The Clark *et al.* "Expected Capacity" Service

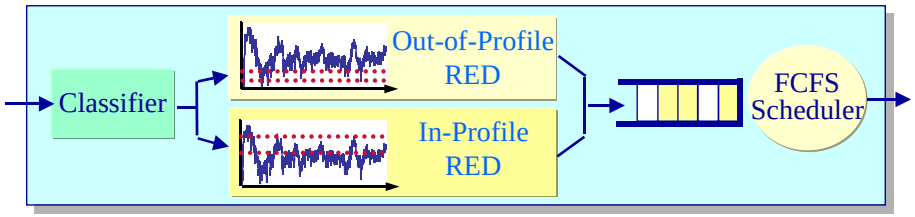
RED with In/Out (RIO)

- Ingress router runs two RED packet droppers in parallel
  - » Apply "harsh RED" to out-of-profile packets & unmarked packets
  - » Apply "lenient RED" to in-profile packets



The Clark *et al.* Expected Capacity Service

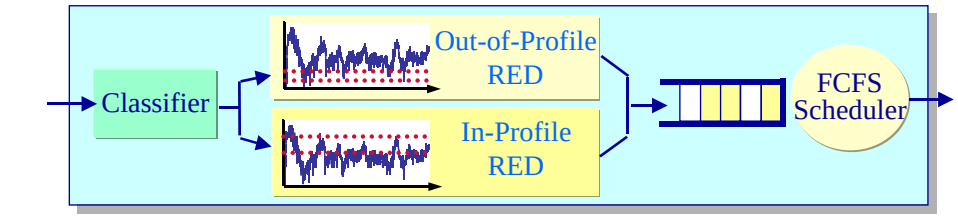
RED with In/Out (RIO)



- The "In" RED engine tracks the average number of in-profile packets in the queue (*ave_in*)
  - » Also uses separate values min_{th}_in, max_{th}_in, and P_{max}_in
- The "Out" RED engine is a "normal" RED engine
  - » *ave* is the total number of packets in the queue (In + Out)
  - » In general
    - ❖ $min_{th}\_out < min_{th}\_in$
    - ❖ $max_{th}\_out < max_{th}\_in$
    - ❖ $P_{max}\_out > P_{max}\_in$

The Clark *et al.* Expected Capacity Service

RED with In/Out (RIO)



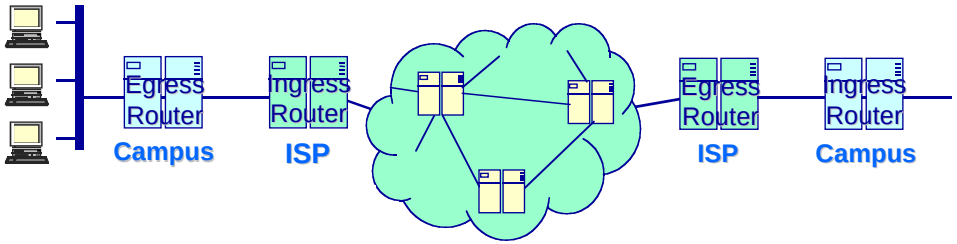
- Under RIO, in-profile marked traffic can always occupy at least min_{th}_in queue locations
- Thus in-profile traffic is allocated at least bandwidth

$$B_{in} = \frac{P \times min_{th}\_in}{P \times max_{th}\_out} C$$

where C is the link capacity and P is the average packet size

The Clark *et al.* Expected Capacity Service

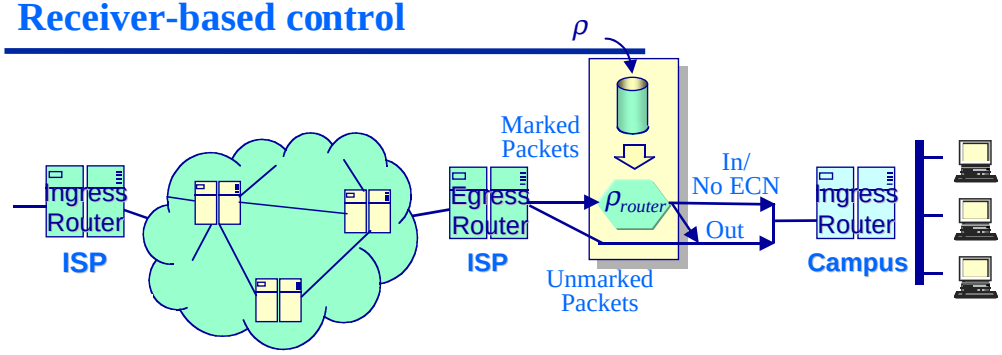
Sender-based *v.* Receiver-based control



- Current scheme "charges" the sender for the transmission service
- Might it not make sense to charge the receiver?
  - » Might not the receiver want to control how much data it receives?

The Clark *et al.* Expected Capacity Service

Receiver-based control



- Use ECN to signal the *receiver* of congestion
  - » Routers run the "normal" RED with ECN algorithm
- A profile meter at the receiver checks the compliance of the arriving stream with the profile
  - » Arriving in-profile packets have the ECN bit cleared (if set)
- Surviving ECN information is fed back to the sender

The Clark *et al.* Expected Capacity Service

Issues

- Specification of the expected capacity
  - » Specification for individual flows or aggregates
- Specification of the end-point of the service
  - » How can a flow ensure that it gets bandwidth to the network it desires?
- Is one service model enough?
  - » Assured service is primarily a throughput service
  - » How about a service for latency sensitive applications?

