Chapter 3: Multimedia Systems – Communication Aspects and Services

3.3: Quality of Service and Resource Management

- QoS Parameters
- QoS Negotiation
- Admission Control
- Traffic Shaping
- Scheduling
- Error Control
- Resource Monitoring and Adaptation

Quality of Service

Multimedia streaming has certain demands to the delivery of the data; the most important parameters are:

**Throughput** (bits/s)
- Which minimum/maximum/average data rate is necessary?

**Transmission delay** (ms)
- Which maximum delay is tolerable?

**Jitter** (ms)
- Which fluctuations in the transmission delay are tolerable?

**Availability** (%)
- With which probability the communication service is available?

Different applications have different requirements to these parameters.

<table>
<thead>
<tr>
<th>Application</th>
<th>Reliability</th>
<th>Delay</th>
<th>Jitter</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Mail</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>File Transfer</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Web Access</td>
<td>high</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Remote Login</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>Audio streaming</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>medium</td>
</tr>
<tr>
<td>Video streaming</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>IP Telephony</td>
<td>low</td>
<td>high</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Video Conference</td>
<td>low</td>
<td>high</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>

*high – medium – low*: requirements to the QoS parameter by the application; critical parameters; a certain degree of fulfillment of the application's requirement is needed for good application usage, but problems could be tolerated for a short time;

*red*: highly critical parameters; if the requirement is not fulfilled, usage is not possible.

Translation of QoS Parameters

QoS parameters like before are network specific parameters; an application works with more abstract terms – during connection establishment, translation between layers is necessary:

- **Translation „Human Interface“ – „Application QoS“**
  - Tuning service provides Graphical User Interface (GUI) for input/output of application QoS

- **Translation „Application QoS“ – „System QoS“**
  - Translates video frame size to network packet size
  - May be connected with segmentation/reassembly functions

- **Translation „System QoS“ – „Network QoS“**
  - Maps system QoS (e.g. packet end-to-end delay) to the network QoS parameters (e.g. end-to-end delay of cells in ATM networks)

Translation is bidirectional. This can cause problems:

- Example: Video rate and video frame size determine the throughput for the network. Thus a throughput reduction may result in either lowering the quality of the image or the rate of successive images.
Scaling

Scaling is used for adapting a stream to available resources:
- Subsample a data stream to present a fraction of its original contents
  - **Transparent scaling**: can be applied independently from upper protocol layers, e.g. by dropping some identifiable portions of the data stream
  - **Non-transparent scaling**: requires interactions of the transport system with upper layers, e.g. modification of some coding algorithm parameters or recording in a different format.
- Scaling can be applied to audio and to video data:
  - Transparent scaling is not good for audio, because presenting a fraction is easily noticed by the human listener
    → Non-transparent scaling, e.g. by changing the sampling rate is preferred for audio
  - For video, the suitable scaling method depends on the compression methods:
    - Temporal scaling reduces number of transmitted video frames, best suited for video with individual self-contained frames
    - Spatial scaling reduces number of pixels of each image
    - Frequency scaling reduces the number of DCT coefficients in coding algorithm
    - Amplitude scaling reduces the color depth by coarser quantization of the DCT coefficients

Resource Management

Main goal of Resource Management: providing guaranteed delivery of multimedia data
- Three main actions:
  - Reserve and allocate resources (end-to-end) during establishment (QoS negotiation, resource reservation)
  - Provide resources according to a QoS specification (Admission control, traffic shaping, scheduling)
  - Adapt to resource changes during data processing (Resource monitoring and adaptation)

Relations between QoS and resources:
- QoS parameters ⇔ resource quantity allocated to the service and resource scheduling
  → Resource Management handles different mappings between QoS parameters and the corresponding resources

QoS Negotiation

To give guarantees to QoS parameters, QoS values for those parameters have to be negotiated before transmission:
- **Bilateral Peer-to-Peer-Negotiation**: takes place between two service users
  - Service provider is not allowed to modify proposed values
  - Callee may modify proposed QoS values
- **Bilateral Layer-to-Layer-Negotiation**: takes place between service user and service provider
  - Callee may modify proposed QoS values
- **Unilateral Negotiation**: service user or service provider is not allowed to modify proposed values
  Principle: „take it or leave it“ (dictatorship!)
QoS Negotiation

Triangular Negotiation for a Bounded Target:
- QoS represented through two bounds:
  - Target (average value)
  - Lowest quality available (LQA, minimal value)
- Goal: negotiate target value

Triangular Negotiation for Contractual Value:
- QoS represented through two bounds:
  - Minimal requested value
  - Bound of strengthening
- Goal: agree on a contractual value.

