Traffic Management and QoS

Outline

TQ.1  Traffic management functions and services
TQ.2  Traffic characteristics and service models
TQ.3  Basic queueing analysis
TQ.4  Congestion control and avoidance
TQ.5  Packet scheduling disciplines
TQ.6  QoS and Internet service models
    Q.6.1  Guaranteed service and ATM
    Q.6.2  IntServ and RSVP
    Q.6.3  DiffServ
    Q.6.4  MPLS traffic engineering
Traffic Management and QoS

TQ.1  Functions and Services

TQ.1  Traffic management functions and services
TQ.2  Traffic characteristics and service models
TQ.3  Basic queueing analysis
TQ.4  Congestion control and avoidance
TQ.5  Packet scheduling disciplines
TQ.6  QoS and Internet service models
Network Layer Control
Hybrid Layer/Plane Cube

Layer 3: traffic management in control plane
significant interaction with management plane

Management plane

Data plane

Control plane

L8 L7 L5 L4 L3 L2.5 L2 L1.5 L1

social application session transport network virtual link link MAC physical
Network Layer
Service and Interfaces

• Network layer 3 is above link layer 2
  – *addressing*: network layer identifier for end systems (hosts)
  – *forwarding*: transfers packets hop-by-hop
    • using link layer services
    • network layer responsible for determining *which* next hop
  – *routing*: determination of path to forward packets
  – *signalling*: messages to control network layer behaviour
  – *traffic management*: actions to manage E2E service

• Network layer service to transport layer (L4)
  – deliver TPDUs to destination transport entity
Switches
Functions: Routing and Signalling

- Traffic management
Network Layer
Service Model and Traffic Management

- **Service model** describes
  - service the network provides for end-to-end data transfer

- **Traffic management**
  - actions the network takes to manage this service

- **Conflicting goals**
  - sufficient resources to deliver required service to users
  - minimise network resource use to keep costs low

- **Components**
  - congestion control and avoidance
  - resource provisioning and (perhaps) QoS
Traffic Management and QoS

TQ.2 Traffic Characteristics & Service Models

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines
TQ.5 QoS and Internet service models
Traffic Characteristics

Service Models

• Networks provide a service model to applications
• Best effort
  – no service guarantees to application
• Probabilistic guarantees
  – statistical guarantees of performance parameters
• Absolute guarantees
  – guarantees of performance parameters
Traffic Characteristics
Service Models: Best Effort

• Best effort
  – no service guarantees to application
  – only “best effort” attempt to eventually deliver packets

• Resource allocation
  – over provisioning to provide reasonable service
  – no resource allocation to individual flows
  – may schedule among flows for fairness
Traffic Characteristics

Service Models: Probabilistic Guarantees

• Probabilistic guarantees
  – statistical guarantees of performance parameters

• Resource allocation
  – allocation of resources to meet probabilistic service bounds
  – assumes benefits from statistical multiplexing
  – average rate + burstiness allocation
  – degree of over provisioning helps meet service guarantees
Traffic Characteristics
Service Models: Absolute Guarantees

• Absolute guarantees
  – hard guarantees of performance parameters

• Resource allocation
  – allocation of resources to meet absolute service bounds
  – worst case allocation of resources
    • peak rate
  – circuit emulation
Traffic Characteristics

Traffic Parameters

- Throughput or rate
- Delay
- Jitter: delay variance
- Reliability
- Delivery order
Traffic Characteristics

Throughput or Rate

- **Main parameters**
  - peak rate $r_p$
  - average rate $r_a$
  - burstiness (peak to average ratio or max burst size)
Traffic Characteristics

Delay

- Delay [s] is the sum of components:
  - speed-of-light path delay
    - through transmission media
    - through circuits in nodes
  - object transmission delay
  - queueing delay in network nodes
    - due to blocking in a switch fabric (try to avoid)
    - due to contention for an output link
Traffic Characteristics

Jitter

- Jitter: delay variance around mean delay
  - $d_{\text{min}}$ is the minimum path delay
  - additional delay due to variable queueing and processing
Traffic Characteristics

Reliability

- Reliability
  - packet loss rate
  - maximum burst error size
Traffic Characteristics

Delivery Order

• Delivery order
  – whether packets must be delivered in order
  – may be required by E2E protocol or application
Traffic Characteristics

Traffic Classes

• The *traffic class* specifies the characteristics of a flow
  – constant bit rate
    • specified as a single rate
  – variable bit rate
    • specified as (peak rate, average rate, burstiness)
  – best effort
    • a desired minimum rate may be specified

• Mechanisms to support traffic classes
  – over-provisioning
  – QoS mechanisms
Traffic Characteristics

Traffic Matrix

- Traffic can be characterised by a traffic matrix
- matrix that describes traffic from every input → output pair
- element $i, j$ is traffic parameter from $i$ to $j$
  - engineered capacity [b/s or Erlangs]
  - load [b/s or Erlangs] typically over some time interval
  - delay
  - loss rate
## Traffic Characteristics

### Traffic Matrix

- **Example traffic matrix:** load [Mb/s]
  - traffic load from $i \rightarrow j$ and $j \rightarrow i$ (typically asymmetric)
  - requires measurements by service provider

- **e.g.**
  - $r_{\text{KCK}\rightarrow\text{ZÜR}} = 3.4$
  - $r_{\text{ZÜR}\rightarrow\text{KCK}} = 6.1$

<table>
<thead>
<tr>
<th></th>
<th>BOS</th>
<th>KCK</th>
<th>LAN</th>
<th>MKE</th>
<th>SAN</th>
<th>STL</th>
<th>ZÜR</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td>–</td>
<td>7.1</td>
<td>1.4</td>
<td>6.4</td>
<td>4.9</td>
<td>4.4</td>
<td>9.6</td>
</tr>
<tr>
<td>KCK</td>
<td>0.2</td>
<td>–</td>
<td>5.7</td>
<td>9.2</td>
<td>2.7</td>
<td>1.7</td>
<td>3.4</td>
</tr>
<tr>
<td>LAN</td>
<td>9.3</td>
<td>4.2</td>
<td>–</td>
<td>0.2</td>
<td>1.5</td>
<td>8.0</td>
<td>6.6</td>
</tr>
<tr>
<td>MKE</td>
<td>6.1</td>
<td>8.4</td>
<td>4.3</td>
<td>–</td>
<td>8.7</td>
<td>3.3</td>
<td>8.3</td>
</tr>
<tr>
<td>SAN</td>
<td>2.2</td>
<td>5.4</td>
<td>9.8</td>
<td>4.3</td>
<td>–</td>
<td>5.2</td>
<td>3.8</td>
</tr>
<tr>
<td>STL</td>
<td>3.5</td>
<td>9.3</td>
<td>1.4</td>
<td>7.3</td>
<td>8.1</td>
<td>–</td>
<td>1.1</td>
</tr>
<tr>
<td>ZÜR</td>
<td>2.9</td>
<td>6.1</td>
<td>4.6</td>
<td>2.7</td>
<td>0.1</td>
<td>4.6</td>
<td>–</td>
</tr>
</tbody>
</table>
Traffic Management and QoS

