

Quality of Service in IP Networks

- Introduction
- Relevance
- IntServ Summary

Purpose



- IP packet switching is asynchronous
 - ◆ Link bandwidths are chosen such that the normal (*average*) load plus a safety margin is supported
 - ◆ But there are always some links where the margin is small (e. g. WAN links)
- QoS mechanisms assure that specific important services do **not degrade under network congestion situations**

Packet Switching Problems



- **QoS is determined by the weakest link** in the chain between sender and receiver
 - ♦ In/out Bandwidth mismatch
 - ♦ Aggregation points
- **Over-provisioning? (\$)**
 - ♦ No *guarantee*
- **QoS mechanisms manage network resources – reliably (...today)**

Services with different requirements



- **Normal user data, assured service with committed burst size**
 - ♦ General user data applications (Web, Mail, ...)
- **Realtime, jitter-sensitive**
 - ♦ Voice/telephony
 - ♦ Audio/Video conferencing, possibly multicast
- **Streaming applications, fixed bandwidth**
 - ♦ Audio streaming, possibly multicast
 - ♦ Video streaming, possibly multicast (e. g. surveillance)
- **Critical data applications, low drop probability**
 - ♦ Transactional data
 - ♦ Cluster computing, sync within data centers
 - ♦ Multicast data distribution (PGM)

Services with different requirements (cont.)

- **Network management**
 - ♦ often connectionless, e. g. SNMP, Syslog; or sessions: SDEE, HTTPS, SSH, ...
- **Network control traffic, reserved bandwidth & low latency**
 - ♦ Routing protocols (!), BPDUs, ...
- **Network security management, dedicated bandwidth**
 - ♦ SDEE, RDEP, Syslog, ...
- **Scavenger, best effort & rate-limited**
 - ♦ Kazaa, E-Mule, Skype
- **DoS-prone services, rate-limiting**
 - ♦ ICMP

Voice versus Data

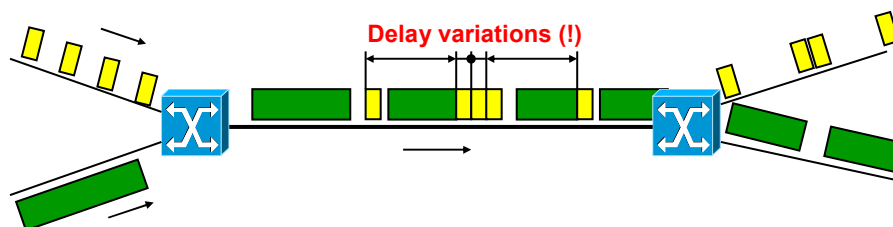
- **Voice is "blocking"**
 - ♦ Call is continuous and "low-rate" (steady 64 kbit/s + Overhead)
 - ♦ Can be modeled via Erlang's equations
- **Data is bursty**
 - ♦ User reads webpage (no traffic)
 - ♦ User clicks on link (600 Kbytes, quickly)
 - ♦ Self-similarity