QoS Negotiation

QoS negotiation is provided by different Reservation Protocols:
1. **ST-II (Stream Protocol II)** for single sender/multiple receivers
   - QoS parameters related to **throughput** are negotiated by triangular negotiation for a bounded target
   - QoS parameters related to **delay** are not negotiated, only examined
     - Calling user specifies maximum transit delay
     - Each ST-agent on the way estimates the average transit delay
     - Provider presents total estimated transit delay in the indication
     - Called user decides if the (expected) delay is sufficient
   - QoS parameters related to **error control** are not negotiated

2. **RSVP (Reservation Protocol)** for multiple senders/multiple receivers and **RCAP (Real Time Channel Administration Protocol)** for single sender/single receiver
   - QoS parameters are negotiated by triangular negotiation
   - Needed for all protocols: **Resource Admission Control**

Resource Reservation Protocol RSVP

In RSVP, three categories for reservation can be specified:
1.) **Best Effort**: Take all free resources you can get
2.) **Rate Sensitive**: a guaranteed (minimal) data transmission rate is necessary
3.) **Delay Sensitive**: a guaranteed (maximal) delay is necessary

Procedure of RSVP:
- Reservation is made in advance, before data are sent
- The sender sends a *RSVP Path Message* to the receiver. Information about the routers on the network path are being collected and communicated to the receiver. (Additionally, also flow specifications for the receiver are provided.)
- The receiver sends back *RSVP Reservation Messages* along the collected path, which contain its flow specification. Every router on the path reserves resources accordingly.
- If a reservation is not possible with a router, an error message is sent back.
- As soon as the RSVP Reservation Message reaches the sender, the complete path is reserved, the sender begins to send its data.
- To delete outdated reservations, timeouts are defined. If the receiver does not refresh its reservation before expiration of the timer, it is deleted.
- Also usable with multicast
Real Time Channel Administration Protocol
RCAP

- Only for single sender/single receiver
- Reservations are made by immediate allocation in one round trip

**Server tests allocate resources by**
- **pessimistic approach**: worst case allocation, e.g. allocating bandwidth according to peak rate
- **optimistic approach**: allocates resource to an average workload, e.g. allocating bandwidth according to average rate over a specific time

Destination tests in receiver computes local constraints of the intermediate nodes
- Adjust resources at way back to the sender

Server tests
- Deterministic test: deterministic bandwidth test for guaranteed services
- Statistical test: statistical bandwidth test for guaranteed services
- Delay bound computation: computes minimal node delay, depends on scheduling strategy
- Buffer test: allocates buffer and computes maximum node delay according to reserved buffer

Traffic Shaping

**Traffic Shaping:**
- Description of the flow's characteristics is given to the network provider such that the network knows which traffic is to expect (admission control)
- Given a description of the flow's traffic, the network provider can determine if the flow should be allowed to send
- Moreover, the network provider can monitor the traffic of the flow and check that the flow behaves as promised (monitoring and policing)
- Traffic sent into the network by a source can be *shaped* due to the flow description

Traffic Shaping schemes may be characterized by e.g.:
- Allowed peak rate
- Maximum burst size
- Average rate over a determined interval
  These parameters may be combined
Traffic Shaping by “Leaky Bucket”

Simplest way of traffic shaping: Try to shape all traffic into isochronous flows by a bucket scheme (leaky bucket). The bucket can be emptied with a maximum rate $\rho$.