TQ.3 Basic Queueing Analysis

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines
TQ.6 QoS and Internet service models
Queueing Analysis
Basic Model

- Packets arrive in a queue at rate $\lambda$ [1/s]
- Packets are served at a rate $\mu$ [1/s]
  - service time of $s$ [s]
- System utilisation is $\rho = \lambda / \mu$ [Erlangs] ([1] = [s/s])
  - expressed as number in (0...1) or %
Queueing Analysis

Little’s Theorem

- \( N = \lambda T \)
- **Variables**
  - \( N \) – average number of customers in system
  - \( \lambda \) – arrival rate [1/s]
  - \( T \) – average time spent in system [s]
- **Holds for *all* distributions**
- **Intuition**
  - as service time increases, so do number in system
  - as arrival rate increases, so do number in system
Queueing Analysis
Distributions

• Arrival and service rates determine performance
  - average rates matter, but...
• Distribution of rates are critical
  - uniform: all interarrival or service times equal
    • simple but unrealistic for most systems
  - exponential or Poisson
    • \( \Pr[k \text{ arrivals in interval } T] = \frac{(\lambda T)^k}{k!} e^{-\lambda T} \)
  - bursty: arrivals clumped into groups
  - self-similar: distribution similar at different granularities
    • evidence of self-similar traffic in Internet
    • challenges benefits of statistical multiplexing
Queueing Analysis

Notation

• A/S/N
  – A and S: distribution of arrivals and service times
    G – general independent
    M – exponential (Poisson)
    D – deterministic or uniform
  – N: number of servers

• Examples
  – M/M/1: exponential arrival, exponential service, single server
M/M/1 Queueing Analysis

Basic Results

• Utilization $\rho = \frac{\lambda}{\mu}$
  - arrival rate / service rate
• Average time in queue
  - note what happens as $\rho \rightarrow 1$
• Average time in system

$$W = \frac{\rho/\mu}{1-\rho}$$

$$T = \frac{1/\mu}{1-\rho}$$
M/M/1 Queueing Analysis

Discussion

• M/M/1 equations very simple
  – so they are frequently used in closed-form analysis
  – reflects independent arrivals
    • e.g. Web transactions
  – does not accurately represent traffic in network
    • bursty traffic
    • correlated traffic

• G/G/1 equations very complex!
  – requires integration of a function
  – difficult to analyse networks of generalised queues
Analysis Techniques

Analytical

- Analytical queueing analysis
  - mathematical analysis of system
  - useful for small simple system
  - *exact* behaviour of *complex* system difficult or impossible
    - not possible to precisely analyse large network of real traffic
  - critical problem:
    - simplifying assumptions must be made for complex systems
    - usefulness of results depends on these assumptions!

- Simulation
- Emulation
- Measurement
Analysis Techniques

Simulation

• Queueing analysis
• Simulation
  – build software *simulation model* of system
    • generally more complex models possible than analytical
  – critical problem:
    • real system must be abstracted to software model
    • abstractions usually contain simplifying assumptions
    • usefulness of results depends on abstraction and model!

• Emulation
• Measurement
Analysis Techniques

Emulation

- Queueing analysis
- Simulation
- Emulation
  - hardware component designed to behave like real system
    - e.g. link emulator
  - allows control of behaviour not possible in real system
  - critical problem
    - emulator usually simplification of real system
    - usefulness of results depends on simplification!
  - emulator frequently a component of simulation or testbed
- Measurement
Analysis Techniques

Measurement

• Queueing analysis
• Simulation
• Emulation
• Measurement
  – measurement of actual system
    • may be a testbed – usually a small replica of real network
  – critical problem:
    • testbed may not reflect scale of real network
    • access to measurement data difficult
      – due to scale of real network
      – due to administrative concerns, including privacy
Analysis Techniques

Verification

- Combination of techniques
  - each technique has strengths and weaknesses
  - should be combined as necessary

- Verification of models
  - compare results of two techniques to verify model
  - example:
    - compare queueing and simulation model of simple system
    - gives confidence of simulation model for more complex system
Analysis Techniques

Design Cycle

• Use each technique as appropriate in design cycle
• Design cycle
  – mathematical analysis and/or simulation
    • iterate on design parameters until desired behaviour
    • gives insight on how to build *prototype* system
  – construct prototype and measure in testbed
    • analysis in small scale
    • iterate design; modify simulation model if necessary
  – build real system on large scale
    • measure to confirm design and modify future deployments
Traffic Management and QoS

TQ.4 Congestion Control and Avoidance

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines
TQ.6 QoS and Internet service models
Congestion Control

Definitions

- **Flow control**
  - control transmission to avoid overwhelming *receiver*
  - this is an exclusively a transport layer function *lecture TL*

- **Congestion control**
  - control transmission to avoid overwhelming *network* paths
  - this is a network layer function
    - that may interact with the transport or application layers
Congestion Control

Ideal Network

- Ideal network:
  - throughput: carried load = offered load (45° line)
  - zero delay (flat line)

*Why isn’t this possible?*
Congestion Control
Real Network

• Delay can’t be zero: speed-of-light
  – delay through network channels
  – delay along paths in network nodes

• Throughput can’t be infinite
  – channel capacity limits bandwidth
  – switching rate of components limits bit rate
Congestion Control

Desired Real Network Behaviour

- Desired network:
  - carried load = offered load up to share of capacity

*what happens to delay?*
Congestion Control

Causes of Congestion

- Congestion when offered load $\rightarrow$ link capacity
- Best effort
  - occurs whenever load is high
- Probabilistic guarantees
  - occurs when load exceeds resource reservations
- Absolute guarantees
  - can’t happen (in theory)
Congestion Control
Infinite Buffers / No Retransmission

• Congestion when offered load → link capacity
• Infinite buffers
  – queue length builds to infinity
  – $d \rightarrow \infty$
Congestion Control
Finite Buffers with Retransmission

- Finite buffers cause packet loss
  - retransmissions contribute to offered load
    - assuming reliable protocol
  - throughput curve levels off
Congestion Control
Multihop Network

• Multiple hops
  – downstream packet loss
  – upstream transmission wasted
Congestion Control

Consequences of Congestion

• Delay *increases*
  – due to packet queuing in network nodes
  – due to retransmissions when packets overflow buffers
    • finite buffers must drop packets when full

• Throughput
  – levels off gradually (with real traffic)
  – then *decreases*
    • due to retransmissions when packets dropped
    • particularly over multiple hops
  – *congestion collapse*
    • “cliff” of the throughput curve
Congestion control *reacts* to congestion
- operates at the cliff of the curve to prevent collapse

