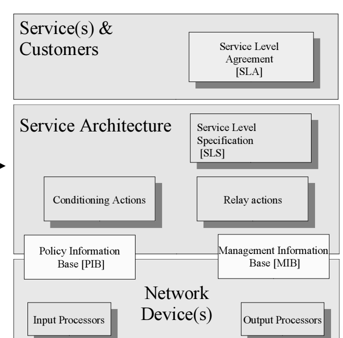
Integrated Services in the Internet

Lecture for QoS in the Internet –course

17.11.2005 Mika Ilvesmäki

The QoS story so far...

- Where are we in this lecture:
  - Low level mechanisms (building blocks of the QoS) have been dealt with
    - Schedulers, queuing, routing
  - Time to advance to building service architectures using the building blocks
  - Time to apply engineering visions



Knowledge gained in this lecture

- After this lecture you should be able to
  - Explain the constraints and objectives for the development of Integrated Services –architecture
  - Explain the service classes of IntServ and use the flow model to estimate traffic behavior in an IntServ router
  - Explain, in a detailed level, the architecture of an IntServ router
  - Understand the weaknesses of the per-flow approach

Outline

- History and preliminary concepts
  - types of Internet applications
  - general QoS concepts
- *Concepts of IntServ*
  - flow model
  - service classes
- *Building the IntServ-router*
  - routing, scheduling
- Notes on future

History

- It was 1991...
  - and there was not (that much) traffic in the internet
  - No WWW until 1993
  - no other multimedia... yet
    - multicast was already designed, but it was just starting with IETF audio- and videocasts in MBone
- Some people observed some, and anticipated more, problems due to multimedia-applications

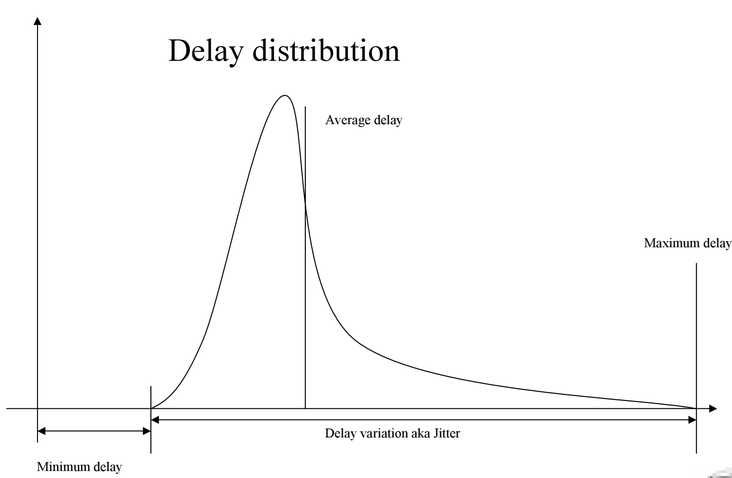
Application types

- Elastic
  - All tolerant ”old-fashioned” Internet applications
    - FTP, Usenet News, E-mail,
- Tolerant playback applications
  - One-way video feed, oneway broadcast
    - Some tolerance achieved with play-out buffers
- Intolerant playback applications
  - Applications that need data to be delivered in real-time
    - low delay, no jitters, enough bandwidth
  - Two way conversations (IP phone)

Quantitative QoS-parameters

- *Available bit rate/ bandwidth*
  - How fast you are allowed to send packets to the network?
- *Packet discard / Data loss*
  - What packets are dropped in case of congestion?
- *Delay*
  - Time for the packet to reach its destination
  - How long is your data relevant?
- *Variation of delay / Jitter*
  - effectively kills the usability of Voice over IP –applications

Delay and delay variation



Original design objectives for IntServ

- Build a multicast network with videoconferencing ability
  - Only a few senders at a time
    - real-time
    - low packet loss
    - no congestion control (UDP)
  - VoIP not expected!!
- Protect multimedia traffic from TCP effects and vice cersa
- Objective: Preserve the datagram model of IP networks AND provide support for resource reservations and end-to-end performance guarantees to individual or groups of traffic flows

Integrated Services

- Provide Best Effort as before
  - no reservations for TCP traffic
  - possibility to use adaptive applications
  - sometimes BandWidth is enough
- Provide resources for multimedia traffic
  - multicast streams are long lasting, therefore state setups are ok
    - Caveat!!: VoIP is not OK !!
- Provide services for individual users and their applications!!
  - aka per-flow approach
- Capability requirements (to build IntServ-networks):
  - functions in individual network elements (router enhancements)
  - way(s) to communicate the requests between elements (protocol: RSVP)