- Data is transmitted uniformly with rate $\rho$ → transfer is smoothed
- Bursts bigger than the bucket capacity $\beta$ are discarded → bounded delay
- Easy to implement (FIFO + timer) → Leaky Bucket generates a steady stream but has limitations for bursty traffic

Traffic Shaping by “Token Bucket”

Variation of the leaky bucket scheme: data may be sent if tokens are available in the bucket. Tokens are generated with a certain rate $\rho$ (determined by the network). The generation pauses, if the bucket is full. When we are now trying to send a packet of size $b$ tokens ($b < \beta$):

- If $B \geq b$, then the packet is sent immediately
- If $B < b$ tokens are available, the packet has to wait for the missing $(b-B)$ tokens → Token Bucket permits burstiness but bounds it. The long term transmission rate will be bounded by $\rho$

Token Bucket with Leaky Bucket

Token bucket has a certain problem:

- If a flow stops for a while, then the token bucket will be full. When the flow restarts then the network could be monopolized for a while by this flow

This can be avoided by a combination of token bucket and leaky bucket:

- The flow is regulated by token bucket but smoothed by a subsequent leaky bucket both of size $\beta$. The leaky bucket is emptied with a rate $C$.
- $C$ should be substantially larger than $\rho$ since the scheme should be basically a token bucket → bursty traffic is permitted but the scheme regulates it
- The long term average rate is bounded by $\rho$ ($< C$)

Scheduling

Scheduling: determining a sending order of data packets in intermediate nodes (routers)

- Simple scheduling strategies (e.g. FIFO) are not suitable for QoS based scheduling because strict performance guarantees are not provided
- There are different scheduling principles which may provide these guarantees:
  - Weighted Fair Queueing
  - Virtual Clock
  - Delay Earliest Due Date (D-EDD)
  - Jitter Earliest Due Date (J-EDD)
  - Stop-and-Go
  - Rate Controlled Static Priority
- Differences to simple scheduling strategies:
  - More complex queueing
  - Often time-stamped
  - More complex computation e.g. of deadlines
  - Non-work-conserving (work-conserving = server works whenever there is something to do)
Scheduling inside the Network

How can we guarantee that the system keeps its promises for the connections even if traffic shaping rules are violated?

1. Nagle’s algorithm
   - Router in a network keeps a queue for every connection and serves the connections in a packet-wise round-robin strategy
   - Advantages:
     - Connections which offer too many packets will be "punished"
     - Easy to implement
   - Disadvantages:
     - A very long packet will be “dangerous” for others
     - Inconvenient for some traffic patterns (if a packet arrives just after the server was there then the waiting time could become substantial)
   - Thus some modifications were proposed

2. “Bit by Bit Round Robin” (fair queueing)
   - One queue per connection (as before)
   - But: bitwise round robin (instead as of packet-wise Round Robin)
   [assume that the connections would be served bit by bit in a round robin manner; take for transmission the full packet which would have the shortest delivery time if this strategy would be applied; thus a long packet might be interrupted by a newly arriving short packet]

3. “Packet-wise Round Robin”
   - As 2., but a packet in transmission will not be interrupted; the maximum deviation from 2. is given by the maximum length of one packet

4. “Weighted fair queueing”
   - Fair queueing gives equal fraction of bandwidth per connection (1/n of total bandwidth if n connections are active)
   - This may not be reasonable if (for example):
     - Connection 1 is between supercomputers
     - Connection 2 is between workstations
   - In this case we would like to give more bandwidth to connection 1
   4.1 Divide the bandwidth into cycles of m bits (where m > n = number of active connections). One bit (at least) per cycle is allocated to each connection. Allocate the m - n “extra bits” to the connections with higher need
   4.2 Each connection i gets a weighted service rate \( g_i \) from the router (\( g_i \) must be greater than \( \rho_i \) if the outside traffic shaping is made by token bucket with leaky bucket rate control; i.e. if \( \rho_i \) is the filling rate of the token bucket for connection i).