Trends in Congestion Control and Quality-of-Service

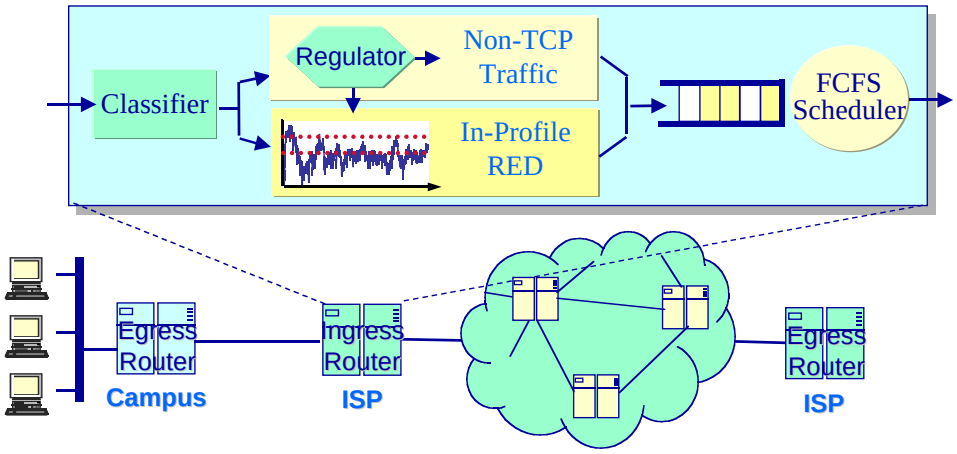
Outline

- Background
  - » Congestion control and quality-of-service on the Internet today
- Active Queue Management for advanced congestion control
  - » Random Early Detection (RED)
  - » Explicit Congestion Notification (ECN)
  - » Dealing with non-congestion-responsive sources
- Active Queue Management for service allocation
  - » "Controlled load" service
  - » "Expected capacity" service
  - » "Premium" service
  - » "Expedited forwarding" service
  - » Differentiated services (*diffserv*) architecture for the Internet

Realizing a "Premium" Service

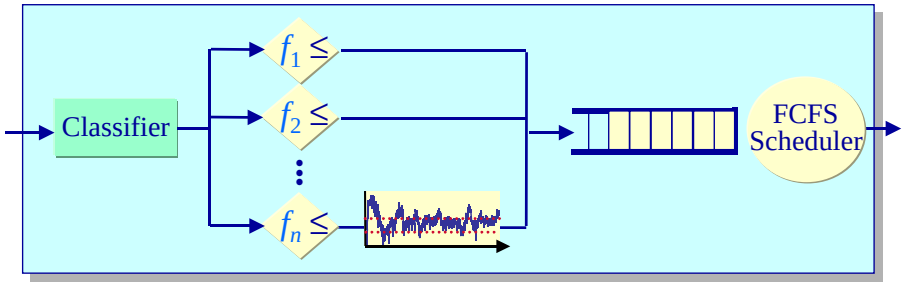
Can it be done with a single queue?

- The Clark *et al.* RIO scheme can be extended to provide a premium service
  - » Can also be made more resilient to unresponsive flows



Premium Service via Queue Management

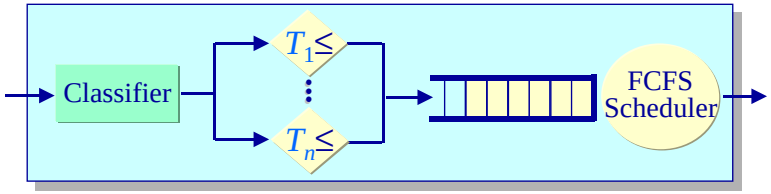
"Class-based thresholds"



- Designate a set of traffic classes and allocate a fraction of a router's buffer capacity to each class
- Once a class is occupying its limit of queue elements, discard *all* arriving packets
- Within a traffic class, further active queue management may be performed

Class-Based Thresholds

Analysis



- A CBT router is parameterized by:
  - » n, the number of classes
  - » $\{T_1, T_2, \ldots, T_n\}$ a set of class thresholds
- If class i is allocated capacity T_i then it will receive at least bandwidth

$$B_i = \frac{P_i T_i}{\sum_{j=1}^{n} P_j T_j} C$$

where C is the link capacity and P_i is the average class i packet size

Class-Based Thresholds

Analysis

- The bandwidth actually received by a class is a function of that consumed by other classes
- Let $w_i = B_i/C$ be the "weight" of traffic class i
  - » The expected link utilization of class i traffic
- If class j consumes ($load_j < B_j$) then class i receives at least bandwidth

$$B_i' = B_i + \frac{w_i}{\sum_{\substack{k=1 \\ k \neq j}}^{n} w_k} (B_j - load_j)$$

- CBT ensures weighted MAX-MIN fair allocation of bandwidth

Fairness

Max-min fair share

- Consider a set of n flows that require c_1, c_2, …, c_n bits per second of bandwidth
- "Fairness" implies that…
  - » No flow receives more bandwidth than it requires
  - » If a flow receives less bandwidth than it requires then it receives the same amount of bandwidth as all other unsatisfied flows

Initially each process gets C/n of the link's capacity.

