

# Multimedia Content Management (3)

---

Prof. Ralf Möller, TUHH, STS

## ■ Previous lecture:

- Requirements
- Generic MMCMS system architecture

## ■ Contents today:

- Distributed multimedia systems, streaming
- Quality of service (QoS) principles

# Chapter 4: Selected Components

---

- The story so far:

- Part 1 – Overview

- Part 2 – Use cases and requirements

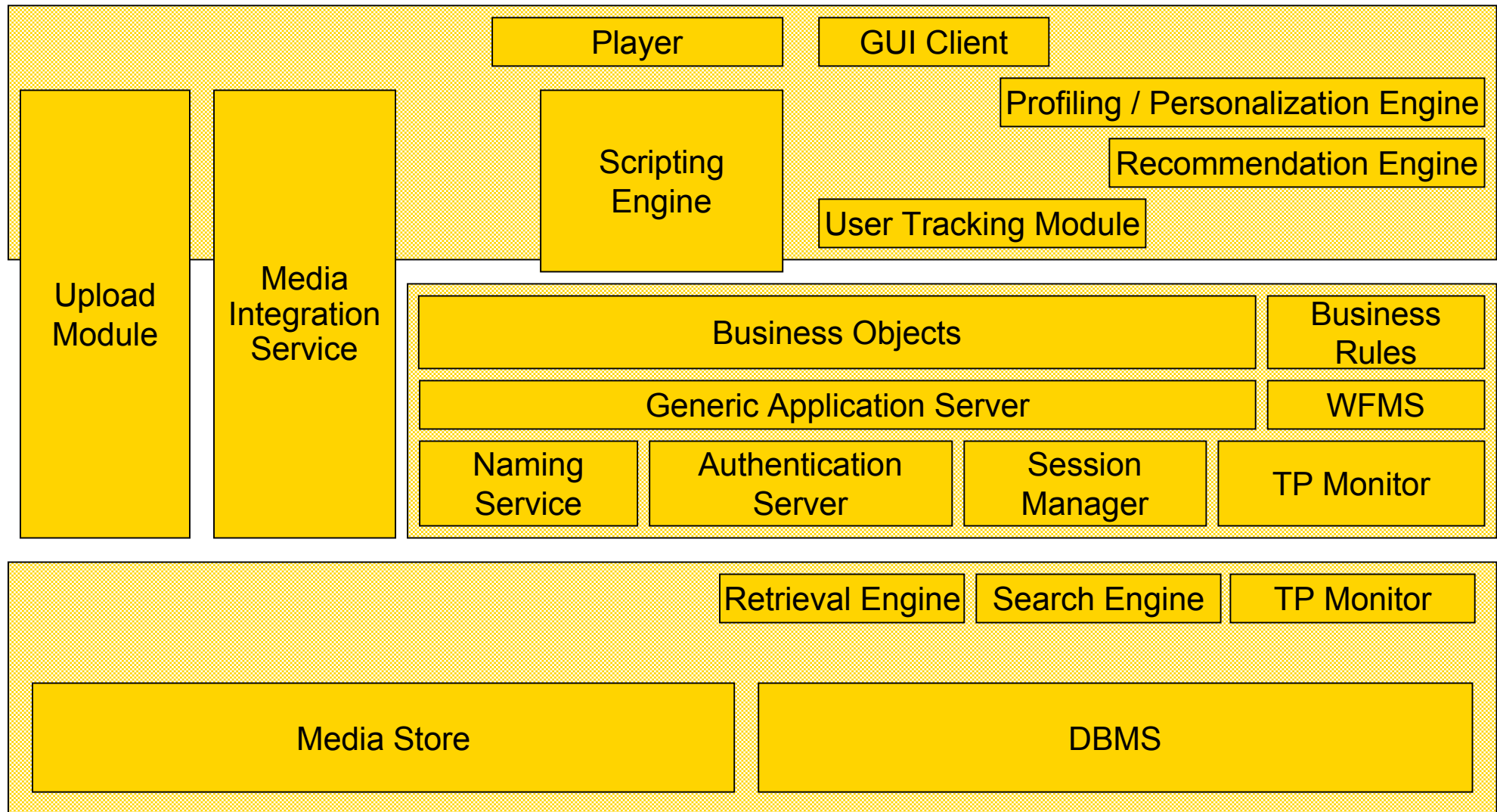
- Part 3 – Architecture

- Now we take a closer look at some components of a MMCMS identified in the architecture:

- Part 4 – Selected components

- Today: Media Server, Media Integration Service, Streaming Media Player

# MMCMS Architecture Diagram



# WFMS

---

- Two definitions by the Workflow Management Coalition (WfMC):

Definition: *Workflow*

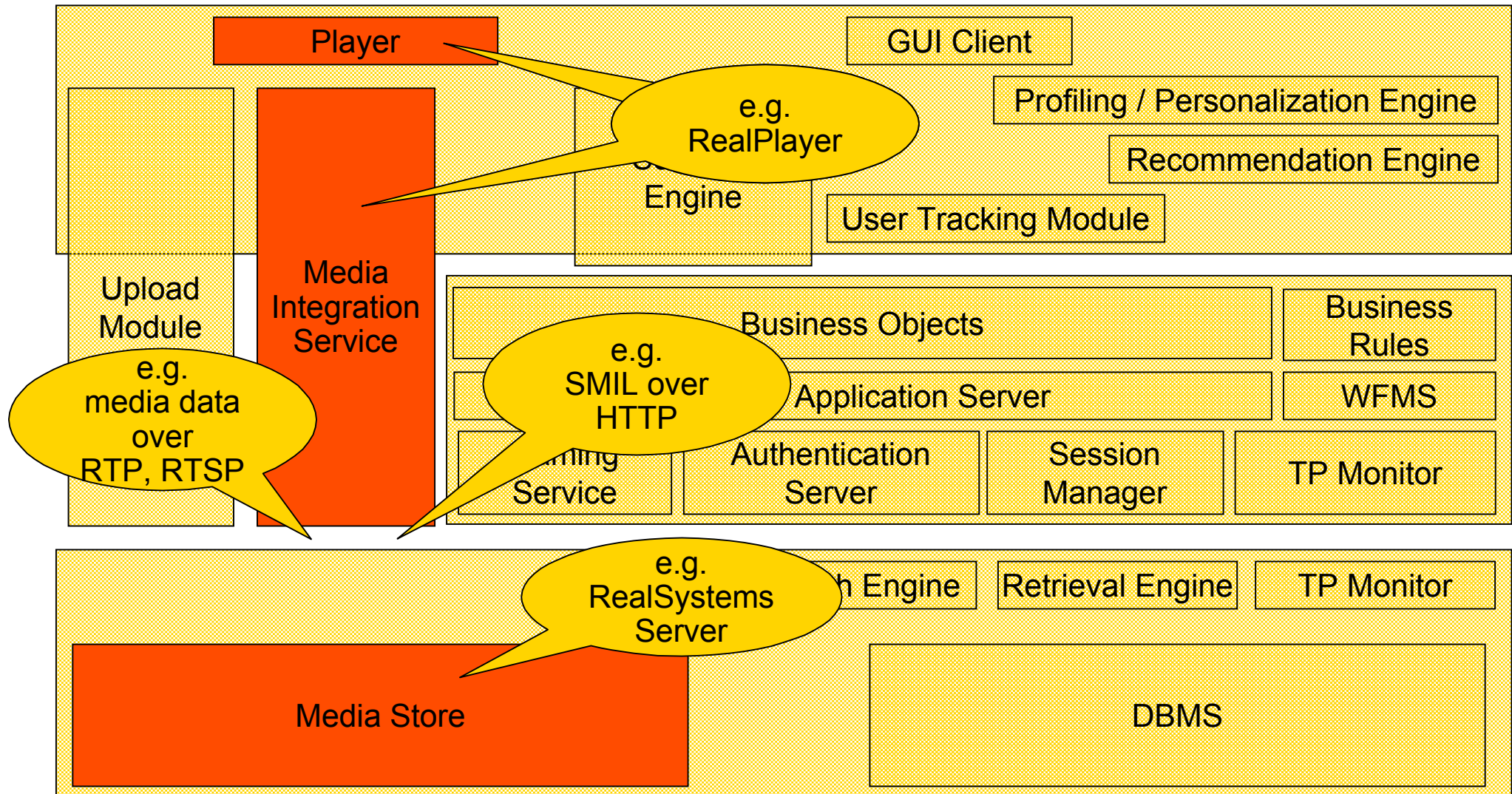
The computerized facilitation or automation of a business process, in whole or in part