Integrated as in Integrating real-time services to best-effort network

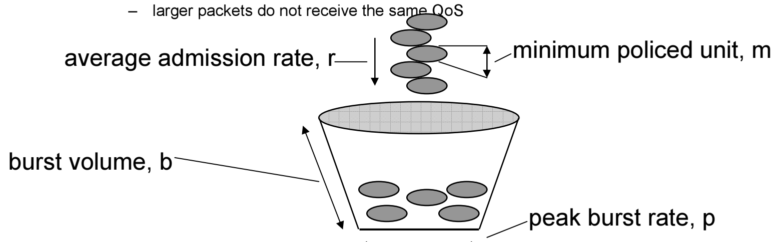
Tools of IntServ

- Signalling to convey the traffic contract
  - RSVP, separate lecture
- Admission control to determine whether new flow fits into the network
- Policing & Shaping to keep the existing flows within the negotiated parameters
- Network design & engineering to keep blocking probability low

HELSINKI UNIVERSITY OF TECHNOLOGY
Mika Ilvesmäki, D.Sc. (Tech.)

Flow model of IntServ

- A flow (in IntServ) is a distinguishable stream of related datagrams that results from a single user activity and requires the same QoS
  - the finest granularity of packet stream that can be identified
- Flow model described by a leaky bucket
  - token rate, rate of leaky bucket (r): 1 byte/s - 40 Terabytes/s
  - token-bucket depth (b): 1 byte - 250 Gbytes
  - peak traffic rate (p): 1 byte - 40 Terabytes/s
  - minimum policed unit (m): amount of data in the IP packet (other protocols, user data)
  - maximum packet size (M): maximum size of the packet within this flow (bytes)
    - larger packets do not receive the same QoS



Controlled load service (RFC 2211)

- provides unloaded network conditions
  - for applications requiring reliable and enhanced best-effort service
  - aims to provide service that closely approximates traditional best-effort in a lightly loaded or unloaded network environment -> better than best effort
- intended for adaptive applications
  - applications provide network an estimation of the traffic it is about to send
  - acceptance (by the network) of a controlled load request implies a commitment to provide better than best-effort
- priority service with admission control
- no fragmentation, packets must comply to MTU

Guaranteed service (RFC 2212)

- for non-adaptive applications requiring fixed delay bound and a bandwidth guarantee
- WFQ service (refer to lecture on queuing mechanisms)
- **computes and controls the maximum queuing delay**
  - guarantees that packets will arrive within a certain delivery time and will not be discarded because of queue overflows, provided that flow's traffic stays within the bounds of its specified traffic parameters
- does not control minimal or average delay of traffic, nor is there control or minimization for jitter
- no packet fragmentation is allowed, packets larger than M are nonconforming.
- traffic policing with simple policing and reshaping

Delay calculation for Guaranteed Service

End-to-end queuing delay:

$$Q_{delay} = \frac{(b-M)(p-R)}{R(p-r)} + \frac{(M+C_{tot})}{R} + D_{tot}, \text{ if } p > R \geq r \quad \text{or} \quad Q_{delay} = \frac{(M+C_{tot})}{R} + D_{tot}, \text{ if } R \geq p \geq r$$

- p=peak rate of flow (bytes/s) (***Tspec***)
- b=bucket depth (bytes) (***Tspec***)
- r=token bucket rate (bytes/s) (***Tspec***)
- R=bandwidth (service link rate) (***Rspec***)
- m=minimum policed unit (bytes) (***Tspec***)
- M=maximum datagram size (bytes) (***Tspec***)
- C=packet delay caused by flow parameters (bytes) (***Rspec***)
- D=rate independent delay caused by network nodes (μs)

- The delay estimates are based on a so called fluid model
  - C and D indicate the deviation of the node from the ideal fluid model
- There is no control (in GS) for
  - minimal or average delay
  - propagation delay
- No estimate for jitter
- Only thing promised is the maximum delay.