*What should congestion control do?*
Congestion Control

Local Action

- Local action at point of congestion
- Drops packets: discard policy
  - tail drop simplest: drop packets that overflow buffers
  - more intelligent policies possible
    - need flow or connection state in switches
    - discriminate *which* flows are causing congestion and penalise
Congestion Control
End-to-End Action

- End-to-end action: throttle source
  - reduce transmission rates
  - prevent unnecessary retransmissions
- Explicit congestion control
  - message to signal source to throttle
- Implicit
  - sender assumes that loss is due to congestion
  - source throttles with tail drop of a packet

*More later*
Congestion Control
Preventing Congestion Collapse

• Congestion control reacts to congestion occurring
  – may be too late to avoid large-scale congestion collapse

*Can we do better?*

*consider how a node could detect congestion*
Congestion Control

Congestion Avoidance

- Congestion control reacts to congestion occurring
  - may be too late to avoid large-scale congestion collapse
- Congestion *avoidance* attempts to *prevent*
  - measure impending congestion
  - detection of increasing queue occupancy
- Important with high bandwidth-×-delay product
  - response to an event happens after more bits transferred
  - it may be too late to react
    - all bits causing problem may already be transmitted
    - other cause of problem may have gone away
• Congestion avoidance *predicts* congestion
  – operates at the knee of the curve to prevent collapse

*What should congestion avoidance do?*
Congestion Avoidance

Goals

• Congestion avoidance
  – prevent congestion from occurring

• Keep delay low
  – buffer occupancy → 0 in steady state
  – buffering needed for
    • transient conditions
    • traffic bursts
Congestion Avoidance
Mechanisms

- Queue threshold detects *impending* congestion
  - less than buffer size (before tail drop)
- Explicit congestion avoidance
  - signal source to throttle or adjust rate
- Implicit congestion avoidance
  - AQM: active queue management
  - e.g. RED (random early detection)
- Fragment discard should cause PDU discard
  - PPD/EPD (partial/early packet discard) in ATM network
  - IP fragment discard possible, but better avoid fragmentation
Congestion Avoidance & Control

Implicit

- Sender assume loss is congestion and throttles
  - assumes that *all* losses are due to congestion
- End-to-end error and congestion control combined
  - missing ACK assumed to be congestion
    - causes throttling and then retransmission
  - TCP does this *Lecture TL*
  - reasonable when all losses *are* due to congestion
    - fiber optic channels connected by reliable switches
  - performs poorly when significant losses in channel
    - mobile wireless links  [Krishnan 2004]
    - under-provisioned CDMA (including optical CDMA)
Closed-loop congestion control
  - feedback from network necessary if no hard reservations
Implicit Congestion Avoidance

AQM: Random Early Detection

- RED (random early detection) [Floyd 1993]
  - drop packets

- Compute average queue length over an interval
  - EWMA: exponentially weighted moving average
    \[ \tilde{q} = (1 - \alpha)\bar{q} + \alpha q \]
    - \( q \) = instantaneous queue length
    - \( \alpha \) = weighting factor

- Drop packets if threshold exceeded
  - if \( \bar{q} < \theta_{\text{min}} \) do nothing
  - if \( \theta_{\text{min}} \leq \bar{q} \leq \theta_{\text{max}} \) drop with probability proportional to \( \bar{q} \)
  - if \( \bar{q} > \theta_{\text{max}} \) drop every packet
Implicit Congestion Avoidance
AQM: Random Early Detection

- Drop packets if threshold exceeded
  - if $q < \theta_{min}$ do nothing
  - if $\theta_{min} \leq q \leq \theta_{max}$ drop with probability proportional to $\frac{q}{\theta_{max}}$
  - if $q > \theta_{max}$ drop every packet
Implicit Congestion Avoidance
AQM: Random Early Detection

- RED advantages
  - causes TCP sources to throttle to relieve congestion

- RED disadvantages
  - random drops don’t necessarily penalise offending source
  - UDP and non-TCP friendly transport ignores RED drops
    - at the cost of TCP flows

- RED deployment
  - AQM recommended by [RFC 2309]
    - RED default recommendation
  - not possible to directly measure deployment E2E
Congestion Control & Avoidance

Explicit

- Congested node signals sender to throttle
  - ECN: *explicit congestion notification*

- Explicit signalling message
  - BECN or FECN (backward or forward ECN)
  - example: BECN/FECN in ATM networks
  - example: ICMP source quench
  - requires ability to address signalling message to source

- Congestion marking of headers in packets
  - example: ECN in IP networks
FECN Performance

- Forward notification requires full RTT
  - may necessary if no signalling back-channel

\[ C = 1 \]
Congestion Control & Avoidance

BECN Performance

- Backward signaling reduces control loop delay
  - half on average
  - important in high bandwidth-delay networks (*why?*)
Congestion Control & Avoidance

Internet ECN

- Explicit congestion notification in the Internet
- BECN (backward ECN)
  - ICMP source quench (never deployed)
- FECN (forward ECN)
  - ECN (explicit congestion notification)
ICMP Source Quench
Overview

- **ICMP Source quench**
  - type = 4
  - code = 0
  - payload is congesting datagram + 64B of its payload

- **Congested router**
  - signals source
  - TCP throttles

- **Never widely deployed**
  - in spite of [RFC 1122]
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
  - ECT (ECN capable transport) bit in IP header
- IP router marks packet when congestion impending
  - CE (congestion experienced) bit in IP header
  - FECN since packets marked; receiver must reply to source
- Receiver turns around congestion notification
  - ECE (ECN echo) flag in TCP header for ACKs until...
- Sender acknowledges ECE received
  - CWR (congestion window reduced) bit in TCP header
Internet ECN
IP Packet Fields

- **ECN fields**
  - replaces part of IPv4 TOS
  - \( b_6 - b_7 \) – DSCP
  - \( b_6 \): ECT – ECN capable transport
  - \( b_7 \): CE – congestion experienced
Internet ECN
TCP Segment Flags

- **Control flags (1 bit each)**
  - CWR: cong. win. reduced
  - ECE: ECN echo
  - URG: urgent data
  - ACK: acknowledgement
  - PSH: push data
  - RST: reset connection
  - SYN: connection setup
  - FIN: connection teardown

- **32 bits**
  - source port #
  - dest port #
  - sequence number
  - ack number
  - receive window
  - checksum
  - urg data pointer
  - options (variable length)
  - application payload (variable length)
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
  - ECT (ECN capable transport) bit in IP header set to 1
- IP router marks packet when congestion impending
- Receiver turns around congestion notification
- Sender acknowledges ECE received
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router passes packet unmarked if no congestion
  - CE (congestion experienced) bit in IP header left at 0
- Receiver turns around congestion notification
- Sender acknowledges ECE received
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router marks packet when congestion impending
  - CE (congestion experienced) bit in IP header set to 1
- Receiver turns around congestion notification
- Sender acknowledges ECE received
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router marks packet when congestion impending
- Receiver turns around congestion notification
  - ECE (ECN echo) flag in TCP ACK header set to 1
- Sender acknowledges ECE received
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router marks packet when congestion impending
- Receiver turns around congestion notification
  - ECE flag continues to be set in case ACKs dropped until...
- Sender acknowledges ECE received
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router marks packet when congestion impending
- Receiver turns around congestion notification
- Sender acknowledges ECE received
  - CWR (congestion window reduced) TCP flag set to 1
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router marks packet when congestion impending
- Receiver turns around congestion notification
- Sender acknowledges ECE received
  - note that if CWR packet lost it will be retransmitted
Internet ECN
Signalling Message Sequence

