Quality of Service

- Traditional Internet gives single class of best-effort service
  - Even though ToS bits were included in the original IP header
- Treats all packets the same
  - All customers
  - All applications
- Should Internet give better quality service to some packets?
  - Why?
  - Why not?

Three Relevant Factors

- Application performance
- Bandwidth required to provide performance
- Complexity/cost of required mechanisms

Providing Better Service

- Routing or Forwarding
- Scheduling or Dropping
- Relative or Absolute

Relative QoS

- Priority scheduling
  - Favored packets get lower delay and lower drop rate
- Priority dropping
  - All sent packets get same average delay
- Why bother with priority dropping?

Differentiated Services (DiffServ)

- Goal: offer different levels of service
  - Organized around domains
  - Edge and core routers
- Edge routers
  - Sort packets into classes (based on variety of factors)
  - Police/shape traffic
  - Set bits (DSCP) in packet header
- Core routers
  - Handle packet (PHB) based on DSCP
**DiffServ Architecture**

**Traffic Policing/Shaping**
- Token bucket \((r,b)\) [everyone know this?]
- Police: if token is available, packet is considered “in”
  - Otherwise considered “out”
- Shape: packet is delayed until token is available

**Implementing Drop Priority**
- RED in/out (RIO)
- Separate dropping curves for in and out traffic
  - Out curve measures all packets
  - In curve measures only in packets

**Sender and Receiver Versions**
- Sender-based version:
  - Sender (or token bucket next to sender) sets in/out bits
  - Routers service with priority
- Receiver-based version: use ECN
  - Put incoming packets through token bucket
  - If packet is “in”, cancel any ECN bits
  - Receiver only told about congestion for “out” packets

**Combining Drop and Delay Priority**
- Delay priority traffic gets high forwarding priority
- Drop priority traffic uses RIO

**Why Does Giving Priority Help?**
- Making service for one class of traffic better means that service for another class of traffic must get worse
- Why does that help?
From Relative to Absolute Service

- Priority mechanisms can only deliver absolute assurances if total load is regulated
- Service Level Agreements (SLAs) specify:
  - Amount user (organization, etc.) can send
  - Level of service delivered to that traffic
- Premium Service (DiffServ) offers low (unspecified) delay and no drops
  - Acceptance of proposed SLAs managed by “Bandwidth Broker”
  - Only over long time scales

Providing Assurances

- SLAs are typically defined without restriction on destination
- Can’t provision network efficiently, but may not matter

Inter-Domain Premium DiffServ

- Achieve end-to-end bandwidth guarantee
- But is this done for all paths?

From DiffServ to IntServ

- Can easily provide some traffic better service than others
  - Making absolute assurances requires controlling load
- DiffServ worst-case provisioning very inefficient
  - Based on aggregate offered load, not for a specific path
- What about fine-grain assurances about QoS?
  - Per-flow, not per traffic class
- Requires admission control for each flow
  - E.g., reservations

Major Philosophical Change

- Per-flow admission control is drastic change to the Internet
  - But best-effort still available (used for most traffic)
- We will first discuss whether this is a good idea
  - Going back to basics about application performance, etc.
- We will then talk about how one might do this
  - Cursory overview, because details are in the dustbin of history

Reservations or Best-Effort

- Basic question:
  - Should we admit all flows (BE), or
  - Refuse some to preserve good service for current flows (R)
- Precedents:
  - The telephone network uses admission control
  - The current Internet does not
- Which one is right? Huge ideological battle!!
- How can we decide?
  - Which provides better application performance?
Modeling Application Performance

- Not a simple function of delay/jitter/loss
- Depends on user perception
  - e.g., picture quality, etc.
- Depends on adaptive application behavior
  - Adjust sending rate
  - Adjust coding (to mask errors)
  - Adjust “playback point” (later)
- For a given application, can describe performance as a function of available bandwidth