What we want to control

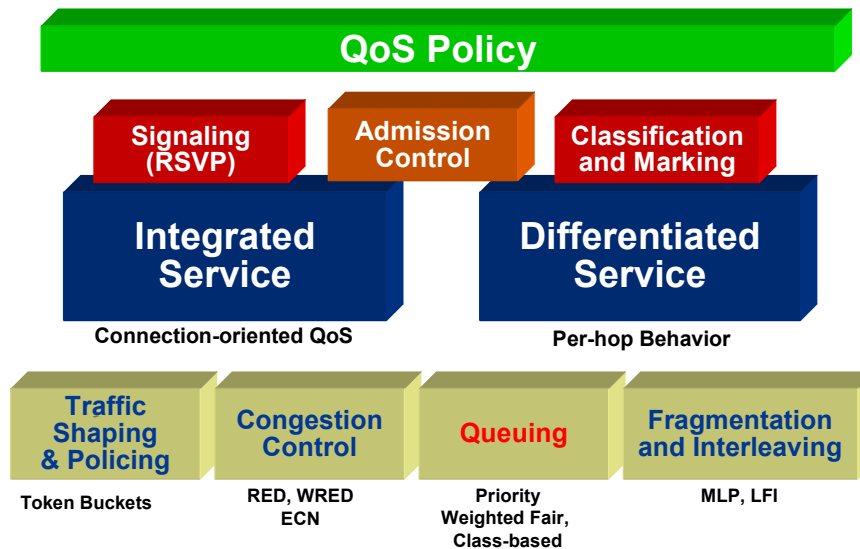


- Delay – one way versus RTT
 - ◆ Also called latency
- Delay variations = 'jitter'
- Packet loss – drop probability
- Average rate and burst size
 - ◆ Bandwidth policing
- Network utilization
 - ◆ Desynchronizing TCP sessions
- Packet reordering
 - ◆ Through badly configured queuing

Jitter through serialization delays



Building Blocks for QoS



QoS Components (1)



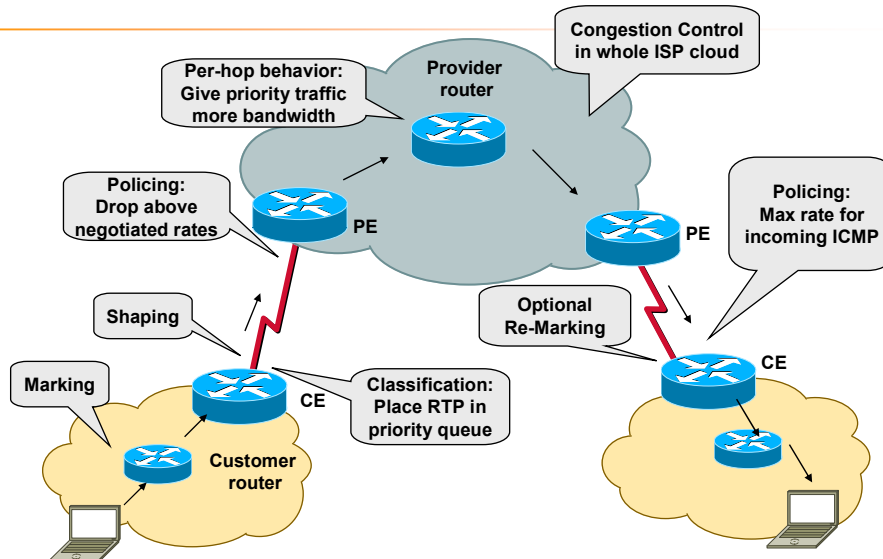
- **Signaling**
 - ♦ E. g. resource **reservation**
- **Queuing**
 - ♦ Prioritize some packets over others and try to remain **fair**
- **Classification**
 - ♦ Detection of specific traffic classes or traffic **flows**
- **Marking**
 - ♦ Assigned a **label** to classified packets to request a special service

QoS Components (2)



- **Integrated and Differentiated Service**
 - ♦ Two fundamental **QoS philosophies**
- **Traffic shaping**
 - ♦ Smoothens bursty traffic by **introducing delay**
- **Congestion control**
 - ♦ Reduces packet rate when network congestion occurs
- **Admission control**
 - ♦ Provides QoS features only to dedicated users
- **QoS policy**
 - ♦ Fundamental QoS agreements specifying detail how to handle traffic, traffic classes, signaling, etc.

Differentiated Services: An Overview



Queuing Practically



- **Queues should be empty on average**
- **A sustained queue depth > 0 typically leads to congestion very soon !**
- **Queues should just smooth bursts and give important packets higher preference**

Growing Queue Depths



- **Short-term variations in arrival rates**
 - ◆ Irrelevant; smoothed by queues
- **Long-term input rate mismatch**
 - ◆ Excess packets are discarded
 - ◆ Drop rate proportional to excess queue
- **Long-term TCP window-delay mismatch**
 - ◆ Window size should be $BW \times RTT$
 - ◆ But TCP can only measure RTT
 - ◆ When delay varies, the network might oscillate between congestion and almost silence

QoS Practically



- QoS for "voice-only" network infrastructure is fairly simple
 - ♦ QoS becomes a "real" task as soon as voice and data shares the same media
- RSVP is rarely used in real world
 - ♦ Creates dynamic queue reservations for VoIP flows (WFQ/LLQ) but scalability issues
- TOS / IP-Precedence is used to mark voice traffic
 - ♦ VoIP traffic uses IP-Precedence "critical" (5)
- Queuing
 - ♦ Low Latency Queuing (LLQ) = PQ-CBWFQ (Priority Q'ing Class Based Weighted Fair Q'ing)
 - ♦ Give real-time traffic the best possible treatment regarding delay and jitter

QoS Practically (cont.)