- Capability to support ECN signalled by source
- IP router marks packet when congestion impending
- Receiver turns around congestion notification
- Sender acknowledges ECE received
  - on receipt receiver resets ECE to 0 in subsequent ACKs
Congestion Control & Avoidance

Open-Loop End-to-End Rate Control

- **Initial negotiation**
  - *admission control* limits traffic to available net resources
  - receiver negotiates what it can accept (flow control)

- **Steady state**
  - *policing* enforces traffic contract from transmitter
    - excess traffic may be *marked* and passed if capacity available
Congestion Control & Avoidance
Hybrid End-to-End Rate/Congestion Control

- Open loop rate control for statistical guarantees
- Overbooking of resources requires congestion control
  - dynamic rate adjustments to sender
Traffic Management and QoS
TQ.5 Packet Scheduling Disciplines

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines
TQ.6 QoS and Internet service models
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)

operation?
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

• FIFO (first in first out)
  – send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

• FIFO (first in first out)
  – send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
Packet Scheduling

FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available

What happens when the queue fills?
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
- Discard policy: when full queue
  - tail drop: drop arriving packet
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
- Discard policy: when full queue
  - tail drop: drop arriving packet

*alternatives?*
Packet Scheduling
FIFO Queueing

- FIFO (first in first out)
  - send in order of arrival to queue when link available
- Discard policy: when full queue
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
  - random: drop/remove randomly *why?*
Packet Scheduling

**FIFO Queueing**

- FIFO (first in first out)
  - send in order of arrival to queue when link available
- Discard policy: when full queue
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
  - random: drop/remove randomly: RED congestion avoidance
Packet Scheduling

FIFO Queueing

Advantages and Disadvantages?
Packet Scheduling

FIFO Advantages

- FIFO is simplest discipline
  - very simple software management
  - very simple hardware
    - shift register is a series of flip-flops
Packet Scheduling
FIFO Disadvantages

- FIFO is simplest discipline
  - very simple software management
  - very simple hardware
    - shift register is a series of flip-flops

- All packets treated equally
  implications?
Packet Scheduling

FIFO Disadvantages

• FIFO is simplest discipline
• All packets treated equally
  – flows not necessarily treated *fairly*
    • important for best effort service
  – flows not discriminated by service class
    • e.g. gold vs. silver vs. bronze
  – variable packet size not accounted for
    • small packets wait behind large packets
      (recall message switching)
Packet Scheduling

FIFO Disadvantages

- FIFO is simplest discipline
- All packets treated equally
  - flows not necessarily treated fairly
    - important for best effort service
  - flows not discriminated by service class
    - e.g. gold vs. silver vs. bronze
  - variable packet size not accounted for
    - small packets wait behind large packets
      (recall message switching)

*Alternatives?*
Packet Scheduling
Priority Queueing

- Priority queueing
  - separate traffic classes by priority
  - serve highest priority first
  - provides *relative* service differentiation

*Implementation?*
Packet Scheduling

Priority Queueing

- Priority queueing
  - FIFO queue for each traffic class
- Packet *classifier* sorts into proper queues
  - e.g. source address, port, or QoS field
Packet Scheduling
Priority Queueing

- Priority queueing
  - FIFO queue for each traffic class
- Packet classifier sorts into proper queues
  - e.g. source address, port, or QoS field
Packet Scheduling
Priority Queueing

- Priority queueing
  - FIFO queue for each traffic class
- Packet classifier sorts into proper queues
  - e.g. source address, port, or QoS field
Priority queueing
- FIFO queue for each traffic class

Packet classifier sorts into proper queues
- e.g. source address, port, or QoS field
Packet Scheduling
Priority Queueing

- Priority queueing
  - FIFO queue for each traffic class
- Packet classifier sorts into proper queues
  - e.g. source address, port, or QoS field
• Priority queueing
  – FIFO queue for each traffic class
• Packet classifier sorts into proper queues
  – e.g. source address, port, or QoS field
Packet Scheduling
Priority Queueing

- Priority queueing
  - FIFO queue for each traffic class
  - queues served in priority order
- Packet classifier sorts into proper queues
  - e.g. source address, port, or QoS field
- Higher priority traffic classes receive better service
Packet Scheduling
Priority Queueing

Advantages and Disadvantages?
Packet Scheduling
Priority Queueing

- Priority queuing advantages
  - relatively simple discipline
  - priority queues
    - multiple physical FIFO queues
    - memory and pointer management to partition between queues
Packet Scheduling
Priority Queueing

• Priority queueing advantages
  – relatively simple discipline

• Priority queueing disadvantages
  – provides service differentiation
  – but lower priorities may get no service: starvation
  – no mechanism for proportional service

alternatives?
Packet Scheduling
Round Robin Queueing

• Round robin
  – FIFO queue for each class or flow
  – attempt to provide fair service among best-effort flows

*Implementation?*
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- **Round robin**
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- **Packet classifier** sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling

Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows

- Packet classifier sorts into proper queues by flow id

- Packets are served round-robin
  - cyclically taking turns
Packet Scheduling
Round Robin Queueing

- Round robin
  - FIFO queue for each class or flow
  - attempt to provide fair service among best-effort flows
- Packet *classifier* sorts into proper queues by *flow id*
- Packets are served *round-robin*
  - cyclically taking turns; *skip if empty*
Packet Scheduling
Round Robin Queueing

*Advantages and Disadvantages?*
Packet Scheduling
Round Robin Advantages

- Round robin is simple discipline
  - very simple software management
  - very simple hardware
    - round robin is a counter the points to a particular queue
  - in per flow round-robin senders *are* treated equally
    - in terms of number of packets served
Packet Scheduling
Round Robin Disadvantages

- Round robin is simple discipline
- All classes or flows treated equally

implications?
Packet Scheduling

Round Robin Disadvantages

- Round robin is simple discipline
- All classes or flows treated equally
  - senders not necessarily treated *fairly*
  - variable packet size not accounted for
    - small packets wait behind large packets
      (recall message switching)
  - senders not discriminated by service class
    - e.g. gold vs. silver vs. bronze
Packet Scheduling

Round Robin Disadvantages

- Round robin is simple discipline
- All classes or flows treated equally
  - senders not necessarily treated *fairly*
  - variable packet size not accounted for
    - small packets wait behind large packets
      (recall message switching)
  - senders not discriminated by service class
    - e.g. gold vs. silver vs. bronze