Classes of Application

- Traditional data applications: “elastic”
  - Tolerant of delay
  - Tolerant of loss
- Streaming media applications: “real-time”
  - Less tolerant of delay
  - Less tolerant of loss
  - Often of the “playback” variety

Playback Applications

- Video/audio stream being sent
- “Played back” at receiver
- Receiver picks time to play back content
  - “playback point”
- Playback point:
  - Moves: distortion
  - Late: delay
  - Misses packets: “drops”

Classes of Applications

- Elastic:
  - Tolerant of delays and losses
- Real-time:
  - Rigid (can’t tolerate failures/distortion, can’t move playback point)
  - Adaptive (can tolerate failures, can move playback point)
- Characterized by how performance depends on bandwidth

Elastic Applications

[Graph showing utility increasing with increasing bandwidth]

Rigid Real-Time Applications

[Graph showing utility constant with increasing bandwidth]
Adaptive Real-Time Applications

Back to Question

- Reservations versus Best-Effort
- Which is better?

Thought Experiment

- Consider a large set of flows running over the same link
- Two Options:
  - (1) We let all flows use the link (BE)
  - (2) We limit the usage (R)
- Which produces higher utility?

Mathematical Formulation

- Performance (utility) function of bandwidth: \( U(b) \)
- Capacity of link: \( C \)
- Total performance (utility) with \( k \) flows: \( P(k) \)
  \[ P(k) = k \cdot U(\frac{C}{k}) \]
- What value of \( k \) maximizes \( P(k) \):
  - If \( k_{\text{max}} = \infty \), then best-effort is best
  - If \( k_{\text{max}} < \infty \), then reservations are best

Results

- If \( u(b) \) is concave, then \( k_{\text{max}} = \infty \) (why??)
  - Try it with \( U(b) = 1 - \frac{1}{1 + b} \)
  - \( P(k) = k \cdot U(\frac{1 + C/k}{k}) \)
  - For large \( k \), \( P(k) \approx C - C^2/k \)
- If \( u(b) \) is convex around origin, then \( k_{\text{max}} < \infty \) (why??)
  - Try it with \( U(b) = b^2 \)
  - \( P(k) = C^2/k \)

Conclusion

- Elastic: Best-effort best (concave utility)
- Real-time: Reservations best (convex around origin)
  - Rigid: obvious
  - Adaptive: only small region of convexity (we think)
- Internet and phone system each designed appropriately!!
Adaptive Real-Time Applications

What Have We Ignored?

- Three factors to consider:
  - Application performance: done
  - Bandwidth required: held constant
  - Complexity of mechanisms: ignored!

- How do we compensate for the complexity of the admission control mechanism?

Reservations vs Best-Effort

- Let $R(C)$ denote the total utility using reservations
- Let $B(C)$ denote the total utility using best-effort
- For real-time traffic, we know $R(C) > B(C)$
- But, what is the value of $\Delta$ such that $R(C) = B(C + \Delta)$
- We have to weigh the bandwidth advantage of reservations against the complexity advantage of best-effort

Variable Load

- Assume the load (number of flows) varies: prob($k$)
  - Poisson
  - Exponential
  - Algebraic

- As $C$ becomes large:
  - Poisson: $\Delta \rightarrow 0$
  - Exponential: $\Delta \rightarrow 0$ (adaptive) $\Delta \rightarrow \ln C$ (rigid)
  - Poisson: $\Delta \sim C$ with $\Delta/C < 2$
  - Extensions to this model ( retries, hysteresis) remove bound, but don’t change other asymptotics
  - But they do change quantitative results for smaller $C$

Why Shouldn’t This Kill IntServ?

- Analysis assumes bandwidth is plentiful
- If we are stuck with $C$ on the same order as average load, then the performance advantage of IntServ is pronounced
- Is bandwidth plentiful?

