La qualité de service

C. Pham
Université de Pau et des Pays de l’Adour
Département Informatique
http://www.univ-pau.fr/~cpham
Congduc.Pham@univ-pau.fr
Multimedia, real time on the Internet

Real-time applications
- Interactive applications are sensitive to packet delays (telephone)
- Non-interactive applications can adapt to a wider range of packet delays (audio, video broadcasts)
- Guarantee of maximum delay is useful
Time-constrained applications

- **Elastic applications**
  - Interactive data transfer (e.g. HTTP, FTP)
    - Sensitive to the average delay, not to the distribution tail
  - Bulk data transfer (e.g. mail and news delivery)
    - Delay insensitive
  - Best effort works well

Document is only useful when it is completely received. This means *average* packet delay is important, not *maximum* packet delay.
Discussion

■ **What is the problem?**
  - Different applications have different delay, bandwidth, and jitter needs
  - Some applications are very sensitive to changing network conditions: the packet arrival time distribution is important

■ **Solutions**
  - Make applications adaptive
  - Build more flexibility into network
Why Better-than-Best-Effort (QoS)?

- To support a wider range of applications
  - Real-time, Multimedia, etc

- To develop sustainable economic models and new private networking services
  - Current flat priced models, and best-effort services do not cut it for businesses
What do we have now?

Multimedia applications: network audio and video

Impairments:

- excessive delay: gaps in rendered audio, video
- excessive data loss
Quality of Service: What is it?

Multimedia applications: network audio and video

QoS

network provides application with *level of performance needed for application to function.*
What is QoS?

- “Better performance” as described by a set of parameters or measured by a set of metrics.

- **Generic parameters:**
  - Bandwidth
  - Delay, Delay-jitter
  - Packet loss rate (or loss probability)

- **Transport/Application-specific parameters:**
  - Timeouts
  - Percentage of “important” packets lost
What is QoS (contd) ?

- These parameters can be measured at several granularities:
  - “micro” flow, aggregate flow, population.

- QoS considered “better” if
  - more parameters can be specified
  - QoS can be specified at a fine-granularity.

- QoS spectrum:

  ![Diagram showing QoS spectrum with Best Effort and Leased Line]

  Best Effort  Leased Line

Cours de C. Pham, Univ. Lyon 1
QoS: why don’t we have it?

- QoS a concern since early 1980’s
- Look at what’s happened since 1980:
  - Internet now a million times larger!
  - applications: WWW, Napster, EBay, Gopher
- Why limited progress on QoS?
  - lots of smart people working on it!

Why is the QoS problem so hard?

The Internet is a huge transformative success!
Why was the WWW so “easy”? 

Implemented in hosts, servers at “network edge”
- “on top of” existing network
- “complexity at network edge”
- no changes to network core
Why is QoS more difficult?

- today’s Internet core provides “best effort” service
  - network congestion causes delays, loss
  - no timing guarantees
  - no loss guarantees
- multimedia requires loss, timing constraints met

“The different timing and reliability constraints of real-time communication require new protocols and architectures to be developed”

wet-behind-the-ears researcher, 1982.

New architecture needed for network core!
Where to put QoS?
Améliorer la QoS dans les réseaux IP

- IETF travaille sur des propositions afin de fournir une meilleure QoS dans les réseaux IP, c'est-à-dire aller au-delà du service *best-effort*.

- Les études en cours sont par exemple RSVP, RED, Differentiated Services.

- Modèle simple pour notre étude:
Principes pour garantir la QOS

- Consider a phone application at 1Mbps and an FTP application sharing a 1.5 Mbps link.
  - bursts of FTP can congest the router and cause audio packets to be dropped.
  - want to give priority to audio over FTP

- PRINCIPLE 1: Marking of packets is needed for router to distinguish between different classes; and new router policy to treat packets accordingly
Principles for QOS Guarantees (more)

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

- Alternative to Marking and Policing: allocate a set portion of bandwidth to each application flow; can lead to inefficient use of bandwidth if one of the flows does not use its allocation

- **PRINCIPLE 3**: While providing isolation, it is desirable to use resources as efficiently as possible
Principles for **strong** QOS Guarantees

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

QoS for networked applications

- Packet classification
- Isolation: scheduling and policing
- High resource utilization
- Call admission
Maintenir une qualité de service

- **Eviter les pertes de paquets**
  - dans le réseau:
    • contrôle de l'admission, contrôle de congestion, contrôle de flux,
  - au récepteur:
    • régulation de l'émetteur (fenêtrage), contrôle de flux.