Fairness

Max-min fair share

- Consider a set of n flows that require c_1, c_2, …, c_n bits per second of bandwidth
- "Fairness" implies that…
  - » No flow receives more bandwidth than it requires
  - » If a flow receives less bandwidth than it requires then it receives the same amount of bandwidth as all other unsatisfied flows

Initially each process gets C/n of the link's capacity.
If $c_1 < C/n$ then the unused $C/n - c_1$ is reallocated.

Fairness

Max-min fair share

- Consider a set of n flows that require c_1, c_2, …, c_n bits per second of bandwidth
- "Fairness" implies that…
  - » No flow receives more bandwidth than it requires
  - » If a flow receives less bandwidth than it requires then it receives the same amount of bandwidth as all other unsatisfied flows

Initially each process gets C/n of the link's capacity.
If $c_1 < C/n$ then the unused $C/n - c_1$ is reallocated such that flows c_2-c_n receive

$$C/n + \frac{C/n - c_1}{n - 1}$$

of the link's capacity.

Fairness

Max-min fair share

- Consider a set of n flows that require c_1, c_2, …, c_n bits per second of bandwidth
- "Fairness" implies that…
  - » No flow receives more bandwidth than it requires
  - » If a flow receives less bandwidth than it requires then it receives the same amount of bandwidth as all other unsatisfied flows

Initially each process gets C/n of the link's capacity.
If $c_1 < C/n$ and $c_2 < C/n$ then the unused bandwidth is reallocated such that flows c_3-c_n receive

$$C/n + \frac{C/n - c_1}{n - 1} + \frac{C/n + (C/n - c_1)/(n-1) - c_2}{n - 2}$$

of the link's capacity.

Fairness

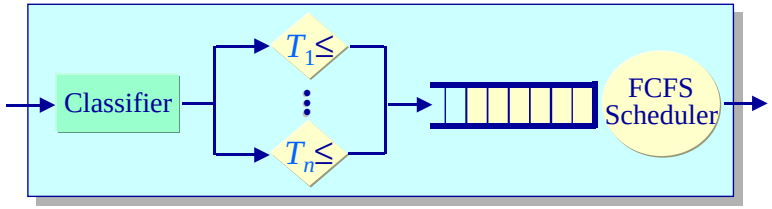
Weighted max-min fair share

- Consider a set of n flows with $w_1, \ldots, w_n$ which represent the relative importance of each flow
- Weighted fairness implies that…
  - » Resources are allocated in order of increasing demand normalized by weight
  - » If a flow receives less bandwidth than it requires then it receives a share of the bandwidth in proportion to its weight

> Connection i with weight w_i and resource requirements c_i is treated as w_i connections with demand c_i. Weighted max-min fairness reduces to "regular" max-min fairness on the resulting $w_1 + w_2 + \ldots + w_n$ logical flows.

Class-Based Thresholds

Analysis



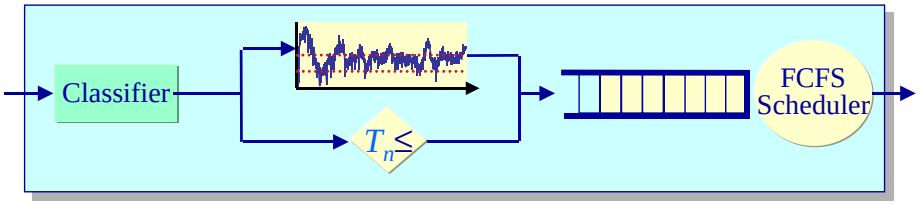
- All traffic classes experience the same worst case delay bound

$$D = \frac{1}{C} \sum_{j=1}^{n} P_j T_j$$

- Thus CBT trades link utilization for delay bounds

Class-Based Thresholds

Implementation & evaluation



- CBT is implemented in Alt-Q on FreeBSD
- The UNC implementation supports three traffic classes:
  - » TCP
  - » marked non-TCP ("well behaved UDP")
  - » non-marked non-TCP (all others)
- Subject TCP flows to RED and non-TCP flows to a weighted average queue occupancy threshold test