Scheduling - Weighted Fair Queueing

- A separate queue per connection
- Round Robin strategy \( \rightarrow \) fairness is provided
- Queueing delay of connection i is bounded by:
  \[
  D_i \leq \frac{\beta_i}{g_i} + \frac{(h_i - 1) \cdot l_i}{g_i} + \sum_{m=1}^{h_i} \frac{I_m}{r_m}, \text{ where}
  \]
  - \( i \) denotes the connection number (or flow number)
  - \( D_i \) denotes the delay of flow i
  - \( \beta_i \) denotes the token bucket size
  - \( g_i \) denotes the weighted flow rate (\( g_i \geq \rho_i \))
  - \( l_i \) denotes the maximum packet size of flow i
  - \( I_m \) denotes the maximum packet size of the network
  - \( h_i \) denotes the total number of hops of flow i
  - \( r_m \) denotes the outbound bandwidth at hop m
Scheduling - Weighted Fair Queueing

Proof of delay bound: \[ D \leq \frac{\beta}{g_i} + \left( \frac{h_i - 1}{g_i} \right) \frac{l_i}{g_i} + \sum_{m=1}^{h_i} \frac{l_m}{r_m} \]

(*) Delay if each router in the network is fully loaded; in this case the token bucket size \( \beta_i \) is served with a rate \( g_i (g_i \geq \rho_i) \). This lasts \( \frac{\beta_i}{g_i} \) seconds.

(**) Somewhat mysterious term: "delay due to packetization effects"

(***) A packet may be delayed (in each hop) by \( l_i \) bit times (\( l_i = \) maximum packet length). Since hop \( m \) serves with rate \( r_m \) this needs \( \frac{l_i}{r_m} \) seconds.

Example:
Let:
- The link bandwidth \( r_1 = r_2 = r_3 = 155 \text{ Mbit/s} \) (ATM speed)
- The maximum packet size of each flow and of the network \( l_1 = l_2 = l_3 = 424 \text{ Bit} \) (ATM cell)
- The weighted flow rate \( g_1 = \rho_1 = 100 \text{ Mbit/s} \) and \( g_2 = \rho_2 = 50 \text{ Mbit/s} \)
- The token bucket size \( \beta_1 = \beta_2 = 1 \text{ Mbit} \)
- Flow 1 is transmitted over 3 hops (\( h_1 = 3 \)), flow 2 over 2 hops (\( h_2 = 2 \))

The delay bounds are
\[ D_1 \leq \frac{\beta_1}{g_1} + \left( \frac{h_1 - 1}{g_1} \right) \frac{l_1}{g_1} + 3 \frac{l_1}{r_1} = 10.0166 \text{ ms} \]
\[ D_2 \leq \frac{\beta_2}{g_2} + \frac{(h_2 - 1)}{g_2} \frac{l_2}{g_2} + 2 \frac{l_2}{r_2} = 20.01395 \text{ ms} \]

Scheduling - Delay Earliest Due Date (D-EDD)

Incoming packets are timestamped
Deadline is computed according to
- average rate or
- peak rate
- The new packet is inserted in calendar (time ordered) queue
- Scheduler serves packet with deadline same as actual serving time
  → non-work-conserving discipline.
  → D-EDD provides deterministic delay bound with granularity bounded by the timer granularity!

Implementation Issues:
- Queueing:
  - Per-Connection-Queueing: needs more complex buffer management
  - Granularity of Queueing: depends on network, e.g. cell in ATM, byte in packet switched networks, bits in our example
- Scheduling:
  - Granularity of weights: unit of the weighted flow \( g_i \)?
    → Depends on granularity of queueing!
  - Strategy is non work-conserving: needs timer control if the bandwidth is not completely allocated
- Advantage:
  - deterministic delay bounds!
- Disadvantage:
  - difficult implementation
### Scheduling - Rate Controlled Servers

To renew the traffic characteristics, in every node of the network rate controllers are added:

- A rate controller may regulate:
  - delay jitter (variation of delay)
  - rate jitter (variation of rate)
- Packets are held by a defined eligibility time before scheduled by the bounded delay server
  - timestamp ordering
  - additional complexity
- Local delays can be computed according to the traffic characteristics at the entrance of the network
- Buffer space required is distributed to the nodes along the network path.

### Scheduling - Stop and Go

Consider the traffic consisting of frames with $T$ bits (the transmission of a frame needs $T$ bit times). A frame contains several packets ("Frame" = "Container").

- The router manages input frames and output frames:
  1. A packet of a frame cannot be forwarded until the entire frame has been received.
  2. "Stop and Go": The packet will be forwarded in the next output frame which starts transmission after the input frame has fully arrived.

