Chapter 3: Multimedia Systems – Communication Aspects and Services

3.2: Quality of Service (QoS) and Resource Management

- Quality of Service Parameters
- Admission Control
- Traffic Shaping
- Scheduling
- QoS Architectures for the Internet

Multimedia Requirements

- Streaming stored multimedia
  - Media stored at source
  - Transmitted to client
  - Streaming: client playout begins before all data has arrived to avoid long waiting times
  - Timing constraint for still-to-be transmitted data: in time for playout (low jitter)

- Interactive, Real-Time Multimedia
  - E.g. IP telephony (Voice over IP, VoIP), video conference
  - End-to-end delay requirements:
    - E.g. audio: < 150 msec good, < 400 msec OK
    - Includes application-level and network delays
    - Higher delays noticeable, impair interactivity

Multimedia and Networks

- End-to-end delay is limited by the speed of light but also by the intermediate network nodes (routers)
- Real-time communication demands low end-to-end delays – typically less than 200 msec – and low jitter
- Multimedia transmissions have also a maximum loss tolerance (depending on encoding, only loss of a certain fraction of all packets can be tolerated)
- Data rate of multimedia may change due to the encoding mechanism

• Multimedia requires changes in the "usual" network!
Quality of Service

<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>high</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
yellow: 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

Keeping the QoS

Apply resource management: regard a flow of packets as stream, guarantee the enforcement of quality requirements for the stream as a whole.

- **Over-provisioning:** increased router capacities, buffers and line capacity – but this does not give real guarantees...
- **Scaling:** graceful degradation of the quality, e.g. skip of more B-frames in a MPEG stream, or reduce color depth/frame rate (maybe adaptation to changes in available network capacity)
- **Buffering:** temporary store the data within the receiver. Thus the delay is increased, but the jitter is lowered (audio/video on demand)

To achieve real guarantees, different actions are necessary:

- **QoS Negotiation:** reserve and allocate resources (end-to-end) during communication establishment
- **Traffic Shaping:** prepare a data stream on sender side in a way that it does not flood the network
- **Packet Scheduling:** modify routers in the network to differentiate between different streams; priority is given to multimedia data streams

Buffering

- In streaming, data can arrive with variable rate by network delay and jitter
- Thus: client-side buffering for playout delay to compensate these problems:

  ![Buffering Diagram](https://via.placeholder.com/150)

- But: how long to buffer the data?

  - Packet departures source: 1 2 3 4 5 6 7 8
  - Packet arrives at buffer: 1 2 3 4 5 6 7 8
  - Packet removed from buffer: Time in buffer: 1 2 3 4 5 6 7 8

  ![Buffering Delay vs. Loss Rate Diagram](https://via.placeholder.com/150)

  - Tradeoff for $p$:
    - large $p$: less packet loss
    - small $p$: better interactive experience

Buffering Delay vs. Loss Rate

Receiver attempts to play out each packet exactly $p$ msecs after it was generated

- Packet has timestamp $t$:
  - play out packet at $t + p$
- Packet arrives after $t + p'$:
  - data arrives too late for playout, data “lost”

Tradeoff for $p$:

- large $p$: less packet loss
- small $p$: better interactive experience
Strategies for Guaranteeing QoS

QoS Negotiation and Admission Control

- Can the user requirement be accepted?
- Should reservations be done?

Traffic Shaping

- How to avoid that the network becomes overloaded by single users?
- Scheduling Strategies in the routers: How to handle different connections such that deadlines are satisfied?

QoS Negotiation and Resource Admission Control

RSVP is not a routing protocol, but only an “addition” to such one. A description of the requirements of the receiver in form of a flow specification (Level of the QoS) is needed. Categories of flow specification are:

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

RSVP can be used for Unicast and for Multicast

RSVP Procedure:

1. 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.)
2. The receiver sends back RSVP Reservation Messages along the collected path, which contain its flow specification. Every router on the path reserves resources accordingly.
3. If a reservation is not possible with a router, an error message is sent back.
4. As soon as the RSVP Reservation Message reaches the sender, the complete path is reserved, the sender begins to send its data.
5. To delete outdated reservations, timeouts are defined. If the receiver does not refresh its reservation before expiration of the timer, it is deleted.

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)
- But: peak traffic is not considered. Data can be lost, even if sufficient capacity is available in the network.