Class-Based Thresholds

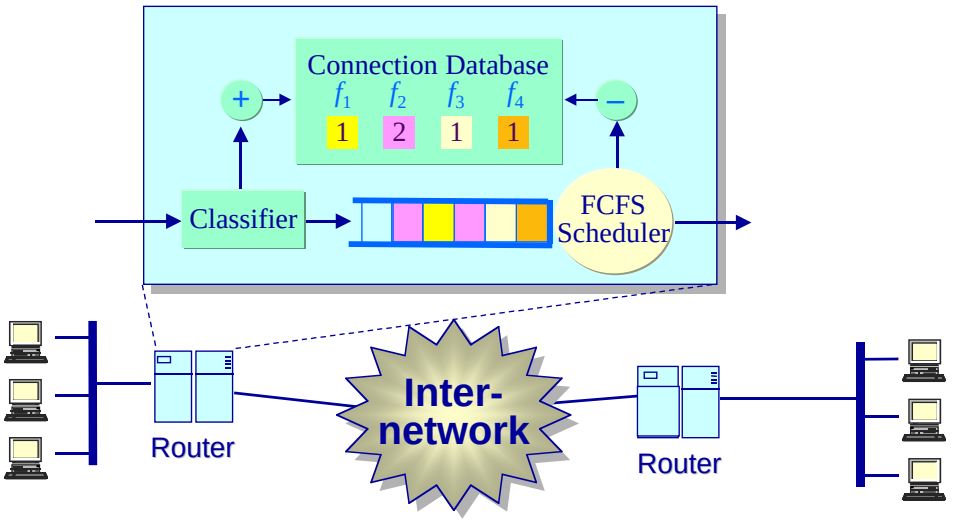
Evaluation

- Compare:
  - » FIFO queuing
  - » RED
  - » CBT
  - » Fair buffer allocation

CBT Evaluation

Fair buffer allocation (FRED)

- Flow Random Early Detection [Lin & Morris 97]



Class-Based Thresholds

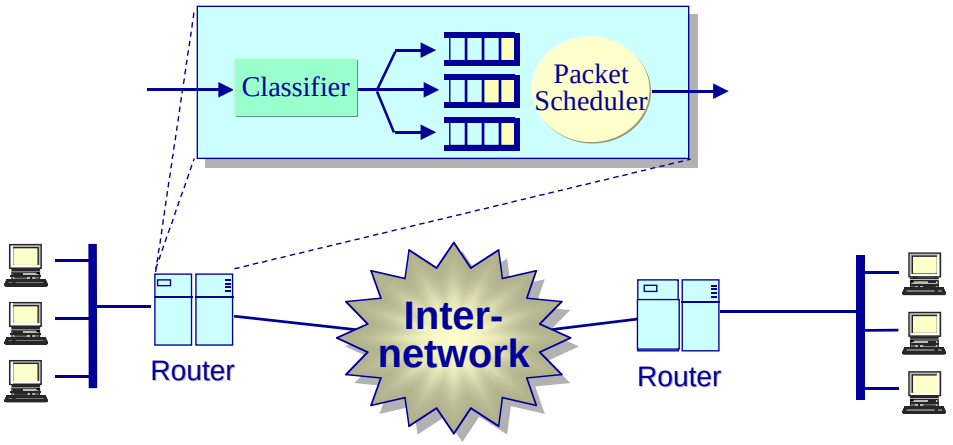
Evaluation

- Compare:
  - » FIFO queuing
  - » RED
  - » CBT
  - » Fair allocation of buffers (FRED)
  - » Packet scheduling

CBT Evaluation

Packet scheduling

- Class-based queuing [Floyd & Jacobson 95]



Class-Based Thresholds

Evaluation

- Compare:
  - » FIFO queuing (Negative baseline)
  - » RED (The Internet of tomorrow)
  - » FRED (RED + Fair allocation of buffers)
  - » CBT
  - » CBQ (Positive baseline)

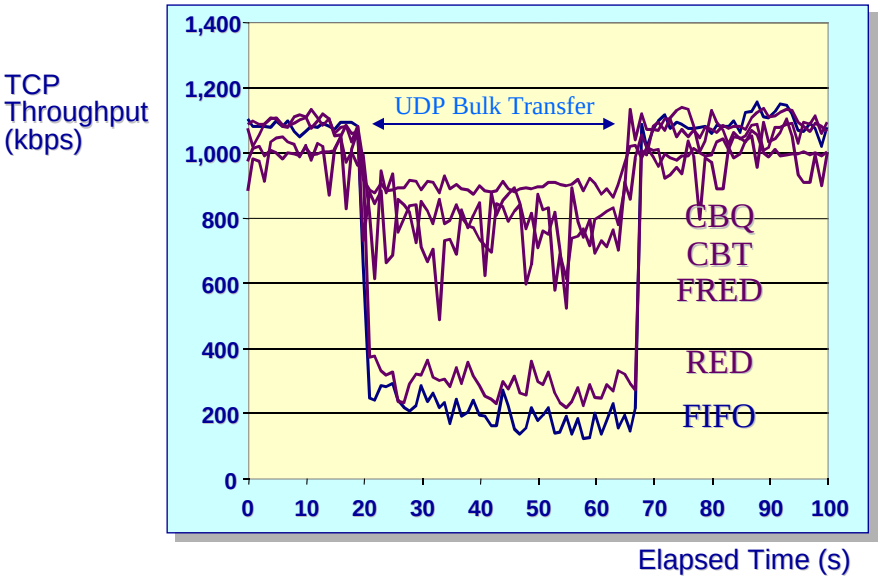
CBT Evaluation
Experimental design

- Share a 10 Mbps link between:
  - » 3,000 simulated users browsing the Web (8-9 Mb/s of HTTP traffic)
  - » 6-10 marked UDP ProShare flows
  - » 1 unmarked UDP bulk transfer