### Minimum delay of a router:

- $T$ bit times (necessary for reception of the frame)
- Maximum delay of a router: $2T$ bit times (if the input frame arrives just after an output frame has started)

### Delay bounds

$$ hT \leq D \leq 2hT $$

Jitter $\delta$ for "Stop and Go":

- The only possible jitter is due to different position of a packet in input frame versus output frame
- The "displacement" will be in the interval $(-T; T)$
- The minimum value $-T$ is observed if the frame enters the router at the end of a frame but leaves it at the beginning of the next frame.

### Scheduling - Rate Controlled Static Priority

Each channel has its own rate controller
- Delayed packet is inserted in FCFS queues
- FCFS queues are served with priorities (non-preemptive)
  - $n$ delay bounds are supported – computation depends on traffic specification
  - Scheduler is non-work-conserving; a first implementation in hardware exists.
### Scheduling - Jitter Earliest Due Date (J-EDD)

In the J-EDD scheme each channel has its own rate controller.
- The difference between the deadline and actual finishing time at the current server is carried in a packet.
- This difference is the time the packet is held back in the regulator of the following node (i.e., next hop compensates jitter of previous hop).
- The deadline in time-ordered queue is computed similar to D-EDD.
- J-EDD provides bounds for local delay and local delay jitter.
- Needs additional timestamp field in packet header.

#### Example:
- Channel 1
- Channel 2
- Channel m

### End-to-End Error Control

Reliability of multimedia streams is necessary because:
- Decompression: some audio and video compression schemes cannot tolerate loss of data.
- Human perception: Loss of e.g., digital audio is detected by a human ear very quickly → lower acceptance of multimedia systems.
- Data integrity: E.g., in a recording application, one cannot recover from an error in the first recording; fortunately, recording applications have often less stringent real-time requirements for the receiver.

Necessary: end-to-end error control.
- End-to-End error control consists of:
  - Error detection
  - Error correction

### End-to-End Error Control - Error Detection

- **Problem:** Isolate the errors, e.g., some wrong color in a frame may not matter (hardly visible to a human user), but wrong frame boundaries cannot be recovered.
- **Example:** Within an MPEG video it is important not to loose the I-frames (contains structural information), P- and B-frame losses are tolerated → Structural information needs to be protected, not always content!
- **Example:** Lateness concept. PDUs that arrive too late are useless for the application (e.g., video frame) → PDUs are periodically time-stamped.
- **Example:** Error detection can apply to the time-stamped PDUs → Requires synchronized system clock at sender and receiver.
End-to-End Error Control - Error Correction

• Traditional method for achieving reliability: retransmission
  ➢ Go-Back-N
  ➢ Selective Repeat
  ➢ Selective Reject

• Not suitable for multimedia communication, because:
  ➢ Explicit acknowledgement increases the amount of data to be stored in the sender, e.g. video data
  ➢ Window-based flow control may force the sender to suspend the (continuous) data stream like video
  ➢ Retransmitted data might be received too late to be consumed in time
  ➢ Traditional mechanisms do not scale to multiple-target communication like multicast scenarios - they are typically designed for point-to-point communication

Other possibilities:

• Partially Reliable Streams
  ➢ Number of retransmitted packets are limited and calculated from the timing constraints of the call setup

• Forward Error Correction (FEC)
  ➢ Additional information is added to the data
  → low end-to-end delay and no exclusive buffering. FEC works only for error detection/correction within packets but not for packet loss

• Priority Channel Coding
  ➢ Multiple data streams are distinguished by different priorities, during congestion periods the network is more likely to discard low-priority packets. For example, I-frames in MPEG-2 are transmitted with high, B- and P-frames with low priority

Partially reliable streams lacks the possibility of introducing a discontinuity or large end-to-end delays with large buffers → preventive schemes like FEC or priority channel coding should be used

Resource Monitoring

Resource Monitoring is an important part of resource management in networks and at end-points

• Monitoring in networks adds overhead
  ➢ Most of the monitoring variables should be optional and monitoring should be able to turned on/off

• Monitoring in end-systems includes supervisor function to continuously monitor the processed QoS parameters (e.g. controlling a compression component)
  ➢ design and implementation is non-trivial

• Resource Administration Protocols provide communication about resources between resource managers at intermediate nodes and end-systems and can be implemented as
  ➢ part of network management protocols like CMIS/CMIP (Common Management Information Service and Protocols) and SNMP (Simple Network Management Protocol)
  ➢ part of the transmission protocol like ST-II or like RCAP