Estimate on required buffer space:

$$B_{size} = M + \frac{(b-M)(p-X)}{(p-r)} + X\left(\frac{C_{sum}}{R} + D_{sum}\right), \text{ where}$$

$$X = \begin{cases} r, \text{ if } \frac{b-M}{p-r} < \frac{C_{sum}}{R} + D_{sum} \\ R, \text{ if } \frac{b-M}{p-r} \geq \frac{C_{sum}}{R} + D_{sum} \wedge \text{p} > \text{r} \\ \text{p, otherwise} \end{cases}$$

TOKEN_BUCKET_TSPEC

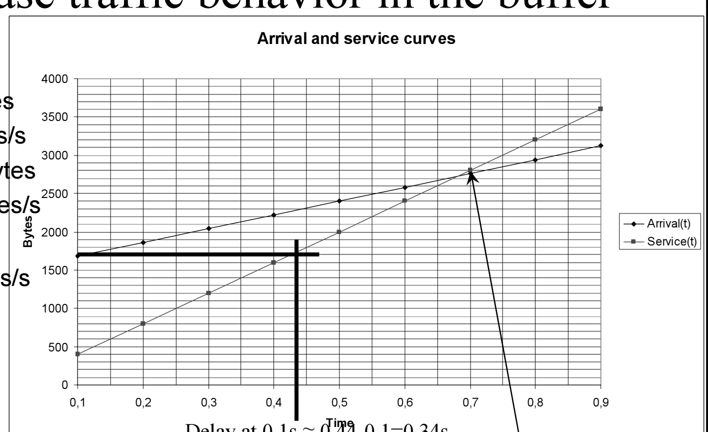
- Guaranteed service is invoked by a sender specifying the flow parameters in the Tspec
- Controlled-load service is described in Tspec
- Describes traffic with
  - bucket rate (r)
  - peak rate (p)
  - bucket depth (b)
  - minimum policed unit (m) and maximum packet size (M)

Rspec

- Receiver determines/requests a desired network service level with Rspec
- Used only in Guaranteed Service
  - Question: How does ”Controlled load” work?
- Describes the service requirements with
  - Service rate (R), R>=r, may be higher than requested (taking into account the p (peak rate)
  - Slack Term (S), microseconds, describing the difference between the desired delay and the delay obtained by using a reservation level of R.

Worst case traffic behavior in the buffer

- Tspec
  - M=1500 bytes
  - p=3500 bytes/s
  - Tbd=1500 bytes
  - tbr=1800 bytes/s
- Rspec
  - R=4000 bytes/s
- Max. Delay
  - M/R=0,375s



Buffer empty!

Note and exercise for final exam: Type in the equations couple of slides back and play with different parameter combinations. Final exam is likely to have a question where you have to explain a figure like the above.

QoS in the Internet-routers

- New router functionality
  - Traffic shaping
  - Admission control
    - To control resources
  - Differential congestion management
    - advanced queue management algorithms
    - CBQ, WFQ, etc.
  - Consistent handling of packets
    - State, 'global' knowledge of policy and QoS/CoS decisions

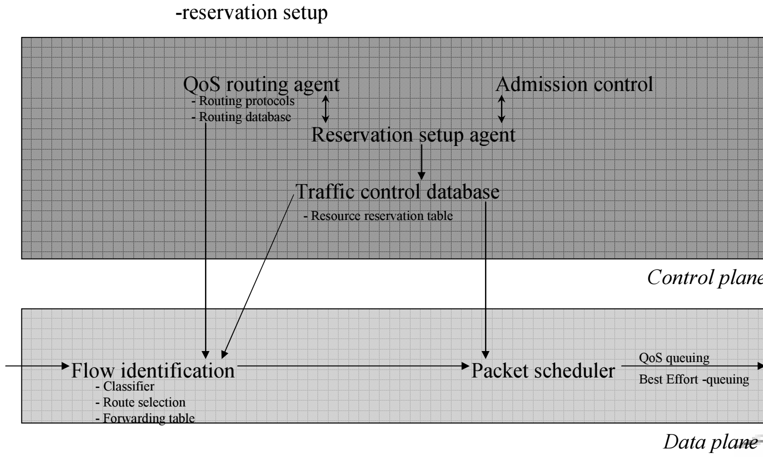
"There is an inescapable requirement for routers to be able to reserve resources in order to provide special QoS for specific user packet streams."

HELSINKI UNIVERSITY OF TECHNOLOGY
Mika Ilvesmäki, D.Sc. (Tech.)