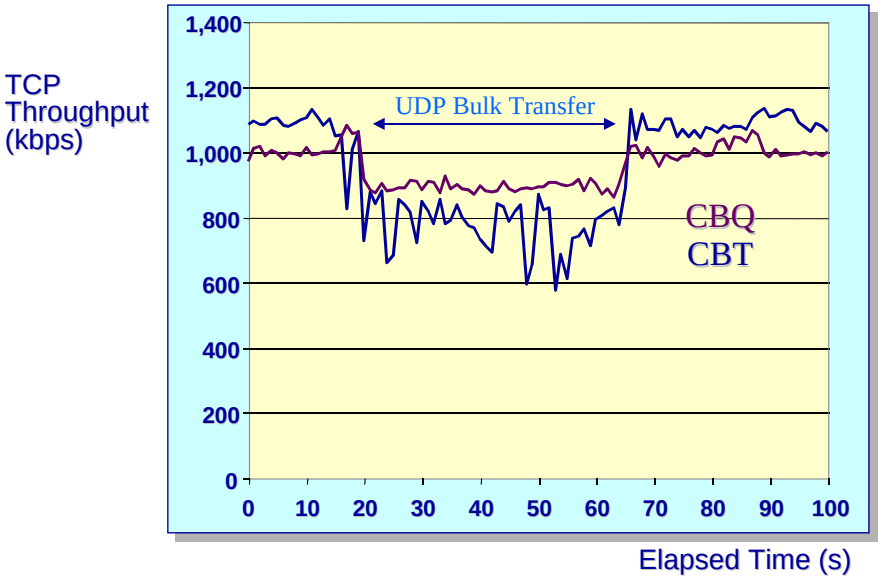
CBT Evaluation
Experimental design

- Performance metrics:
  - » Aggregate TCP throughput
  - » ProShare latency and loss
  - » Algorithm complexity & state requirements

CBT Evaluation
TCP Throughput

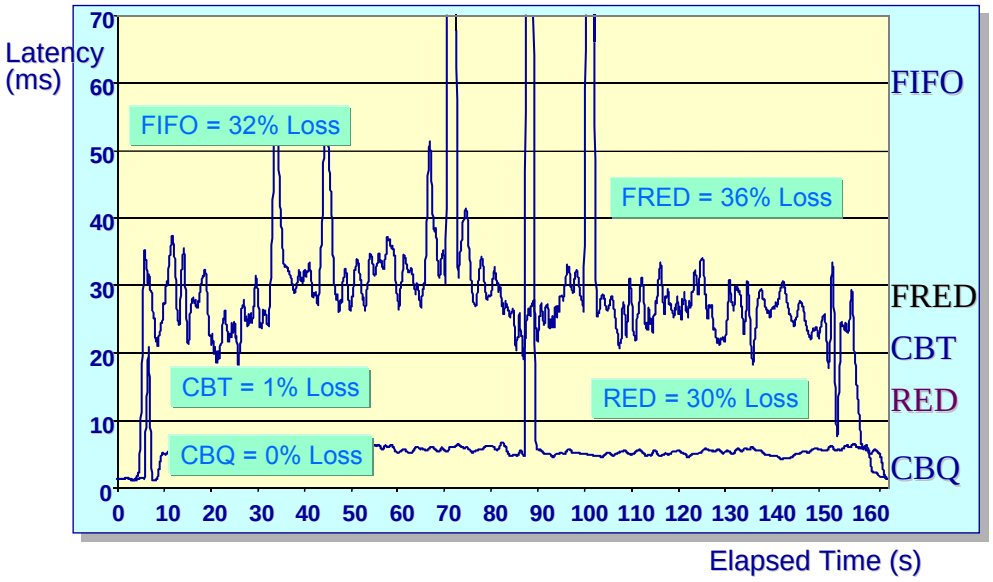


CBT Evaluation
TCP Throughput



CBT Evaluation

ProShare (marked UDP) latency



Trends in Congestion Control and Quality-of-Service

Outline

- Background
  - » Congestion control and quality-of-service on the Internet today
- Active Queue Management for advanced congestion control
  - » Random Early Detection (RED)
  - » Explicit Congestion Notification (ECN)
  - » Dealing with non-congestion-responsive sources
- Active Queue Management for service allocation
  - » "Controlled load" service
  - » "Expected capacity" service
  - » "Premium" service
  - » "Expedited forwarding" service
  - » Differentiated services (*diffserv*) architecture for the Internet

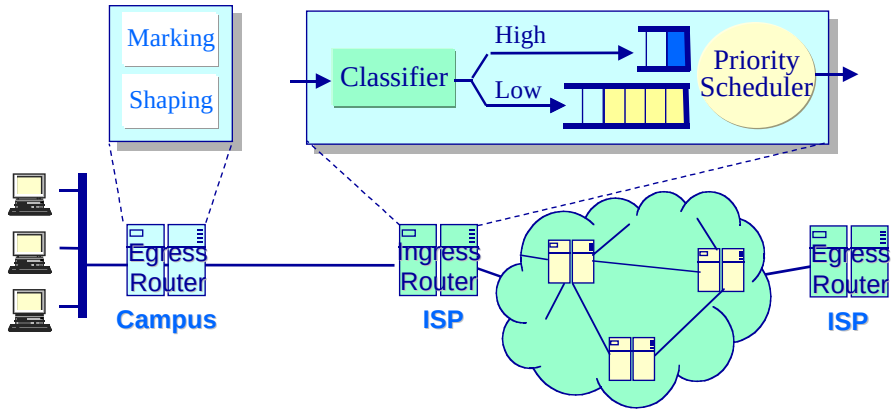
Service Allocation Models for the Internet

Issues

- Specification of the expected capacity
  - » Specification for individual flows or aggregates
- Specification of the end-point of the service
  - » How can a flow ensure that it gets bandwidth to the network it desires?
- Is one service model enough?
  - » Assured service is primarily a throughput service
  - » How about a service for latency sensitive applications?