Definition: *Workflow Management System*

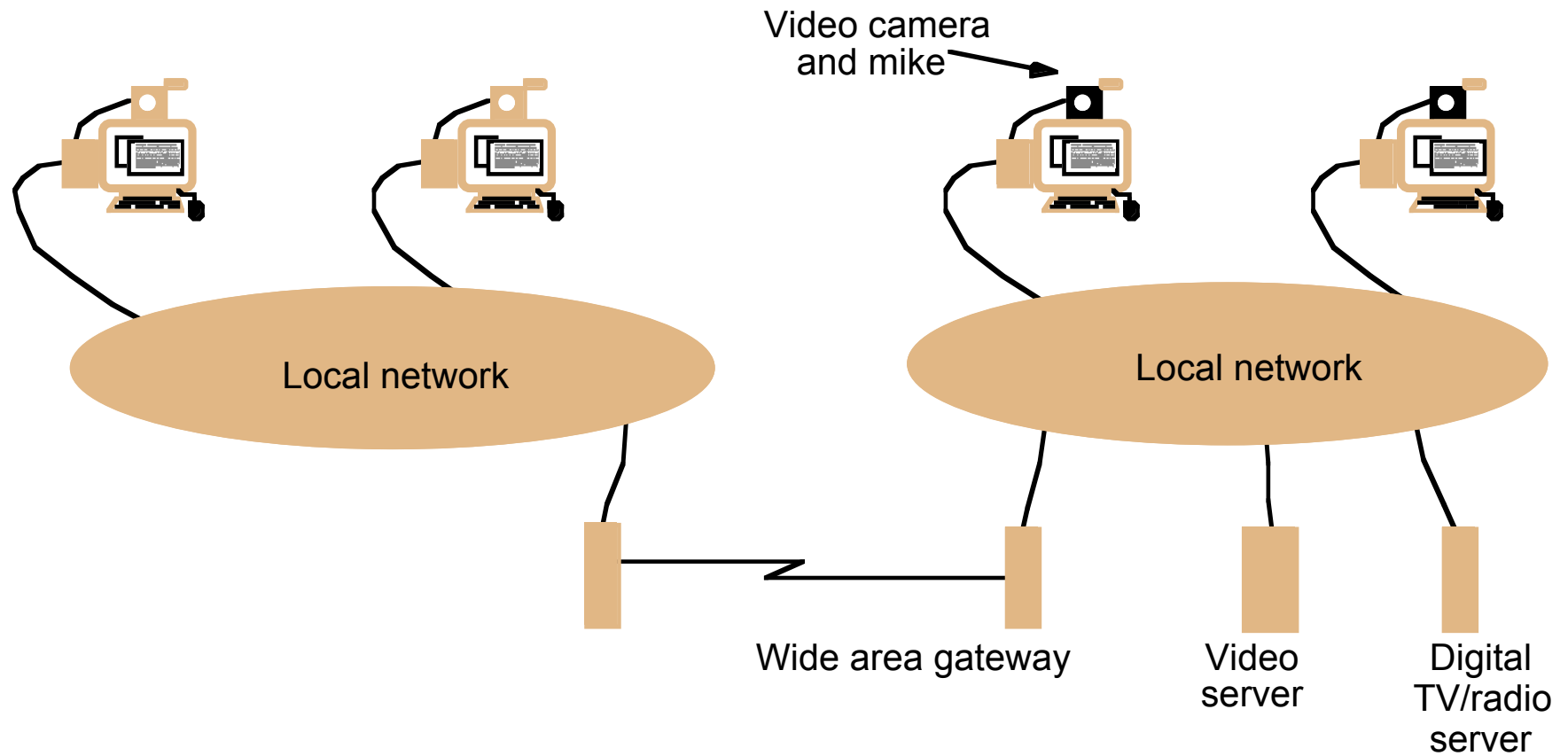
A system that completely defines, manages and executes „workflows“ through the execution of software whose order of execution is driven by a computer representation of the workflow logic