- **Eviter les trop longues latences**
  - notion de priorité
  - ordonnancement intelligents des paquets

---

**FIFO**

**Scheduling Discipline**

Cours de C. Pham, Univ. Lyon 1
The congestion phenomenon

- Too many packets sent to the same interface.
- Difference bandwidth from one network to another

Main consequence: packet losses in routers
The problem of bottlenecks in networks
Causes/coûts de la congestion: scenario 1

- Deux émetteurs, deux récepteurs
- Un routeur, mémoire infinie
- Pas de retransmission

\[\text{delay} \quad \lambda_{\text{in}} \xrightarrow{\lambda_{\text{out}}} \text{router with infinite buffers}\]
**Causes/coûts de la congestion: scenario 2**

- Un routeur, *mémoire finie*
- L’émetteur retransmet les paquets perdus

Diagramme:

- Host A
- Host B
- Router with finite buffers
  - $\lambda_{in}$: original data
  - $\lambda_{in} \rightarrow \lambda_{out}$
  - $\lambda_{in} = \text{original} + \text{retrans}$
Causes/coûts de la congestion: scenario 2

- $\lambda_{\text{in}} = \lambda_{\text{out}}$ (goodput)
- Si la retransmission est parfaite : $\lambda'_{\text{in}} > \lambda_{\text{out}}$
- La retransmission de paquet non perdu rend que dans le cas parfait

"coûts" de la congestion:
- Plus de travail (retrans) pour un même débit utile ("goodput")
- Retransmissions redondantes
Congestion: A Close-up View

- **knee** – point after which
  - throughput increases very slowly
  - delay increases fast

- **cliff** – point after which
  - throughput starts to decrease very fast to zero (congestion collapse)
  - delay approaches infinity

- **Note (in an M/M/1 queue)**
  - delay = $1/(1 - \text{utilization})$
Congestion Control vs. Congestion Avoidance

- **Congestion control goal**
  - stay left of cliff

- **Congestion avoidance goal**
  - stay left of knee

- **Right of cliff:**
  - Congestion collapse

![Graph showing congestion control and avoidance](image)
From the control theory point of view

- Feedback should be frequent, but not too much otherwise there will be oscillations
- Can not control the behavior with a time granularity less than the feedback period
Le contrôle de congestion: principes

- **Réactif**
  - lorsque la congestion est détectée, informer les noeuds en amont et en aval,
  - puis, marquer des paquets, rejeter des paquets, traiter les paquets prioritaires.

- **Préventif**
  - diffusion périodique d'informations d'états (taille des buffers)
  - contrôle continue de la source (Leacky Bucket, Token Bucket...),
  - contrôle de flux, contrôle d'admission.

- **De bout en bout**
  - pas de retour du réseau
  - la congestion est estimée grâce à l'observation des pertes et des délais de bout-en-bout

- **Assisté par le réseau**
  - bit d'annonce de congestion (SNA, DECbit, TCP/ECN, FR, ATM)
Le contrôle de flux, pour le récepteur

**Fenêtrage**
- l'émetteur utilise une fenêtre d'anticipation dans laquelle il va pouvoir envoyer une certaine quantité de données sans acquittements
- la taille de cette fenêtre peut être choisie par le récepteur à la phase de connexion
- si l'émetteur respecte les règles, le récepteur ne sera pas surchargé.

*Cela ne garantit pas que le contrôle de flux sera efficace pour le réseau (voir figure suivante).*
Problème d’un réseau trop faible
Ex: principe du contrôle de congestion dans TCP
- chaque émetteur maintient une deuxième fenêtre de congestion pour le réseau,
- la quantité d'information qu'il est autorisé à transmettre par anticipation est le minimum des 2 fenêtres
- initialement, la fenêtre de congestion est mise à K octets, l'émetteur envoie les données et arme un temporisateur,
- si les données sont acquittées avant l'expiration du temporisateur, on augmente K, et ainsi de suite jusqu'à (i) l'expiration d'un temporisateur ou, (ii) la taille de la fenêtre du récepteur a été atteinte.
- C'est le principe du "slow start"
Slow Start

- La fenêtre de congestion augmente en fait très rapidement!
Le contrôle de congestion dans TCP

- seuil initial à 64K, on augmente K exponentiellement avant et linéairement après (*congestion avoidance*),
- si perte, divise le seuil par 2, et on recommence avec K=1
Utilisation du Round Trip Time