*Alternatives?*
Packet Scheduling
Round Robin Disadvantages

• Round robin is simple discipline
• All classes or flows treated equally
  – senders not necessarily treated *fairly*
  – variable packet size not accounted for
    • small packets wait behind large packets
      (recall message switching)
  – senders not discriminated by service class
    • e.g. gold vs. silver vs. bronze

*Alternatives: combine priority and round robin?*
Packet Scheduling
Weighted Round Robin Queueing

- Weighted round robin
  - FIFO queue for each flow or class (flow aggregates)
- Packet classifier sorts into proper queues
  - by flow id or traffic class
- Packets are served cyclically: round-robin
  weights determine relative service: benefits of priority and RR
Packet Scheduling

Fair Queueing

- Motivation
  - fairness among best effort flows

- GPS: generalised processor sharing
  - ideal fluid model of sharing among flows

- Bit-by-bit round robin
  - would serve a bit at a time from each queue
  - fair regardless of packet sizes
  - not practical to implement directly
Packet Scheduling
Weighted Fair Queueing

- WFQ
  - (weighted fair queueing)
  - FIFO queue flow or flow class
- Simulates *bit-by-bit* round robin
  - schedules packets based on simulated departure time
  - relatively complex to implement
Packet Scheduling

Deficit Round Robin Fair Queueing

- Deficit round robin: simple hardware implementation
  - FIFO queue for each class or flow
  - each flow given a quantum of service

- Packets are served *round-robin*
  - as long as packet can be served within quantum + deficit
  - unused service (with packet waiting) added to deficit
  - provides *long term fairness*
Packet Scheduling
Overengineering vs. Optimality

- Traffic management can be very complex
  - classification in switch input processing
  - scheduling in switch output processing
  - flow or connection management in switches
  - interaction with link-state routing
Packet Scheduling
Overengineering vs. Optimality

- Traffic management can be very complex
  - classification in switch input processing
  - scheduling in switch output processing
  - flow or connection management in switches
  - interaction with link-state routing
- Overengineering can dramatically simplify TM
  - at the cost of extra network resource
    - links
    - line cards
    - switches
Packet Scheduling

Overengineering vs. Optimality

• Traffic management can be very complex
  – classification in switch input processing
  – scheduling in switch output processing
  – flow or connection management in switches
  – interaction with link-state routing

• Overengineering can dramatically simplify TM
  – at the cost of extra network resource
    • links
    • line cards
    • switches

• Goal: balance overengineering vs. optimality
Traffic Management and QoS

TQ.6  QoS and Internet Service Models

TQ.1  Traffic management functions and services
TQ.2  Traffic characteristics and service models
TQ.3  Basic queueing analysis
TQ.4  Congestion control and avoidance
TQ.5  Packet scheduling disciplines
TQ.6  QoS and Internet service models
    Q.6.1  Guaranteed service and ATM
    Q.6.2  IntServ and RSVP
    Q.6.3  DiffServ
    Q.6.4  MPLS traffic engineering
Quality of Service Overview

• **Quality of service** (QoS or QOS)
  – the service model that the network provides to applications
  – along with supporting mechanisms

• Recall service models:
  – best effort
    • no service guarantees to application
  – probabilistic guarantees
    • statistical guarantees of performance parameters
  – absolute guarantees
    • guarantees of performance parameters
Quality of Service
Mechanisms: Best Effort

- Best effort
  - no service guarantees to application
  - no QoS mechanisms necessary (traditional Internet)
Quality of Service
Mechanisms: Best Effort with Fairness

• Best effort
  – no service *guarantees* to application
  – no QoS mechanisms necessary (traditional Internet)

• Fairness
  – desirable to allow *fair sharing* of network
    • scheduling under no congestion
    • discard under impending congestion
  – difficult to discriminate well-behaved and misbehaving flows
    • assuming IPv4 (this is why IPv6 has a flowid flowspec)
  – fair mechanisms substantially more complex to implement
    • but hardware advances in the 2000s make this possible
Internet Service Models

Overview

- **Best effort**
  - traditional service model
  - IPv4 TOS field intended for service differentiation
- **Traditional best effort model is being supplemented**
  - new congestion control mechanisms (e.g. ECN, RED)
  - fair queueing
  - SLAs supported by packet classification and MPLS underlays
- **Integrated services**
  - fine-grained traffic management supported by RSVP
- **Differentiated services**
  - coarse-grained traffic differentiation
Internet Best Effort Architecture
IPv4 Type of Service

- **TOS**: type of service field
  - intended to give hints
  - almost never used
    - except for routing traffic
  - no enforcement mechanisms
- **TOS** fields
  - 3 msb precedence
    - 0–7 = normal–high priority
  - 3 flag bits
    - D: 0/1 = normal/low delay
    - T: 0/1 = normal/high throughput
    - R: 0/1 = normal/high reliability
  - 2 lsb unused

<table>
<thead>
<tr>
<th>TOS</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>frag offset</td>
</tr>
<tr>
<td>TTL</td>
<td>protocol</td>
</tr>
</tbody>
</table>
### Internet Best Effort Architecture

**IPv6 Traffic Class and Flow Label**

- **Traffic class** (8b)
  - differentiates traffic class
  - originally 4b priority
- **Flow label** (20b)
  - identify flows for per flow traffic management
- **TM never well defined**
  - later assumed that DiffServ would define

<table>
<thead>
<tr>
<th>06</th>
<th>class</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>payload length</td>
<td>next hdrl</td>
</tr>
<tr>
<td>source address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>payload (= payload length)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Traffic Management and QoS

TQ.6.1 Guaranteed Service and ATM

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines

TQ.6 QoS and Internet service models
  Q.6.1 Guaranteed service and ATM
  Q.6.2 IntServ and RSVP
  Q.6.3 DiffServ
  Q.6.4 MPLS traffic engineering
Quality of Service
Mechanisms: Guaranteed Service

- Probabilistic or absolute guarantees (strong QoS)
  - guarantees of performance parameters: traffic contract
  - resources reserved to meet traffic contract

- Requires
  - connection or flow establishment
  - admission control:
    - only permit new flows if resources available
  - traffic policing to insure sources adhere to contract

- Conflicting goals:
  - sufficient resources to deliver required QoS to users
  - minimise network resource use to keep costs low
Guaranteed Quality of Service
Connection-Oriented Resource Reservation

- Connections establish state
  - admission control: resources
    - reserved with SETUP
    - committed with CONNECT
  - if resources not available
    - connection REJECTed
  - at end of connection
    - resources must be RELEASEd

- QoS guarantees possible
  - feedback control loops not needed
  - less dependent on congestion control
    - unless resources overbooked
 Guaranteed Quality of Service
 Connection-Oriented Admission Control/Policing

• Initial negotiation
  – traffic contract negotiated with network
  – admission control permits only if resources available