- Functional areas:
  - Build time: defining and modeling workflow processes and constituent activities
  - Run-time: control & management, sequencing enactment of constituent activities
  - Run-time user interaction: managing work lists (“to do items”)

# Streaming Media Components, Products & Protocols



# A Distributed Multimedia System



# Characteristics of Multimedia Systems:

---

- Generation and consumption of continuous data streams in real time
- Huge amounts of audio, video, and other time-based data elements
- In-time arrival of data elements (audio samples and video frames) and their processing is essential
- Late data elements are worthless
- Specification/negotiation of required quality of service (QoS)

# Real Time Systems: Comparison of Characteristics

---

- Flight surveillance, telecommunication, production processes
  - Small amounts of data
  - Hard requirements (deadlines)
  - Worst-case requirements must be met
- Multimedia systems
  - Large amounts of data, heavily distributed
  - Weaker requirements
  - Dynamic requirements
  - QoS specifications for services

# Requirements

---

- Short latencies in order to ensure interactivity
- Synchronization of different media (internally)
- External synchronization (e.g., with display devices)

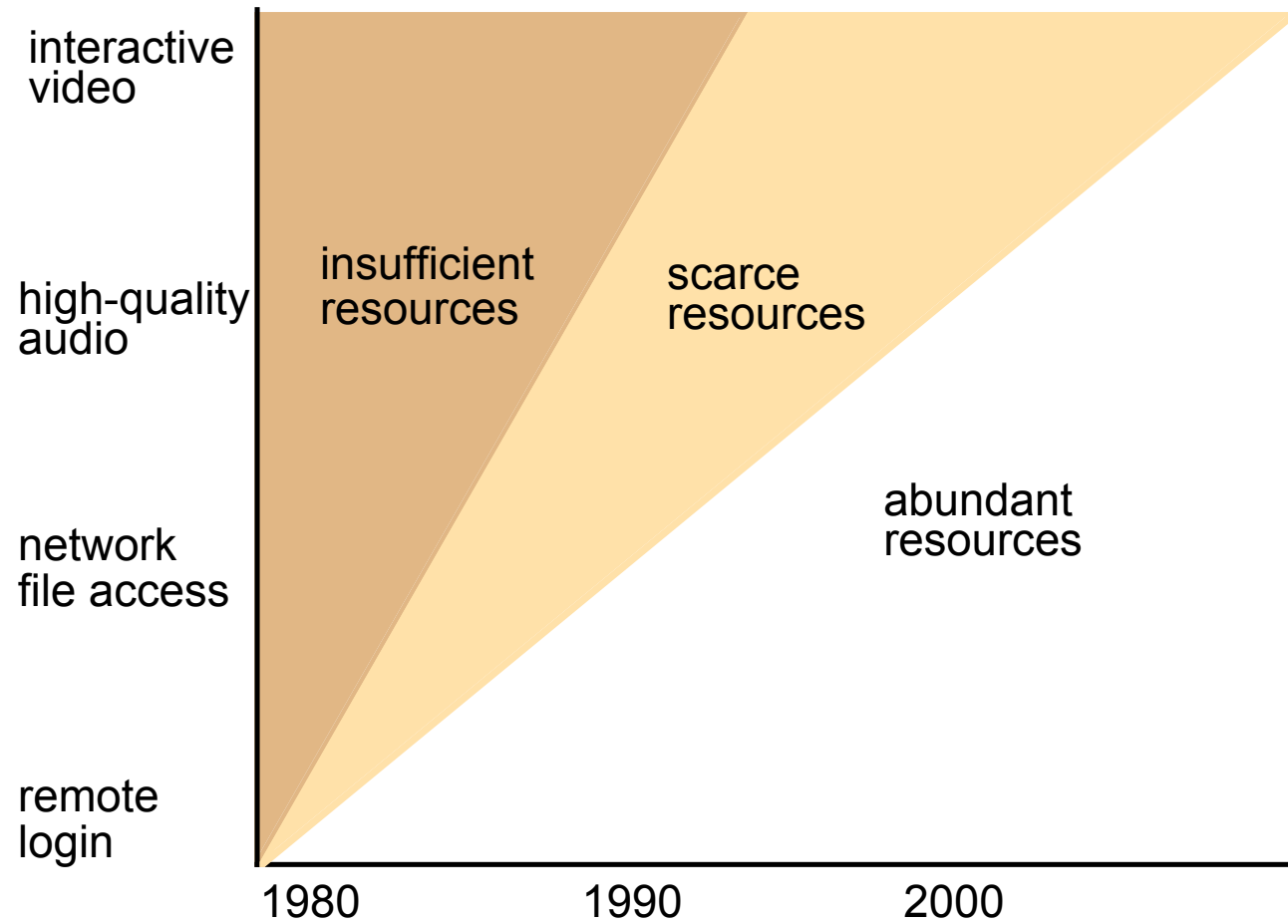
# Today: Best-Effort-Principle

---

- Web-based multimedia applications
- Telephone/video conferences
- Video-on-demand services

# Window of Scarcity for Resources

---



# Characteristics of Typical Multimedia Streams

---

	<i>Data rate (approximate)</i>	<i>Sample or frame size                      frequency</i>	