The Nichols/Jacobson "Two Bit" Architecture

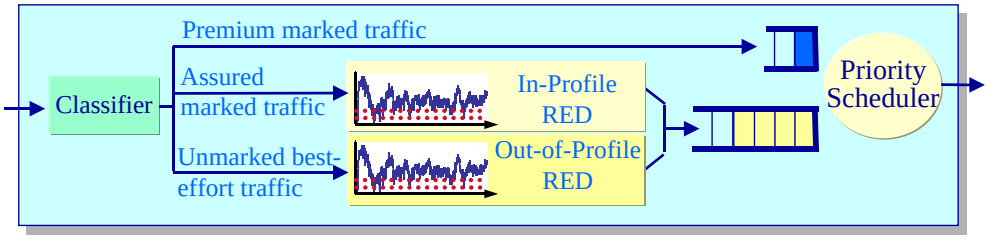
The "expedited forwarding" service



- ISPs allocate and sell capacity for a "premium" service
  - » Packets are marked and policed according to a service profiles
- Premium service is realize by simple priority scheduling

The Nichols/Jacobson "Two Bit" Architecture

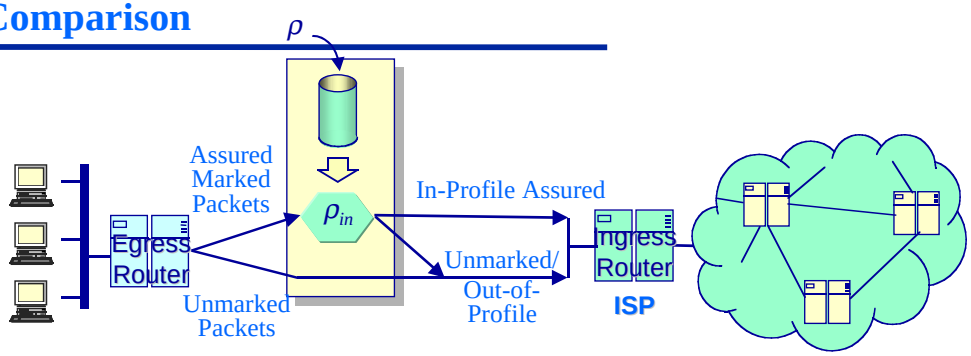
Expedited and assured services



- The *assured* service is easily supported within the low priority queue
  - » Packets are marked and policed according to service profiles as before
- Thus two bits can be used to mark traffic

Assured and Expedited Service

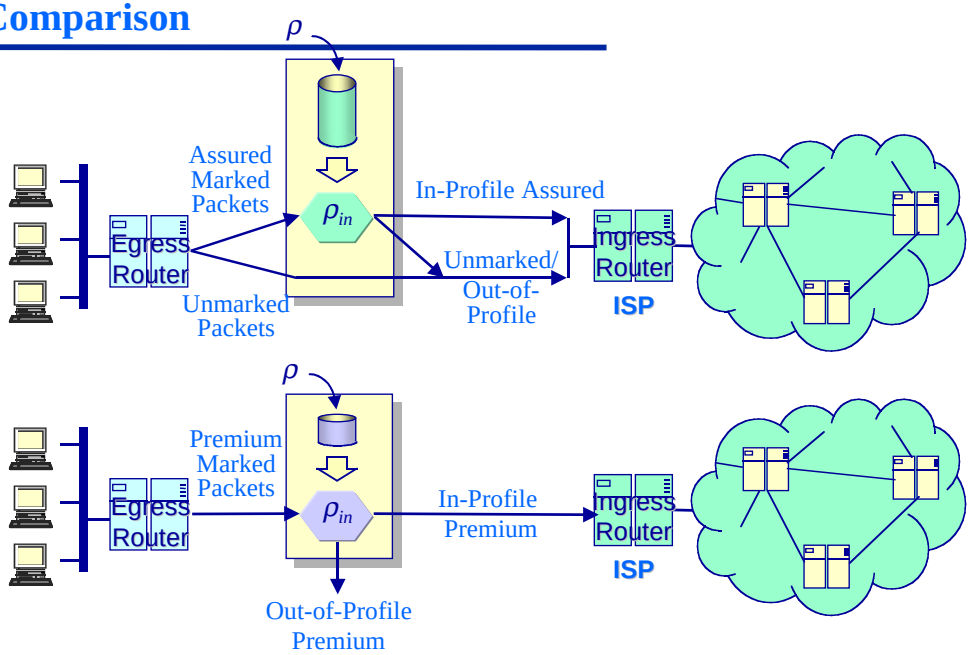
Comparison



- The difference between *assured* and *expedited* services are in the way in capacity is allocated and in the way they are policed
  - » Assured capacity is provisioned/policed according to expected demand
  - » Premium capacity is provisioned/policed according to peak demand