Resource Adaptation

• Dynamically changing the network capacity of each session needs support of dynamic change of QoS parameters

• Resource adaptation requires
  ➢ notification and renegotiation of QoS parameters
  ➢ adaptive resource schemes

• Renegotiation request can come from either
  ➢ the user (e.g. change QoS like video rate) who invokes the resource administration protocol to check the resource if network adaptation is required
  ➢ the host system (e.g. due to workstation overload) which invokes e.g. a notification of QoS degradation to a misbehaved user/application. Network QoS renegotiation may also be invoked by the resource administration protocol
  ➢ the network (due to overload or congestion) which invokes a notification to the host reporting that the allocation of resources must change
Resource Adaptation - Network Adaptation

Proper balancing of network load is desirable and necessary to
- increase network availability
- reclaim resources by network administrators
- reduce the impact of unscheduled, run-time maintenance on clients with guaranteed services

Implementing load balancing policy requires
- routing, performance monitoring (detect load changes)
- dynamic re-routing
- load balancing control to make a decision to re-route a channel

Dynamic re-routing mechanism is done by establishing a new channel (shadow channel) via a new route, sending data on this new channel and tearing down the old channel after the maximum end-to-end delay.

Resource Adaptation - Source Adaptation

Adaptation of the source rate according to the current network resources by
- Evolution of the system state over the time: state of the complete route is used to compute the sending rate of the source
  - State information may be appended periodically to data packets and is sent back to the source. Intermediate nodes update the information if the switch service rate is lower than the proposed rate
  - Feedback is sent with separate control messages to the receiver and back to the source
- Rate control using network feedback: changes in the traffic conditions are detected by feedback from the network
  - Explicit: information about the traffic load or buffer space from the network
  - Implicit: information about packet losses and delay from acknowledgements
- Traffic shaping at source: smoothing traffic at the source
- Hierarchical coding: algorithms producing two or more types of cells describing same block of pixels with different degree of detail → codecs are more complex, use greater amount of bandwidth

Architectures

What have we learned – what changes to our systems are necessary?

**Endpoint architectures** need to incorporate components like
- Resource managers which include
  - Translation services (translate QoS)
  - Admission control (checks for availability of resources)
  - Resource reservation
  - Resource management (monitoring, adaptation, ...)
- Service agents
- Management Information Bases (MIBs) with QoS specification

**Router/Switch architectures** need to employ
- Resource managers which include
  - Packet classifier
  - Packet scheduler
  - Admission control
- Network management
- Management Information Bases

Architectures - Example

- Lancaster’s QoS-Architecture incorporates notions of flow, service contract and flow management over high-performance ATM-based networks
- Heidelberg Transport System (HeiTS) provides continuous media exchange with QoS guarantees (delay, jitter, throughput, reliability) based on ST-II, resource management and real-time mechanisms via multicast
- Tenet Protocol Suite with RCAP (Real-Time Channel Administration Protocol), RTIP (Teal-Time IP), RMTP (Real-Time Message Transport Protocol) and CMTP (Continuous Media Transport Protocol) provides QoS negotiation, resource administration and real-time media transport
- OSI architecture provides QoS in network layer and enhancements in transport layer. It considers QoS specification and negotiation in transport protocols
- RSVP, based on IP, provides resource reservation
- UPenn’s end-point architecture (OMEGA Architecture) provides QoS handling and management at applications and transport subsystems
- Native Mode ATM Protocol provides ATM network QoS guarantees
Today, there are some efforts to bring QoS into the network layer, using the presented principles like RSVP, scheduling, …

IPv6
- Allows for treating data streams, maybe assigned with QoS requirements
- But: will it ever come?

Integrated Services
- Use of RSVP for setting up routes with guaranteed QoS
- But: not usable for higher number of data flows; scalability problems!

Differentiated Services
- Aggregation of data streams with similar characteristics to be scalable
- But: details of the approach are still not worked out

MPLS
- Adding flow labels to protocols like IP to establish paths through the network
- But: also not full developed in the details

→ Still a long way to really introduce QoS into the networks!