- Data Fragmentation and Interleaving
 - ♦ On links with BW < 1,5 Mbit/s
 - ♦ Max. data fragment size must have a serialization delay less than 10ms (<dejitter buffer)
 - ♦ Interleave fragmented data between VoIP packets
 - ♦ "Abuse" PPP-Multilink to fragment on serial-links
 - E. g. Voice over IP over HDLC
- Call Admission Control (CAC)
 - ♦ To avoid oversubscription of links
 - ♦ If there is only enough bandwidth for 10 calls, the 11th call would degrade voice quality on all 11 calls
 - ♦ Consider overhead! One G.729 call needs about 13kbit/s (not just 8kbit/s)

IntServ

Integrated Services

(C) Herbert Haas 2009/05/02

RSVP Basics



- Used to reserve queuing resources per flow
 - ♦ Note that a flow is unidirectional (each telephony session consists of two flows)
- The source sends **PATH** messages downstream
 - ♦ To signal the RSVP service to potential receivers
- The receivers send **RESV** messages upstream
 - ♦ To signal desired resource reservation to all routers along the path upstream
- Each of these intermediate routers reserve a **soft state** i. e. it must be refreshed periodically

(C) Herbert Haas 2009/05/02 www.perihel.at

18

Flow Descriptor



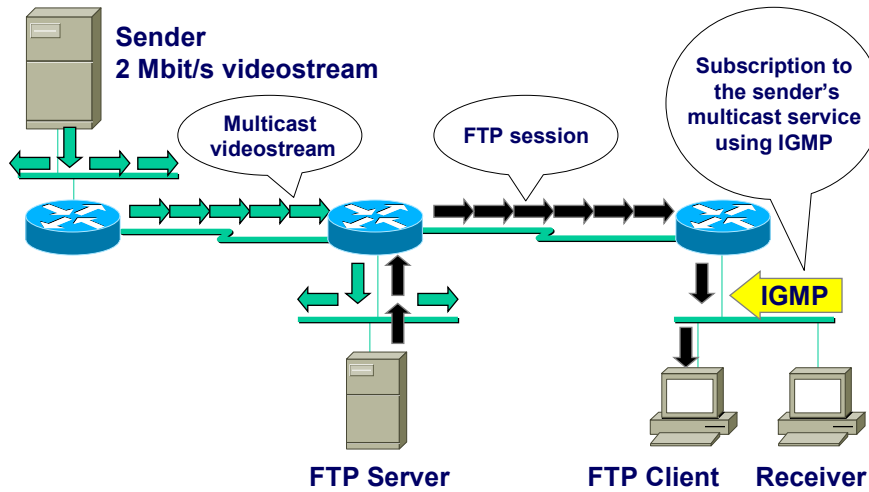
- A Flow Descriptor is carried within RESV messages
 - ♦ Specifies which traffic should receive which resources
- The Flow Descriptor consists of:
 - ♦ **FlowSpec** - used for actual resource reservation
 - Service Class
 - Reservation spec - defines the QoS
 - Traffic spec - describes the data flow
 - ♦ **Fixed Filter** - assigns resource to a single flow
 - Shared Explicit - assigns resource to several flows
 - Wildcard Filter - assigns resources to a general type of flow without specifying the flow