The Overprovisioning Debate

- Some claim bandwidth is plentiful everywhere
  - Cheap
  - Or needed for fail-over
- But that’s within core of ISPs
- Bandwidth is scarce:
  - At edge
  - Between providers
- Intserv would help pay for bandwidth in those places
InSerc

• IntServ = Integrated Services Internet
• Goal: support wider variety of services in single architecture
• Effort largely led by PARC, MIT, USC/ISI
  - I'm the only believer left….

Key IntServ Design Decisions

• Reservations are made by endpoints
  - Network is not making guesses about application requirements
• IntServ is multicast-oriented
  - Assumed that large broadcasts would be a driver of both IntServ
  and multicast
  - Reservations made by receivers
• Soft-state: state in routers always refreshed by endpoints
• Service guarantees are end-to-end on a per-flow basis

Integrated Services Internet

• Flow is QoS abstraction
• Each flow has a fixed or stable path
• Routers along the path maintain state for the flow
• State is used to deliver appropriate service

IntServMechanisms

• Reservation protocol: transmits service request to network
  - TSpec: traffic description
  - RSpec: service description
• Admission control: determines whether to accept request
• Packet scheduling: ensures router meets service rqmts
• Routing: pin routes, look for resource-rich routes

IntServ Services

• Kinds of service assurances:
  - Guaranteed (never fails unless major failure)
  - Predictive (will almost never fail)
• Corresponding admission control:
  - Guaranteed: worst-case
  - No guessing about traffic
  - Predictive: measurement-based
  - Gamble on aggregate behavior changing slowly

Integrated Services Example
**Integrated Services Example**

- Allocate resources - perform per-flow admission control

**Integrated Services Example**

- Install per-flow state

**Integrated Services Example**

- Install per flow state

**Integrated Services Example: Data Path**

- Per-flow classification

**Integrated Services Example: Data Path**

- Per-flow buffer management

**Integrated Services Example**

- Per-flow scheduling
RSVP Reservation Protocol

- Performs signaling to set up reservation state for a session
- A session is a simplex data flow sent to a unicast or a multicast address, characterized by
  - \(<\text{IP dest}, \text{protocol number}, \text{port number}>\)
- Multiple senders and receivers can be in same session

The Big Picture

Sender: sends PATH message via the data delivery path
- Set up the path state each router including the address of previous hop

Receiver sends RESV message on the reverse path
- Specifies the reservation style, QoS desired (RSpec)
- Set up the reservation state at each router

Things to notice
- Receiver initiated reservation
- Decouple routing from reservation

Route Pinning

Problem: asymmetric routes
- You may reserve resources on R->S3->S5->S4->S1->S, but data travels on S->S1->S2->S3->R!
- Solution: use PATH to remember direct path from S to R, i.e., perform route pinning
### PATH and RESV messages

- **PATH** also specifies
  - Source traffic characteristics
    - Use token bucket
- **RESV** specifies
  - Service requirements
  - Source traffic characteristics (from PATH)
  - Filter specification, i.e., what senders can use reservation
  - Based on these routers perform reservation

### Reservation Style

- **Motivation**: achieve more efficient resource
- **Observation**: in a video conferencing when there are M senders, only a few are active simultaneously
  - Multiple senders can share the same reservation
- **Various reservation styles** specify different rules for sharing among senders
- **Key distinction**:
  - Reserved resources (bandwidth)
  - Which packets use those resources

### Reservation Styles: Filters

- **Wildcard filter**: all session packets share resources
  - Good for small number of simultaneously active senders
- **Fixed filter**: no sharing among senders, sender explicitly identified in reservation
  - Sources cannot be modified over time
  - Allows reserved resources to be targeted to particular paths
- **Dynamic filter**: resource shared by senders that are (explicitly) specified
  - Sources can be modified over time
  - Switching between speakers at a conference

### What Did We Miss?

- **Make aggregation central to design**
  - In core, don’t want to keep track of each flow
  - Don’t want to process each RESV message
- **Economics**: user/provider and provider/provider
  - We talked about it (at great length) but didn’t realize how inflexible the providers would be
- **Too complicated**: filter styles a waste of time
- **Multicast focus?**