Assured and Expedited Service

Comparison



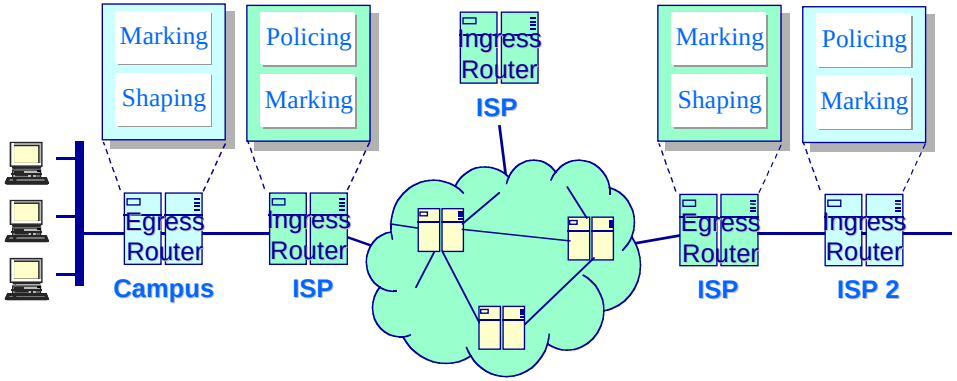
Service Allocation Models for the Internet

Issues

- Specification of the expected capacity
  - » Specification for individual flows or aggregates
- Specification of the end-point of the service
  - » How can a flow ensure that it gets bandwidth to the network it desires?
- Is one service model enough?
  - » Assured service is primarily a throughput service
  - » How about a service for latency sensitive applications?

Bandwidth Allocation

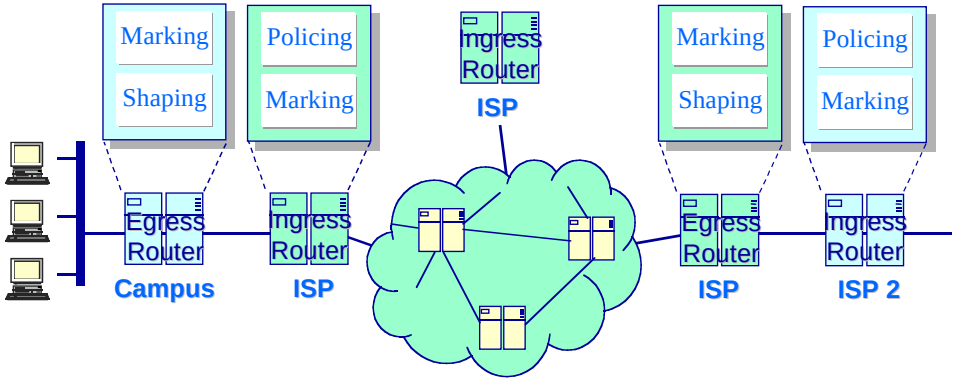
Signaling issues



- Our conceptual model to date is that ISPs statically configure themselves to offer better-than-best-effort services between themselves
- End-to-end services realized through bilateral agreements

Bandwidth Allocation

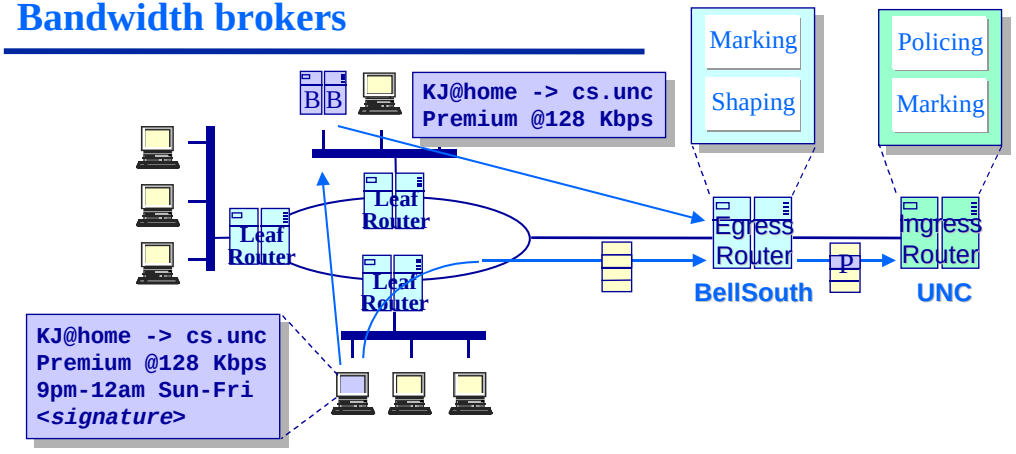
Signaling issues



- Issues:
  - » Identifying flows that are authorized to receive services
  - » Communicating and managing state information in border routers
  - » Coordinating bandwidth allocation in neighboring networks