One RTT

0R

1

One pkt time

1R

①

2

3

2R

②

③

4

6

5

7

3R

④

⑤

⑥

⑦

8

10

12

14

9

11

13

15
Slow Start Sequence Plot

La fenêtre de congestion double à chaque aller/retour
TCP Reno (Jacobson 1990)

SS: Slow Start
CA: Congestion Avoidance
Fast retransmission/fast recovery
TCP Vegas (Brakmo & Peterson 1994)

- Converges, no retransmission
- ... provided buffer is large enough
Queuing Disciplines

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

Traffic Sources

<table>
<thead>
<tr>
<th>Traffic Classes</th>
<th>Traffic Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>Class A</td>
</tr>
<tr>
<td>Class B</td>
<td>Class B</td>
</tr>
<tr>
<td>Class C</td>
<td>Class C</td>
</tr>
</tbody>
</table>

Scheduling

Buffer Management

Drop
Typical Internet Queuing

- **FIFO + drop-tail**
  - Simplest choice
  - Used widely in the Internet

- **FIFO (first-in-first-out)**
  - Implies single class of traffic

- **Drop-tail**
  - Arriving packets get dropped when queue is full regardless of flow or importance

- **Important distinction:**
  - FIFO: scheduling discipline
  - Drop-tail: drop (buffer management) policy
**FIFO + Drop-tail Problems**

- **FIFO Issues:** In a FIFO discipline, the service seen by a flow is **convoluted** with the **arrivals** of packets from all other flows!
  - **No isolation** between flows: full burden on e2e control
  - **No policing:** send more packets → get more service

- **Drop-tail issues:**
  - Routers are forced to have have **large queues** to maintain high utilizations
  - Larger buffers => larger steady state queues/delays
  - **Synchronization:** end hosts react to same events because packets tend to be lost in bursts
  - **Lock-out:** a side effect of burstiness and synchronization is that a few flows can monopolize queue space
Design Objectives

- Keep throughput high and delay low (i.e. knee)
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes
Queue Management Ideas

- **Synchronization, lock-out:**
  - Random drop: drop a randomly chosen packet
  - Drop front: drop packet from head of queue

- **High steady-state queuing vs burstiness:**
  - Early drop: Drop packets before queue full
  - Do not drop packets “too early” because queue may reflect only burstiness and not true overload

- **Misbehaving vs Fragile flows:**
  - Drop packets proportional to queue occupancy of flow
  - Try to protect fragile flows from packet loss (e.g., color them or classify them on the fly)

- **Drop packets vs Mark packets:**
  - Dropping packets interacts w/ reliability mechanisms
  - Mark packets: need to trust end-systems to respond!
Packet Drop Dimensions

Aggregation

Per-connection state

Class-based queuing

Drop position

Single class

Head

Tail

Random location

Early drop

Overflow drop

Cours de C. Pham, Univ. Lyon 1

Réseaux étendus
Random Early Detection (RED)

Max thresh

Min thresh

Average Queue Length

$P(\text{drop})$

$1.0$

$max_p$

$min_{th}$

$max_{th}$

Avg queue length
Random Early Detection (RED)

- **Maintain running average of queue length**
  - Low pass filtering
- **If** $\text{avg } Q < \text{min}_{th}$ **do nothing**
  - Low queuing, send packets through
- **If** $\text{avg } Q > \text{max}_{th}$, **drop packet**
  - Protection from misbehaving sources
- **Else** mark (or drop) packet in a manner proportional to queue length & bias to protect against synchronization
  - $P_b = \max_p(\text{avg} - \text{min}_{th}) / (\text{max}_{th} - \text{min}_{th})$
  - Further, bias $P_b$ by history of unmarked packets
  - $P_a = P_b / (1 - \text{count} \times P_b)$
RED Issues

- **Issues:**
  - Breaks synchronization well
  - Extremely sensitive to parameter settings
  - Wild queue oscillations upon load changes
  - Fail to prevent buffer overflow as #sources increases
  - Does not help fragile flows (e.g., small window flows or retransmitted packets)
  - Does not adequately isolate cooperative flows from non-cooperative flows

- **Isolation:**
  - Fair queuing achieves isolation using per-flow state
  - RED penalty box: Monitor history for packet drops, identify flows that use disproportionate bandwidth
RED with Multiple Thresholds