• Steady state
  – policing enforces traffic contract from transmitter
  – traffic shaping in switches ensures traffic remains compliant
Guaranteed Quality of Service
ATM Traffic Management

- ATM (asynchronous transfer mode)
  - connection-oriented network based on fast-packet switching
  - major motivation was to support QoS for multimedia apps
- ATM traffic management [AF-TM-0101, -0149, -0150]
  - complex set of traffic classes and parameters
  - attempted to be too optimal in resource allocation
    - recall overengineering vs. optimality tradeoff
Guaranteed Quality of Service
ATM Traffic Classes

- Large set of traffic classes originally specified
  - CBR: constant bit rate for circuit emulation
  - rt-VBR: real-time variable bit rate for multimedia applications
  - nrt-VBR: non-real-time variable bit rate for bursty data
  - ABT: fast reservation
  - UBR: unspecified bit rate for best effort service

- Later additions to support Internet traffic
  - ABR: available bit rate with closed loop flow control
    - adaptive but relative complex
  - GFR: guaranteed frame rate
    - response to concerns about ABR complexity
# Guaranteed Quality of Service

## ATM Traffic Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Name</th>
<th>Control</th>
<th>Traffic</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBR</td>
<td>constant bit rate</td>
<td>open loop</td>
<td>real time circuit emulation</td>
<td>peak rate</td>
</tr>
<tr>
<td>rt-VBR</td>
<td>real-time variable bit rate</td>
<td>open loop</td>
<td>real-time multimedia</td>
<td>peak and average rate burst, jitter, latency</td>
</tr>
<tr>
<td>nrt-VBR</td>
<td>non-real-time variable bit rate</td>
<td>open loop</td>
<td>data</td>
<td>peak and average rate burst</td>
</tr>
<tr>
<td>ABT</td>
<td>ATM block transfer</td>
<td>fast reservation open loop</td>
<td>data</td>
<td>peak and average rate burst</td>
</tr>
<tr>
<td>ABR</td>
<td>available bit rate</td>
<td>hybrid</td>
<td>data</td>
<td>minimum rate peak rate*</td>
</tr>
<tr>
<td>GFR</td>
<td>guaranteed frame rate</td>
<td>open loop</td>
<td>data</td>
<td>minimum rate peak rate*</td>
</tr>
<tr>
<td>UBR</td>
<td>unspecified bit rate</td>
<td>none</td>
<td>best effort</td>
<td>peak rate*</td>
</tr>
</tbody>
</table>

* specified in SETUP but not guaranteed
Guaranteed Quality of Service
Leaky Bucket Policing and Traffic Shaping

- Leaky bucket to enforce
  - average rate or
  - peak rate
- Simple implementation
  - packet arrival $\Rightarrow$ token placed in bucket
  - bucket drained at target rate
  - if threshold $L$ exceeded
    - packet dropped or
    - may be marked if excess aggregate capacity available
    - burst length may be limited by $I$
Traffic Management and QoS

TQ.6.2 Internet Integrated Services and RSVP

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines
TQ.6 QoS and Internet service models
  Q.6.1 Guaranteed service and ATM
  Q.6.2 IntServ and RSVP
  Q.6.3 DiffServ
  Q.6.4 MPLS traffic engineering
Internet Service Models

IntServ Overview

- **Integrated services** [RFC 1633]
  - fine-grained per-flow traffic management

- **Mechanisms**
  - resource reservations in switches/routers
    - assumes per packet delay is most important quality
    - soft state
  - flow setup using RSVP signalling protocol
  - optional traffic engineering using other mechanisms

- **Per flow end-to-end traffic classes**
  - guaranteed service
  - controlled load
Internet Service Models
Admission Control

• Reservation specified as
  – R-spec: service requirements specification for network
    • e.g. minimum rate
  – T-spec: traffic descriptor of sender
    • e.g. token bucket parameters
    • will be specified by each receiver since RSVP receiver oriented

• Flow will not be admitted if resources not available

• Signalling protocol
  – carries R-spec and T-spec to routers
  – RSVP is signalling protocol for IntServ
Internet Service Models
IntServ Guaranteed Service Class

- **Guaranteed service** [RFC 2212]
  - guarantees rate and delay bound

- **Tspec (sender traffic descriptor)**
  - leaky bucket with parameters \((r, b)\)
    \[ r = \text{rate} \, [\text{B/s}] \]
    \[ b = \text{burst length} \, [\text{Bytes}] \]
  - peak rate \(p\) [B/s]
  - minimum policed unit \(m\) [B]
    - smaller packets will be counted as \(m\) B long
  - maximum packet size \(M\) [B] \((M \leq \text{MTU})\)

- **Rspec (network service requirements)**
Internet Service Models
IntServ Guaranteed Service Class\textsubscript{2}

- Guaranteed service [RFC 2212]
  - guarantees rate and delay bound
- Tspec (sender traffic descriptor)
- Rspec (network service requirements)
  - rate $R$ [B/s]
  - slack $S$ [$\mu$s]
  - per node delay allowed to meet E2E bound
Internet Service Models
IntServ Controlled Load Class

- **Controlled load** [RFC 2211]
  - does *not* provide performance quantitative guarantees
- **Service model**
  - between best effort and guaranteed service
  - similar to lightly loaded network
  - high percentage of transmitted packets will reach receiver
  - delay of most packets will not exceed minimum path delay
- **Admission of controlled load**
  - network has sufficient resources to avoid congestion
Internet Service Models
RSVP Overview

- RSVP Internet signalling protocol [RFC 2205]
- Designed for IntServ signalling [RFC 2210]
- Also adapted for other signalling purposes
  - e.g. RSVP-TE for MPLS path establishment [RFC 3209]
RSVP
Design Goals

• Application diversity
  – different resource requirements
• Heterogeneous receiver bandwidth requirements
  – different bandwidth along paths
RSVP
Design Goals

- Application diversity
- Heterogeneous receiver bandwidth requirements
- Use existing routing protocols
  - adaptation to changes in underlying unicast, multicast route
- Multicast a first class service
  - adaptation to multicast group membership
  - receiver oriented flow resource reservation
RSVP

Design Goals

- Application diversity
- Heterogeneous receiver bandwidth requirements
- Use existing routing protocols
- Multicast a first class service
- Scalability
  - signalling overhead to grow (at worst) linear in # receivers
- Modular design
  - heterogeneous underlying technologies
RSVP

Functionality

• Control plane for signalling
  – no interaction with forwarding data plane

• Specification only of *what* resources needed
  – only a signalling protocol to convey reservation
  – IntServ (or other mechanism) determines *how* reserved
RSVP

Functionality

- Control plane for signalling
- Specification only of what resources needed
- No determination of path
  - RSVP separate from routing protocols
  - key problem: BGP not QoS aware
  - traffic engineering helps
- No multicast group management
  - done by IGMP
RSVP
Signalling Message Format

- **Raw IP datagrams**
  - protocol ID = 46
  - may also be encapsulated in UDP packets