Pros and Cons - Scalability problems

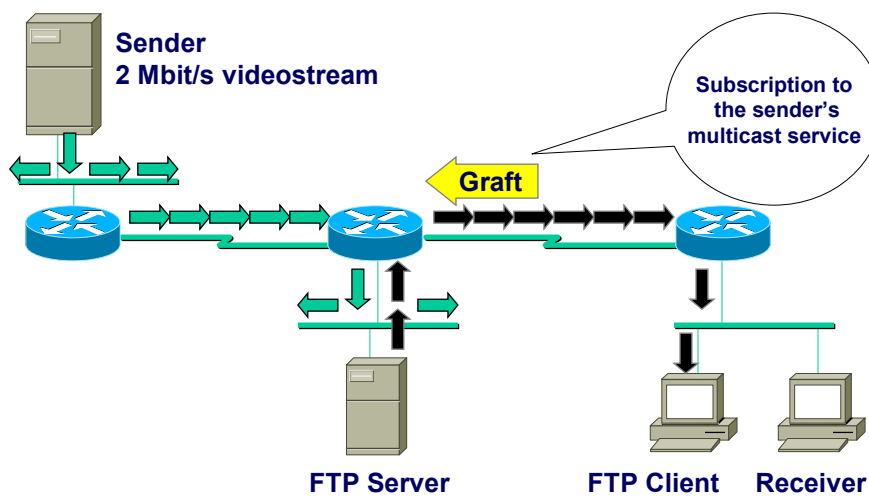


- IntServ follows a stateful and connection-oriented QoS approach –this implies:
 - ♦ + Best traffic control, fine granularity (each flow is serviced independently)
 - ♦ + Routers can deny additional flows when the current queue resources are limited
 - ♦ - All routers must maintain a state for each QoS-related flow
 - ♦ - Missing link problem: If one intermediate router does not support RSVP the whole concept breaks