Discard Probability

“Red” Packets

“Yellow” Packets

“Green” Packets

Average Queue Length

0 1

0 “Red” Threshold “Yellow” Threshold “Green” Threshold Full

source Juha Heinänen
Cours de C. Pham, Univ. Lyon 1
Main ideas

- Decouple congestion & performance measure
- “Price” adjusted to match rate and clear buffer
- Marking probability exponential in `price`
Comparison of AQM Performance

**REM**
queue = 1.5 pkts  
utilization = 92%  
\( \gamma = 0.05, \ \alpha = 0.4, \ \phi = 1.15 \)

**DropTail**

**RED**
min_th = 10 pkts  
max_th = 40 pkts  
max_p = 0.1
Service Specification

- **Loss**: probability that a flow’s packet is lost
- **Delay**: time it takes a packet’s flow to get from source to destination
- **Delay jitter**: maximum difference between the delays experienced by two packets of the flow
- **Bandwidth**: maximum rate at which the source can send traffic
- **QoS spectrum**:

---

**Best Effort** --- **Leased Line**
Hard Real Time: Guaranteed Services

- **Service contract**
  - Network to client: guarantee a deterministic upper bound on delay for each packet in a session
  - Client to network: the session does not send more than it specifies

- **Algorithm support**
  - Admission control based on worst-case analysis
  - Per flow classification/scheduling at routers
Soft Real Time: Controlled Load Service

- **Service contract:**
  - Network to client: similar performance as an unloaded best-effort network
  - Client to network: the session does not send more than it specifies

- **Algorithm Support**
  - Admission control based on measurement of aggregates
  - Scheduling for aggregate possible
Traffic and Service Characterization

- To quantify a service one has two know
  - Flow’s traffic arrival
  - Service provided by the router, i.e., resources reserved at each router

- Examples:
  - Traffic characterization: token bucket
  - Service provided by router: fix rate and fix buffer space
    - Characterized by a service model (service curve framework)
Régulation du trafic: Leacky Bucket

(a) Robinet
Seau percé
Écoulement de l’eau avec un débit constant

(b) Ordinateur hôte
Paquet
Flux non régulé
Le seau contient des paquets
Flux régulé
Réseau

© Pearson Education France
Régulation du trafic: Token Bucket

Un jeton est ajouté dans le seau toutes les $\Delta T$ secondes

Le seau avec trois jetons

(a) Réseau

(b) Réseau

© Pearson Education France
Token Bucket

- **Caractérisé par 3 paramètres (b, r, R)**
  - b: capacité en jetons
  - r: taux de génération des tokens
  - R: taux d'émission maximum (e.g., R = capacité du lien)

- **Un bit est transmis s'il y a un jeton**
  - Quand un bit est transmis, un jeton est consommé
Token Bucket

Example

- $B = 4000$ bits, $R = 1$ Mbps, $C = 10$ Mbps
- Packet length = 1000 bits
- Assume the bucket is initially full and a “large” burst of packets arrives

istoica@cs.cmu.edu
Token Bucket

time = 0

Cours de C. Pham, Univ. Lyon 1
Courbe des arrivées

\[ A(t) - \text{number of bits received up to time } t \]

\[ \text{bits} \]

\[ \text{time} \]

istical@cs.cmu.edu
Caractérisation avec un Token Bucket

- Courbes des arrivées – nombre maximum de bits transmis pendant un temps t
- Utilise le Token Bucket pour borner le taux d'arrivée
What is a Service Model?

- The QoS measures (delay, throughput, loss, cost) depend on offered traffic, and possibly other external processes.
- A service model attempts to characterize the relationship between offered traffic, delivered traffic, and possibly other external processes.
Arrival and Departure Process

\[ R_{in}(t) = \text{arrival process} = \text{amount of data arriving up to time } t \]

\[ R_{out}(t) = \text{departure process} = \text{amount of data departing up to time } t \]
Delay and Buffer Bounds

E(t) = Envelope

Maximum delay

Maximum buffer

S(t) = service curve

bits

Cours de C. Pham, Univ. Lyon 1
Le contrôle d'admission: principes

- **Pessimiste**
  - admission sur le débit crête $D_{\text{max}}$,
  - pour des paquets de taille fixe, on utilise le temps entre 2 paquets,
  - pour des paquets de taille variable, on doit en plus définir un intervalle de temps pour le calcul de $D_{\text{max}}$,
  - simple, sûr, mais gaspille de la bande passante.