Bandwidth Allocation

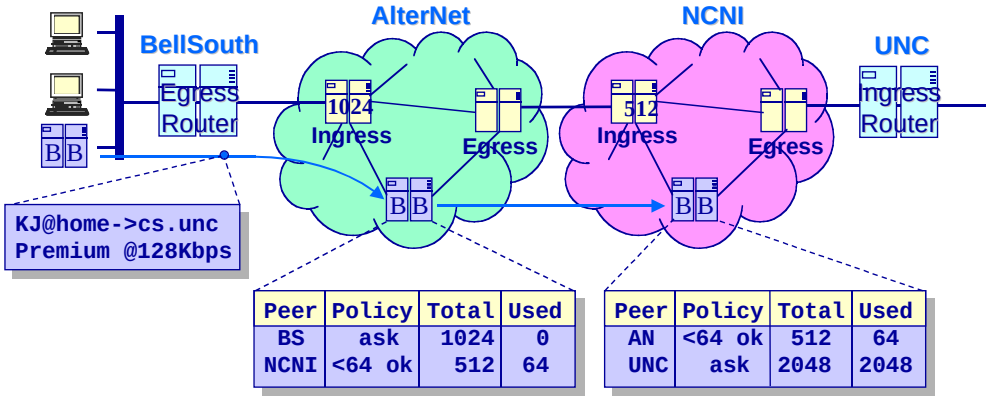
Bandwidth brokers



- "Bandwidth brokers" allocate premium/assured bandwidth on the campus and control egress router(s)
  - » Assume some signaling protocol exists

Dynamic Bandwidth Allocation

Bandwidth brokers



- Bandwidth brokers can realize dynamic global allocation
  - » But state is limited to a small number of routers and connections

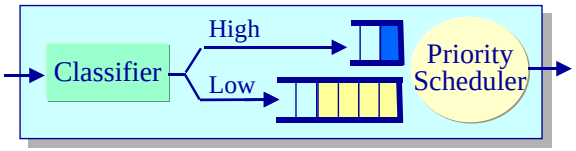
Service Allocation Models for the Internet

Where is this going?

- The IETF is standardizing a set of "router behaviors"
  - » Called "per hop forwarding behaviors" (PHBs)
- Two main PHBs:
  - » Assured forwarding (AF)
  - » Expedited forwarding (EF)
- These are part of a larger framework called the *differentiated services architecture for the Internet* (*diffserv*)

The Differentiated Services Architecture

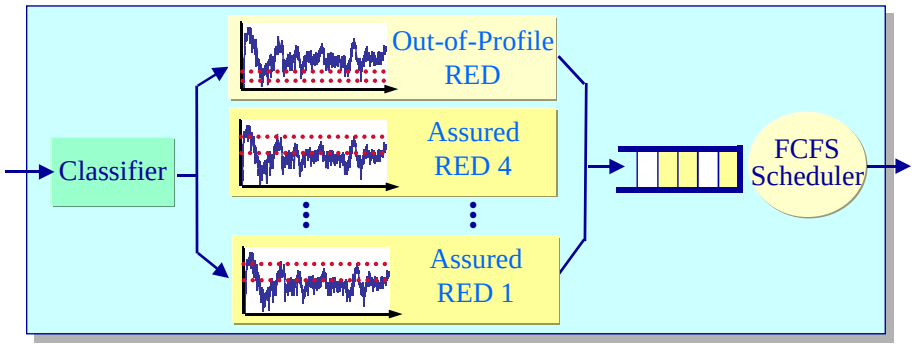
Expedited forwarding (EF)



- EF markings are used to realize a virtual leased-line abstraction
- Semantics:
  - » Maximum arrival rate must be less than the minimum departure rate
  - » Minimum rate should average at least the configured rate when measured over any interval greater than or equal to the time required to send an MTU-sized packet on the output link at the configure rate

The Differentiated Services Architecture

Assured forwarding (AF)



- AF markings will be used to differentiate packets into 4 service classes, each with 3 levels of "drop preference"
  - » Drop preference useful for prioritizing packets within a service class

The Differentiated Services Architecture

When will we see this stuff deployed?

```
(Message inbox:9804)
-------------------------------------------------------------------------
Abilene Premium Service Test Program Launched

April 11th, 2000 - Armonk, NY - To support the QBone, an interdomain
quality of service (QoS) testbed initiative sponsored by Internet2,
Internet2 announced at the recent Spring 2000 Internet2 Member Meeting
the launch of the Abilene Premium Service (APS) test program.
...
The Qbone/Abilene Premium Service aims to provide a low-loss, low-
jitter service to advanced applications. Typically, these are real-
time applications that support either human-to-human collaborations or
human-to-machine remote control, and demand a level of interactivity
that imposes stringent worst-case delay, jitter, and loss requirements
on the underlying network service.
...
The Abilene Premium Service is built on the Expedited Forwarding (EF)
per-hop behavior defined by the IETF Differentiated Services working
group. The basic packet conditioning and forwarding service is
complemented by a measurement infrastructure which will provide
detailed QoS performance data to support end-to-end debugging and
analysis of QoS-enabled paths.
...
```

Service Allocation Models for the Internet

Summary

- Capacity allocation & isolation are required for better-than best effort services
  - » But it need not be on a per-flow basis
- Key principles:
  - » Keep state only at the edges of the network
- Research community is focused on standardizing "forwarding behaviors" rather than "services"
  - » But active queue management and simple priority scheduling are at the heart of proposals for next generation services