RSVP Example



Normal Subscription to Multicast Service

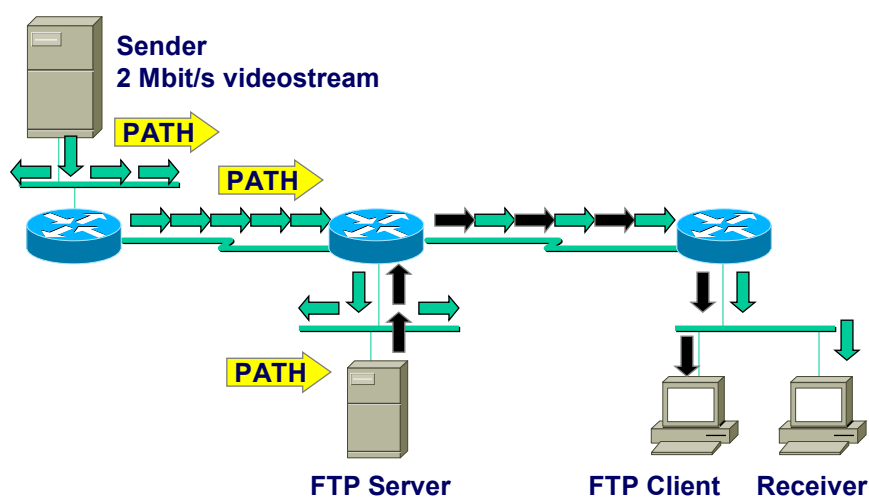


RSVP in Action

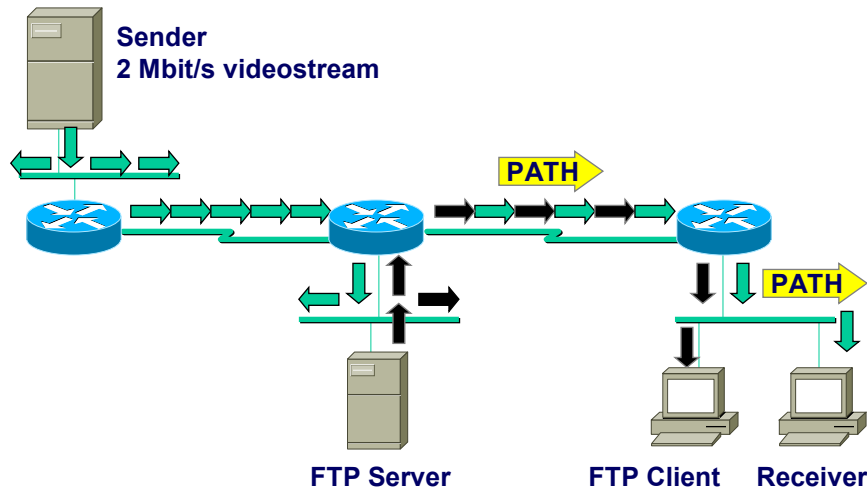


- **Sender sends RSVP PATH messages periodically**
 - ♦ PATH messages include the IP address of the interface through which it is sent
 - ♦ these messages describe the data they intend to send
- **Each RSVP router catches the PATH messages**
 - ♦ it saves the previous hop address, writes its own address as the previous hop, and sends the updated PATH along the same way the application data is using
- **Routers which do not implement RSVP simply forward the PATH message to the next-hop router**
 - ♦ in a network which uses the IntServ model it is not a requirement that all routers implement RSVP

Demand for More Network Resources



RSVP Functionality Announcement via PATH

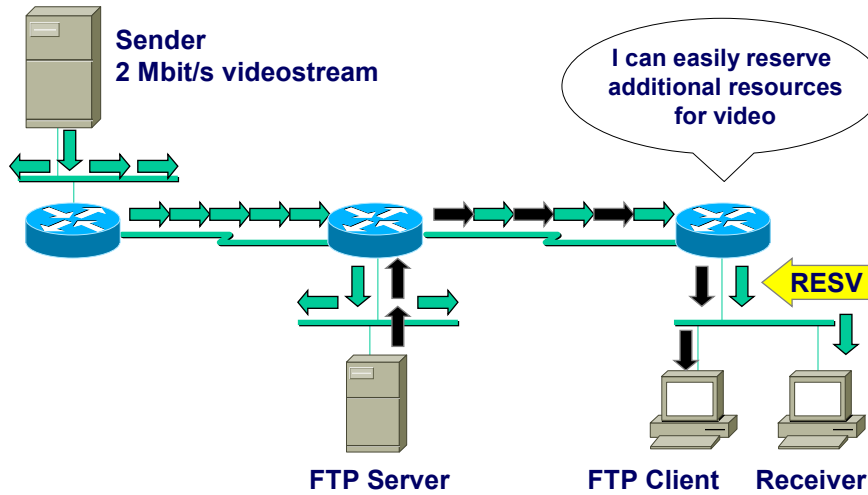


RSVP in Action

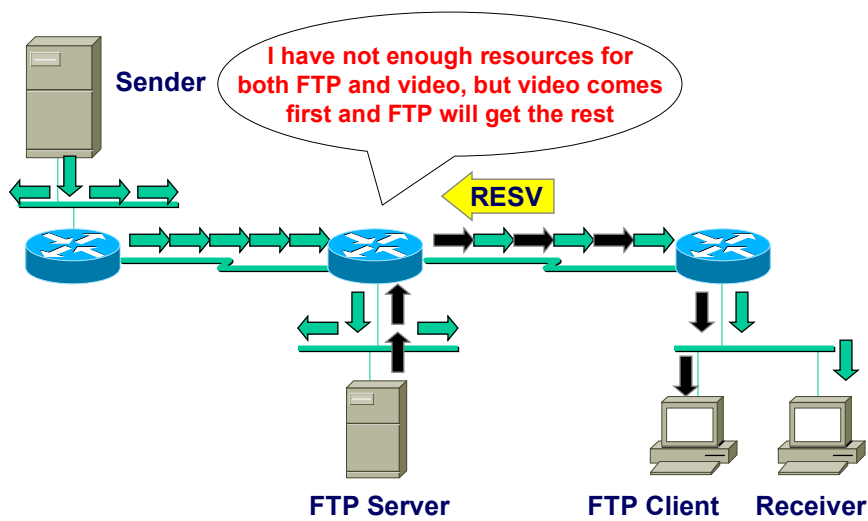


- Receivers identify interesting flows by the TCP/UDP port number
- Receivers initiate a resource reservation by sending periodically a RESV message to the next-hop upstream to the sender
 - ◆ containing a flow descriptor for traffic description and reservation identification
- The RESV messages take the reverse way of PATH messages, means they go from receiver to sender