- **Plus optimiste**
  - plusieurs critères en plus comme de débit moyen, le débit burst,...
  - plus complexe à mettre en œuvre, mais exploite au mieux l'aspect statistique du trafic.
  - Ex: utilisation d'un Token Bucket (RSVP)
L'ordonnancement des paquets: principes

- **Décider quand et quel paquet envoyer sur la ligne**
  - Généralement réalisé sur l'interface de sortie
Politique d'ordonnancement (1)

- **Priority Queuing**: classes have different priorities; class may depend on explicit marking or other header info, eg IP source or destination, TCP Port numbers...
- Transmit a packet from the highest priority class with a non-empty queue
- Preemptive and non-preemptive versions
Politique d'ordonnancement (2)

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

- **Advantages: protection among flows**
  - Misbehaving flows will not affect the performance of well-behaving flows
    - Misbehaving flow – a flow that does not implement any congestion control
  - FIFO does not have such a property

- **Disadvantages:**
  - More complex than FIFO: per flow queue/state
  - Biased toward large packets – a flow receives service proportional to the number of packets
Generalized Processor Sharing (GPS)

- **Assume a fluid model of traffic**
  - Visit each non-empty queue in turn (RR)
  - Serve infinitesimal from each
  - Leads to “max-min” fairness

- **GPS is un-implementable!**
  - We cannot serve infinitesimals, only packets

max-min fairness

Soit un ensemble de sources 1,..,n demandant des ressources $x_1,..,x_n$ avec $x_1 < x_2 .. < x_n$ par exemple. Le serveur a une capacité $C$.

On donne alors $C/n$ à la source 1. Si $C/n > x_1$, on donne $C/n+(C/n-x_1)/(n-1)$ aux $(n-1)$ sources restantes. Si cela est supérieur à $x_2$, on recommence.
Packet Approximation of Fluid System

- **GPS un-implementable**

- **Standard techniques of approximating fluid GPS**
  - Select packet that finishes first in GPS assuming that there are no future arrivals (emulate GPS on the side)

- **Important properties of GPS**
  - Finishing order of packets currently in system independent of future arrivals

- **Implementation based on virtual time**
  - Assign virtual finish time to each packet upon arrival
  - Packets served in increasing order of virtual times
Fair Queuing (FQ)

- Idea: serve packets in the order in which they would have finished transmission in the fluid flow system.
- Mapping bit-by-bit schedule onto packet transmission schedule.
- Transmit packet with the lowest finish time at any given time.
FQ Simple Example

Cannot preempt packet currently being transmitted
Round Number and Finish Number

- **Single flow**: *clock ticks when a bit is transmitted*. For packet k:
  - \( P_k \) = length, \( A_k \) = arrival time, \( S_k \) = begin transmit time, \( F_k \) = finish transmit time
  - \( F_k = S_k + P_k = \max(F_{k-1}, A_k) + P_k \)

- **Multiple flows**: *clock ticks when a bit from all active flows is transmitted* \( \rightarrow \) round number
  - Can calculate \( F_k \) for each packet if number of flows is known at all times
    - \( F_k = \) current round number + size of packet \( k \), inactive case
    - \( F_k = \) largest \( F_k \) in the queue + size of packet \( k \), active case
  - \( F_{i,k,t} = \max(F_{i,k-1,t}, R_t) + P_{i,k,t} \)
  - In packet approximation, finish number indicate a relative order (service tag) in which a packet is to be served. Finish time \( \neq \) finish number
Example

- The round number increases at a rate inversely proportional to the number of active connections
  - Thus is only used for computing finish numbers

- Largest finish number in a connection's queue is the connection's finish number

Example

- Suppose packets of size 1, 2 and 2 units arrive at a FQ scheduler at time for connection A, B and C. Also, assume that a packet of size 2 arrive for connection A at time 4. The link service rate is 1 unit/s. Compute the finish number of all packets.
Illustration
FQ Advantages

- FQ protect well-behaved flows from ill-behaved flows
- Example: 1 UDP (10 Mbps) and 31 TCP’s sharing a 10 Mbps link
Weighted Fair Queueing

- Variation of FQ: Weighted Fair Queueing (WFQ)
- Weighted Fair Queueing: is a generalized Round Robin in which an attempt is made to provide a class with a differentiated amount of service over a given period of time
Implementing WFQ

- WFQ needs per-connection (or per-aggregate) scheduler state → implementation complexity.
  - complex iterated deletion algorithm
  - complex sorting at the output queue on the service tag

- WFQ needs to know the weight assigned for each queue → manual configuration, signalling.

- WFQ is not perfect...