- **Format**
  - **router alert** IP option
    - needed for PATH
    - intermediate processing
  - **common header**
  - followed by variable number of object headers

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>protocol ID</td>
</tr>
<tr>
<td>hl</td>
<td>length</td>
</tr>
<tr>
<td>TOS</td>
<td>flags</td>
</tr>
<tr>
<td>length</td>
<td>header checksum</td>
</tr>
<tr>
<td>fragment id</td>
<td>flag</td>
</tr>
<tr>
<td>TTL</td>
<td>frag offset</td>
</tr>
<tr>
<td>source address</td>
<td>router alert option</td>
</tr>
<tr>
<td>destination address</td>
<td></td>
</tr>
<tr>
<td>send TTL</td>
<td>router alert option</td>
</tr>
<tr>
<td>msg type</td>
<td>reserved</td>
</tr>
<tr>
<td>RSVP checksum</td>
<td>RSVP length</td>
</tr>
<tr>
<td>object length</td>
<td>class #</td>
</tr>
<tr>
<td>object content</td>
<td>c type</td>
</tr>
<tr>
<td></td>
<td>(variable length)</td>
</tr>
</tbody>
</table>

...
Internet Service Models
RSVP Signalling Messages

- Path messages (sender-to-network)
  1. PATH - path establishment message
  3. PATHErr - error response to PATH message
  5. PATHTear - path state teardown (remove) message

- Reservation messages (receiver-to-network)
  2. RESV - reservation message
  4. RESVErr - error response to RESV message
  6. RESVTear - reservation teardown (release) message
  7. RESVConf - reservation confirmation message
Internet Service Models

RSVP Object Classes

- NULL
- SESSION
  - required in all messages
- RSVP_HOP
- TIME_VALUE
- STYLE
- FLOWSPEC
  - desired QoS in RESV msg.
- FILTER_SPEC
- SENDER_TEMPLATE
- SENDER_TSPEC
  - sender TSpec
- ADSPEC
- ERROR_SPEC
- POLICY_DATA
- INTEGRITY
- SCOPE
- RESV_CONF
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- **PATH** from sender
  - establishes path state across group
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- PATH from sender
  - establishes path state across group
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- **PATH** from sender
  - establishes path state across group
- **RESV** from receivers
  - reserves resources for each

![Diagram of RSVP signalling](image)
Internet Service Models
RSVP Signalling Overview

• Designed for IP multicast
  – receiver oriented
• PATH from sender
  – establishes path state across group
• RSVP from receivers
  – reserves resources for each
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- PATH from sender
  - establishes path state across group
- RESV from receivers
  - reserves resources for each
Internet Service Models
RSVP Signalling Overview

• Designed for IP multicast
  – receiver oriented

• PATH from sender
  – establishes path state across group

• RESV from receivers
  – reserves resources for each
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- PATH from sender
  - establishes path state across group
- RESV from receivers
  - reserves resources for each
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- PATH from sender
  - establishes path state across group
- RESV from receivers
  - reserves resources for each
Internet Service Models
RSVP Signalling Overview

- Designed for IP multicast
  - receiver oriented
- PATH from sender
  - establishes path state across group
- RESV from receivers
  - reserves resources for each

Diagram:
- Source (S) to receiver (R1, R2)
- PATH and RESV messages
- Node A and B in the network
Internet Service Models
RSVP PATH Messages

• PATH message communicates:
  – sender information
  – reverse-path-to-sender routing information
  – later upstream forwarding of receiver reservations

• PATH message contents:
  – address: unicast destination, or multicast group
  – flowspec: bandwidth requirements spec
  – filter flag: if yes, record identities of upstream senders
    • to allow packets filtering by source
  – previous hop: upstream router/host ID
  – refresh time: time until this info times out
Internet Service Models
RSVP RESV Messages

• Reservations flow upstream from receiver-to-senders
  – create additional receiver-related state at routers
  – reserve resources

• RESV message contents:
  – desired bandwidth
  – filter type
  – filter spec
Internet Service Models
RSVP Soft State

- State periodically refreshed
  - senders periodically resend PATH to maintain state
  - receivers periodically resend RESV to maintain state
  - new receivers alter reservations with new RESV messages
  - messages have TTL field specifying refresh interval

- Flow state removal
  - explicit teardown: PATHTear and RESVTear
  - removed after timeout
    - prevents accumulation of unused state
Internet Service Models
IntServ and RSVP Discussion

- IntServ and RSVP deployment
  - none in backbones
  - very limited in experimental edge networks
  - needed *everywhere* for end-to-end service

- Concerns
  - scalability of maintaining per flow state [RFC 2208]
    - serious challenge in 1990s
    - but feasible now given advances in hardware technology
  - flexibility
    - only two service classes defined
    - but framework allows more
Traffic Management and QoS

TQ.6.3 Internet DiffServ Architecture

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines

TQ.6 QoS and Internet service models
  Q.6.1 Guaranteed service and ATM
  Q.6.2 IntServ and RSVP
  Q.6.3 DiffServ
  Q.6.4 MPLS traffic engineering
Internet Differentiated Services

DiffServ Overview

- Differentiated services [RFC 2474, 2475, 3260]
- Reaction to IntServ and RSVP
  - concerns about scalability
- Coarse-grained traffic management
  - emphasis on relative performance of aggregate flows
  - static provisioning rather than dynamic resource reservation
  - no per flow signalling like RSVP
- No definition for end-to-end traffic classes
  - rather functional components that allow service provisioning
Internet DiffServ

Functional Placement

- Simple functions in network core
  - PHB: per hop behaviour
  - avoids significant per flow state
- Relatively complex functions at edge
  - flow classification
  - traffic conditioning
Internet DiffServ
Architecture

• Network
  – DS domain: contiguous set of nodes
    • common provisioning policies and PHB definitions
  – DS region: contiguous set of DS domains that offer DiffServ

• Nodes
  – boundary node: node at edge of DS domain
  – interior or core node: nonboundary node within DS domain
Internet DiffServ Architecture

- interior node (PHB PHB)
- boundary node (class/cond)
- premarking (set DSCP)
Internet DiffServ
Service Level Specification

- SLS (service level specification)
  - formerly SLA in DiffServ RFCs [RFC 3260]

- Service agreement between network and customer
  - traffic conditioning specification (TCS)
  - pricing, billing, accounting, auditing
  - reliability, availability, security
  - policy for traffic and routing
Internet DiffServ
Traffic Conditioning Specification

- TCS (traffic conditioning specification)
  - formerly TCA in DiffServ RFCs [RFC 3260]
- Details parameters for traffic profiles and policing
- Examples:
  - traffic profiles, e.g. token bucket parameters for each class
  - performance metrics, e.g. throughput, delay, drop priorities
  - actions for non-conformant packets
  - marking and shaping by service provider
Internet DiffServ
IPv4 DSCP