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 ($b < \beta$):
- There are $B$ ($0 \leq B \leq \beta$) tokens in the bucket at the time we want to send our packet
- 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
  - Delay Earliest Due Date (D-EDD)
  - Jitter Earliest Due Date (J-EDD)
  - Rate Controlled Static Priority
- Differences to simple scheduling strategies:
  - More complex queueing
  - Often time-stamped
  - More complex computation e.g. of deadlines
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. “Weighted fair queueing”
   - Fair queueing gives equal fraction of bandwidth per connection (1/n of total bandwidth if n connections are active)
   - But in some cases we would like to give more bandwidth to connections with high data rate, thus
     - 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” by some strategy to the connections with higher need

Scheduling - Weighted Fair Queueing

Proof of delay bound:

\[ D_i \leq \frac{\beta_i}{g_i} + \frac{(h - 1) \cdot l_i}{g_i} + \sum_{m=1}^{n} \frac{l_m}{r_m} \]

Proven by:

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

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

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

\[
D_1 \leq \frac{\beta_1 + (h_1 - 1) \cdot l_1}{g_1} + 3 \cdot \frac{l_1}{r_1} = 10.0166 \text{ ms}
\]

\[
D_2 \leq \frac{\beta_2 + (h_2 - 1) \cdot l_2}{g_2} + 2 \cdot \frac{l_2}{r_1} = 20.01395 \text{ ms}
\]

dominating term (in this example!)
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 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

QoS in the Internet: Integrated Services

- 1995-1997 standardized by the IETF as an architecture for the transmission of multimedia streams
- Integrated Services (IntServ) uses RSVP for signaling of flow specifications as well as reservations and provides QoS for each data flow individually
- Thus the principle adds a connection-orientation to IP
**Integrated Services**

- **Classes of QoS:**
  - **Guaranteed:** Data rate, delay and reliability are guaranteed. Deviations do not occur, packets are not discarded. Suitable for intolerant real-time applications.
  - **Controlled Load:** “weak” guarantees, small deviations are possible. Principle: for data streams of this class the network behaves as best effort for an unloaded network. No guarantees are given, suitable for tolerant real-time applications.
  - **Best Effort:** as good as possible, normal Internet traffic.

- **Problems:**
  - **Scalability:** for each data flow a router has to maintain own flow specifications
  - How can authorization and priority be treated for a reservation request?
  - The QoS classes are not sufficient to differentiate reasonably between different types of data streams.
  - Possible: use IntServ only “at the edge” of large networks, where only few data flows are present

---

**QoS in the Internet: Differentiated Services**

- Class-based approach (Differentiated Services, DiffServ): do without guarantees, only manage aggregated data streams
- Thus, the complexity in routers is shifted “at the edge” of the network, internal network routers can be kept simpler.
- Divide the network into domains. A domain is a part of the entire network, which supports DiffServ. It consists of access routers for the domain (Ingress routers, yellow) and internal routers (core routers, blue).
- The domain defines service classes, each data flow is assigned to such a class. With coming into the domain (at a Ingress Router), each packet is assigned a class, and in the network the forwarding bases on the class parameters (per-hop behavior). Looking only at the aggregated data flows, a better scalability as for IntServ is achieved.

---

**Application of DiffServ with IP**

- Use of the Type of Service field in IPv4 for the classification (DSCP – Differentiated Service Code Point). The DSCP value defines the per-hop behavior of the packet from one router to the next one.

```
+-----------------+-----------------+-----------------+-----------------+
| Source Address  | Time to Live    | Protocol        | Header Checksum |
| Destination Address | DSCP | free | P | F | F |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

The code point defines the transmission class, which informs a router, how it has to treat the packet in forwarding.
Service Classes

For DiffServ, amongst others the following classes (for specifying transmission behavior) are defined at the moment:

- **Default Forwarding** realizes the usual Best Effort transmission
- **Expedited Forwarding** tries to emulate a rented line
- **Assured Forwarding** uses priorities and discarding probabilities used when an overload of a router is given

Expedited Forwarding

Idea with Expedited Forwarding:
- there are “regular” and “expedited” packets. Expedited packets are passed on in such a way, as if there would be no or only little further traffic. A minimum data transmission rate is guaranteed.
- Routers can manage two separated queues for these packet types (Weighted Fair Queuing).
- Possible: low loss rate/delay/jitter as well as a guaranteed data transmission rate

Assured Forwarding

Improves differentiation: Definition of four priority classes with own resources (at the moment; there also is room for more classes)
- For each class, three probabilities for discarding a packet are defined: low, medium, high
- Thus: altogether 12 different service classes
- Principle:
  - The priority class determines the portion of the transmission capacity of the routers
  - During high load packets of lower priority would be discarded completely
  - Fairness: packets of each priority class should have chances to survive
  - Therefore definition of the probabilities for each class: by suitable selection of the probabilities, a small part of the lowest priority level still is still forwarded, while packets of higher priority classes are already discarded.

1. Classification of the packets in service classes
2. Appropriate choice of DSCP tags
3. Bring the data streams in a form according to their flow specification. Exceeding the specifications leads to discarding data. If thereafter still too much data are present, discarding of packets in accordance with their probability
4. Weighted Fair Queuing in accordance to priority classes
Current Trends in the Internet

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!