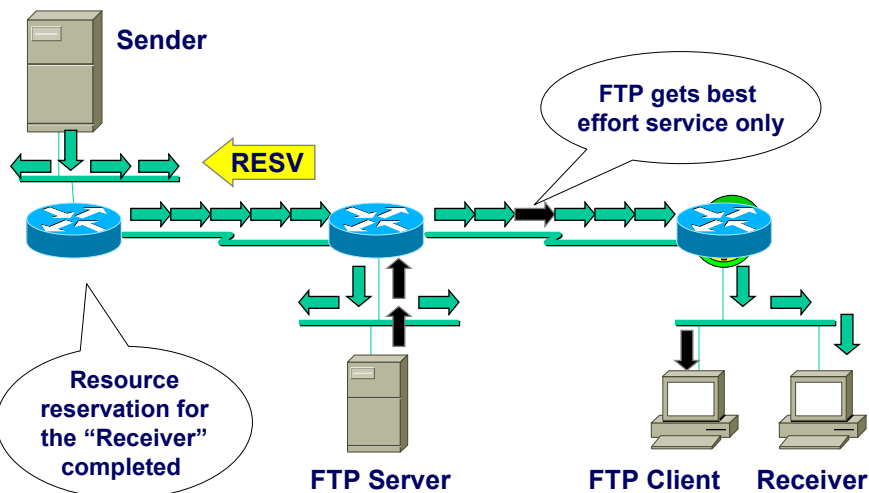
Resource Reservation Upstream 1



Resource Reservation Upstream 2



Resource Reservation Upstream 3



RSVP in Action



- **When a router receives a RESV:**
 - ♦ it passes the request to its admission control function
 - to find out whether there are sufficient resources to implement the reservation request
 - ♦ it passes the request to its policy control function
 - to determine whether policy rules allow the user to make the reservation request
 - ♦ if both checks succeed,
 - it sets parameters in the packet classifier and scheduler functions to implement the requested reservation
 - ♦ it forwards the RESV message to its upstream neighbor

RSVP in Action (cont.)



- **The last router sends a confirmation message back to the receiver (optional RESV CONFIRM)**
 - ♦ **if routers cannot adjust some resources according to RESV, they will refuse the reservations and inform the receiver**
 - **if they can, they merge reservation requests being received and request a reservation from the previous hop router**

RSVP in Action (cont.)



- **RSVP is often used in IP multicast environments**
 - ♦ **several RESV messages for the same sender can be merged together**
 - **for example applications carrying voice or video over IP**
 - ♦ **when the current reservation is equal or greater than that being requested, that RESV message need not be forwarded upstream any longer**
- **Non-RSVP routers within the path are weak links; service degrades to “best effort”**

Removing Reservations



- Reservations can be removed by RSVP TEARDOWN messages:
 - ♦ **Sender initiated:** Using PATH-TEAR to eliminate all downstream path states and associated reservation tables
 - ♦ **Receiver initiated:** Using RESV-TEAR messages to eliminate all upstream reservation states

RSVP Configuration



- By default RSVP sessions are priority-queued which may consume up to 75% of the interface bandwidth
- If LLQ or CBWFQ is configured then RSVP traffic is like any other traffic and a dedicated bandwidth should be configured

```
!!! Enable RSVP and optionally specify the max amount of
!!! bandwidth that may be allocated by RSVP flows (default:
!!! 75% of interface bandwidth for both total and single-flow)
(config-if)# ip rsvp bandwidth [<interface-kbps>]
                               [<single-flow-kbps>]

!!! Turn off resource reservation protocol with keyword
!!! 'none'. Any resource reservation is now done by LLQ and
!!! RSVP could be only used as an admission control mechanism.
(config-if)# ip rsvp resource-provider none | wfq-interface

!!! Disable packet-per-packet classification to improve
!!! performance.
(config-if)# ip rsvp data-packet classification none
```

RSVP as Admission Control Mechanism



- **By disabling the resource reservation feature of RSVP it can still be used as admission control mechanism for DiffServ**
 - ◆ RSVP admits or rejects calls based on a preconfigured amount of bandwidth
 - ◆ But the actual scheduling is only based on LLQ and the DiffServ markings