- **DS**: diff serv field
  - replaces IPv4 TOS
  - 6 msb – DSCP
  - 2 lsb – unallocated (ECN)

- **DSCP**: diff serv code point
  - determines PHB of packet
  - default: 000000
  - class selector: others
  - relative treatment
    - 111X000 > 000000
    - routing precedence
  - > larger DSCP ≥ smaller

<table>
<thead>
<tr>
<th></th>
<th>DSCP</th>
<th>EC</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fragment id</td>
<td>flag</td>
<td>frag offset</td>
</tr>
<tr>
<td>TTL</td>
<td>protocol</td>
<td>header checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td>source address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>destination address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(= hl – 20B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>payload</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(= length – hl – 20B)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Internet DiffServ Classification

- **DiffServ classifier**
  - classifier uses predefined rules to select packets
    - BA: behaviour aggregate (when DSCP already marked)
    - MF: multifield – multidimensional classification
  - marker
    - optional premarking of DSCP
      - at source node based on application requirements
      - at access boundary based on LAN or enterprise network
    - marking of DSCP based on MF classification
Internet DiffServ Conditioning

- Traffic conditioner at region boundary to enforce TCS
  - **meter** measures traffic
  - **remarker** inserts new DSCP for out-of-conformance packets
  - **shaper** delays some packets to conform to TCS
  - **dropper** may drop out-of-conformance packets
Internet DiffServ
DiffServ Per Hop Behaviours

- **PHB**: per hop behaviour
  - forwarding *behaviour* at each node *in* DiffServ domain
    - externally observable and measurable
    - does not specify *mechanism* to be used or *implementation*
  - each PHB has unique id code [RFC 3140]

- **Standard PHBs**
  - expedited forwarding (EF): id 101110
  - assured forwarding (AF): id cccpp0
    - ccc = class ∈ {001, 010, 011, 100}
    - pp = drop precedence ∈ {01, 11, 10}
  - others possible (but no current IETF activity)
Internet DiffServ
Expedited Forwarding PHB

- **EF** (expedited forwarding) PHB [RFC 3246, 3247]
  - forwarding similar to high priority queueing
- Typically used to construct low delay & jitter service
  - requires sufficient *provisioning* of resources for **EF**
  - if **EF** traffic exceeds resources delay bounds will not be met
- Packets served at rate of at least $R$
  - independent of non-**EF** offered load
  - logical link with a minimum guaranteed rate
- Implementation alternatives (recall: not specified)
  - priority queueing with token bucket
  - **WFQ** such that **EF** traffic has large enough weight
Internet DiffServ

Assured Forwarding PHB₁

- AF (assured forwarding) PHB [RFC 2597]
- Intended to allow relative levels of service
  - classes provisioned to partition bandwidth
- Four *forwarding classes*
  - each class has a bandwidth guarantee
    - allocated a minimum amount of buffers and rate
  - packet order preserved within each class
  - each class has three *drop precedences*
    - which packets are dropped during congestion *within* class
    - highest drop precedence dropped first
  - mechanism to share bandwidth above min. not specified
Internet DiffServ
Assured Forwarding PHB$_2$

• Implementation properties
  – attempt to minimise long term congestion
    while allowing short term fluctuation
    • smoothing function to measure congestion level
  – dropping insensitive to short-term traffic characteristics
    discard from flows of = long-term traffic char with = prob.
  – discard rate per flow proportional to its fraction of total
  – gradual discard in response to congestion

• Implementation alternatives (recall: not specified)
  – WFQ to partition bandwidth among AF classes
  – random dropping, e.g. RED
Internet DiffServ

DiffServ Region Behaviours

- **PRB**: per region behaviour [RFC 3086]
  - edge-to-edge service through a DS region (or domain)
    - not the same at end-to-end unless only one region or domain
- **PRB** is combination of:
  - interior node PHB
  - boundary node classification and conditioning
- **Examples**:
  - **BE** (best effort) PRB
    - best effort with provisioning to support SLS
  - **VW** (virtual wire) PRB [draft-ietf-diffserv-pdb-vw]
    - circuit emulation: EF PHB with minimum departure rate
    - conditioning to assure that arrival rate < departure rate
Traffic Management and QoS

TQ.6.4 MPLS Traffic Engineering

TQ.1 Traffic management functions and services
TQ.2 Traffic characteristics and service models
TQ.3 Basic queueing analysis
TQ.4 Congestion control and avoidance
TQ.5 Packet scheduling disciplines
TQ.6 QoS and Internet service models
   Q.6.1 Guaranteed service and ATM
   Q.6.2 IntServ and RSVP
   Q.6.3 DiffServ
   Q.6.4 MPLS traffic engineering
Internet Traffic Engineering
Overview

• Provisioning and managing resources
  – sufficient resources to meet customer SLAs
  – limit cost of network infrastructure
  – determine proper placement of infrastructure to balance

• Problem: IP routing not QoS aware
  – generally destination based with no load balancing
  – although OSPF has capability

• Recall:
  – IntServ/RSVP and DiffServ do not do this
  – contrast: ATM PNNI combines QoS, traffic engineering, routing
Internet Traffic Engineering

Alternative Solutions

- Overlay
  - overlay topology engineered on virtual underlay
    - ATM
    - MPLS: multiprotocol label switching
      - required for interdomain routing (BGP4)

- Link-state weight
  - OSPF link state weight manipulation
  - can’t be done interdomain
Internet Traffic Engineering

MPLS Overview

- **MPLS**: multiprotocol label switching
  - fast packet switching roots
  - label swapping

- **Initial motivation**: label swapping in Internet
  - fast packet forwarding
  - without complexity and incompatibility of IP over ATM
  - but IP CIDR lookup at line speed solved in 1990s

- **Traffic engineering motivation**
  - provide set of engineered paths for underlay
Internet Traffic Engineering
MPLS for Traffic Engineering

• LSP: label switched path
  – generated for
    • traffic engineering
    • load balance

• Label distribution signalling protocols
  – LDP: label distribution protocol [RFC 3036]
    • initial protocol but no traffic engineering capabilities
  – RSVP-TE (RSVP traffic engineering) [RFC 3209]
    • chosen by IETF [RFC 3468]
    • RSVP with new object classes
  – CR-LDP (constraint routing LDP) [RFC 3212]
    • alternative IETF protocol not chosen
Internet Traffic Engineering
MPLS Shim Format

- Label shim
  - encapsulates IP packet
  - switches swap label
  - stacked labels
    - allows net hierarchy (ala ATM VP/VC)
Traffic Management & QoS
Further Reading and Additional References


Traffic Management & QoS

Acknowledgements

Some material in these foils comes from the textbook supplementary materials:

• Kurose & Ross,
  *Computer Networking:*
  *A Top-Down Approach Featuring the Internet*

• Sterbenz & Touch,
  *High-Speed Networking:*
  *A Systematic Approach to High-Bandwidth Low-Latency Communication*
  http://hsn-book.sterbenz.org