IntServ router implementation reference model

In IntServ the resources are explicitly managed with

-packet scheduler
-classifiers
-admission control
-reservation setup



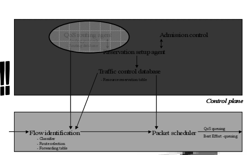
IntServ node characterization

- General descriptive parameters used to characterize the QoS capabilities of nodes in the path of a packet flow (RFC 2215)
  - NON_IS_HOP
    - the break bit indicating a break in the QoS chain
    - set by the device that is not IntServ compliant or knows such devices to exist in the path
  - NUMBER_OF_IS HOPS
  - AVAILABLE_BANDWIDTH
    - 1 byte/s ... 40 Terabytes/s
  - MINIMUM_PATH_LATENCY
    - speed-of-light + packet processing limitations
  - PATH_MTU
  - TOKEN_BUCKET_TSPEC
    - only used by the sender and the edge node

Router blocks: QoS routing

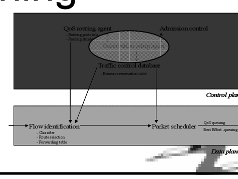
- Current Internet uses distributed route calculation
  - Every router decides for itself what is the best route to a given destination.
- In the future Internet route calculation has to be more centralized
  - Ability to compute the path at the source
  - Ability to distribute information about network topology and link attributes
  - Ability to do explicit routing
  - Resource reservations and link attrribute updates

QoS routing is not specified in any manner within the IntServ!!



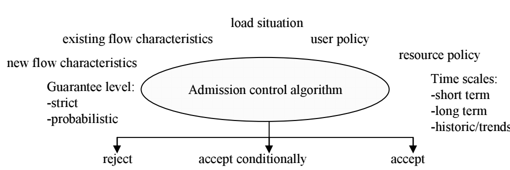
Router blocks: Reservation setup

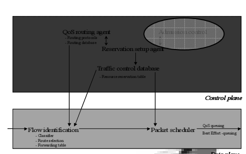
- Need for a protocol
  - RSVP
- Hop by hop state establishment
  - traffic characteristics
- Billing/accounting setup
- More on RSVP in the provisioning lecture



Router blocks: Admission control

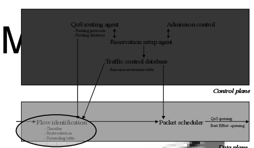
- Before a flow is accepted it has to pass the admission control test
- Parameter based
  - precise characterization of a traffic flow
  - difficulty of accurately modeling traffic
- Measurement based
  - probabilistic traffic characterization
  - good level of guarantee to resource utilization ratio





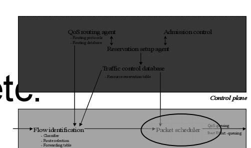
Router blocks: Flow identification

- Identify to what flows (if any) packets belong to
  - must be performed to every incoming packet
    - Multifield classification decides the appropriate queue
  - requires fast hardware if (and when) performed at wire speed
  - 64 byte packets arrive in 622 M back to back in less than 1μs



Router blocks: Packet scheduling

- Refer to the previous lecture on scheduling algorithms
  - WFQ (primary choice)
    - explained with the fluid model
  - GPS
  - PGPS
  - WF^{2}Q
  - Hierarchical WFQ
  - SCFQ, WRR, DRR, CRR etc. etc.



IntServ problems

- Resources
  - OK in small networks
    - provides for end-to-end exact QoS
  - What about large networks?
    - router capacity for resource reservation cannot be scaled on per-flow basis (in the Internet core)
- For IntServ to function, especially for Guaranteed Service, every node on the path must implement the IntServ functionality
- Router requirements are high
  - RSVP, admission control, MF classification and packet scheduling

Integrated Services will be deployed first in intranets and other local environments where scaling and policy control are much less challenging.
-Bob Braden-

Future of IntServ

- In the core there might be a large amount of reservations to be updated, so you have to:
  - not isolate individual flows
  - map flows into fixed number of service classes
  - don’t bother reservation messages
  - keep state on the edges
  - > DiffServ (next week’s lecture)

The problems of per-flow approach

- Scalability
  - If the amount of information grows faster or at the same pace in the core as it does at the edge the solution in question DOES NOT SCALE well.
- Millions of users are hard to manage one by one according to their individual wishes.
  - qualitative QoS -> not IntServ
- It is easier to decide *which* packet is forwarded and which dropped or delayed than to determine *when* a packet should be forwarded.
  - qualitative QoS -> not IntServ
- Qualitative is easier to <u>implement</u> than quantitative
  - IntServ is not likely to be the <u>widely</u> implemented QoS solution!!