- Router manufacturers have implemented as early as 1996 WFQ in their products
  - from CISCO 1600 series
  - Fore System ATM switches
Big Picture

- **FQ does not eliminate congestion → it just manages the congestion**
- **You need both end-host congestion control and router support for congestion control**
  - end-host congestion control to adapt
  - router congestion control to protect/isolate
- **Don’t forget buffer management: you still need to drop in case of congestion. Which packet’s would you drop in FQ?**
  - one possibility: packet from the longest queue
Further readings

- See http://www.cnaf.infn.it/~ferrari/ispn.html for Quality of Service list of papers
- See http://www.cnaf.infn.it/~ferrari/sched.html for scheduling list of papers
QoS ARCHITECTURES
Stateless vs. Stateful QoS Solutions

- **Stateless solutions** – routers maintain no fine grained state about traffic
  - ↑ scalable, robust
  - ↓ weak services

- **Stateful solutions** – routers maintain per-flow state
  - ↑ powerful services
    - • guaranteed services + high resource utilization
    - • fine grained differentiation
    - • protection
  - ↓ much less scalable and robust
Integrated Services (IntServ)

- An architecture for providing QoS guarantees in IP networks for individual application sessions
- Relies on resource reservation, and routers need to maintain state information of allocated resources (eg: g) and respond to new Call setup requests
Integrated Services Model

- **Flow specification**
  - Leaky Bucket, Token Bucket
- **Routing**
- **Admission control**
- **Policy control**
- **Resource reservation**
  - RSVP
- **Packet scheduling**
  - WFQ, CBQ, RED
Integrated Services: Classes

- **Guaranteed QOS:** this class is provided with firm bounds on queuing delay at a router; envisioned for hard real-time applications that are highly sensitive to end-to-end delay expectation and variance.

- **Controlled Load:** this class is provided a QOS closely approximating that provided by an unloaded router; envisioned for today’s IP network real-time applications which perform well in an unloaded network.
Signaling semantics

- Classic scheme: sender initiated
- SETUP, SETUP_ACK, SETUP_RESPONSE
- Admission control
- Tentative resource reservation and confirmation
- Simplex and duplex setup; no multicast support
RSVP for the IntServ approach

- **Resource reSerVation Protocol**
- **What is RSVP?**
  - Method for application to specify desired QoS to net
  - Switch state establishment protocol (signaling)
  - Multicast friendly, receiver-oriented
  - Simplex reservations (single direction)
- **Why run RSVP?**
  - Allows precise allocation of network resources
  - Guarantees on quality of service
  - Heterogeneous bandwidth support for multicast
  - Scalable (?)

source Gordon Schaffee
Cours de C. Pham, Univ. Lyon 1
Resource Reservation

- Senders advertise using PATH message
- Receivers reserve using RESV message
  - Flowspec + filterspec + policy data
  - Travels upstream in reverse direction of Path message
- Merging of reservations
- Sender/receiver notified of changes
RSVP Functional Diagram

Host

- Application
- Policy Control
- Admissions Control
- Packet Classifier
- Packet Scheduler

Router

- Routing Process
- Policy Control
- Admissions Control
- Packet Classifier
- Packet Scheduler
Stateful Solution: Guaranteed Services

- Achieve per-flow bandwidth and delay guarantees
  - Example: guarantee 1MBps and < 100 ms delay to a flow
Stateful Solution: Guaranteed Services

- Allocate resources - perform per-flow admission control
Stateful Solution: Guaranteed Services

- Install per-flow state
Stateful Solution: Guaranteed Services

- Challenge: maintain per-flow state consistent
Stateful Solution: Guaranteed Services

- Per-flow classification
Stateful Solution: Guaranteed Services

- Per-flow buffer management
Stateful Solution: Guaranteed Services

- Per-flow scheduling
Stateful Solution Complexity

- **Data path**
  - Per-flow classification
  - Per-flow buffer management
  - Per-flow scheduling

- **Control path**
  - install and maintain per-flow state for data and control paths
Stateless vs. Stateful

- **Stateless solutions are more**
  - scalable
  - robust

- **Stateful solutions provide more powerful and flexible services**
  - guaranteed services + high resource utilization
  - fine grained differentiation
  - protection
Question

- Can we achieve the **best of two worlds**, i.e., provide services implemented by **stateful** networks while maintaining advantages of **stateless** architectures?
  - Yes, in some interesting cases. DPS, CSFQ.

- Can we provide **reduced state services**, i.e., maintain state only for larger granular flows rather than end-to-end flows?
  - Yes: Diff-serv