Telephone speech	64 kbps	8 bits	8000/sec
CD-quality sound	1.4 Mbps	16 bits	44,000/sec
Standard TV video (uncompressed)	120 Mbps	up to 640 x 480 pixels x 16 bits	24/sec
Standard TV video (MPEG-1 compressed)	1.5 Mbps	variable	24/sec
HDTV video (uncompressed)	1000–3000 Mbps	up to 1920 x 1080 pixels x 24 bits	24–60/sec
HDTV video (MPEG-2 compressed)	10–30 Mbps	variable	24–60/sec

---

# Capabilities of a Media Streaming System

---

- Streaming Server (Media Store):  
Deliver data in a format suitable for streaming, i.e. the data can be decoded while the download is in progress (e.g. MPEG, real audio, real video).
- Enable real time broadcasting of media data.
- Offer services based on protocols for delivery of real time data.
- Media Integration Service:  
Synchronize different streams to integrate them into one consistent presentation.

# Properties of Streaming Formats (1)

---

## ■ Streaming format

- Usually used for time-dependent media
- Data can be decoded in real-time during reception.
  - Presentation quality is limited by bandwidth
  - Compression of the original multimedia data.
- Data comes in small pieces which can be decoded independently from other pieces.
  - Loss of data does not compromise the transmission as a whole.
  - Ability to skip back and forth

# Properties of Streaming Formats (2)

---

## ■ Advantages:

### ■ Reduced latency

- | presentation can start before download is complete

### ■ Reduced space and transmission requirements

- | client does not need the whole media object locally
- | only transmits the part of the media object that is actually presented

## ■ Disadvantages:

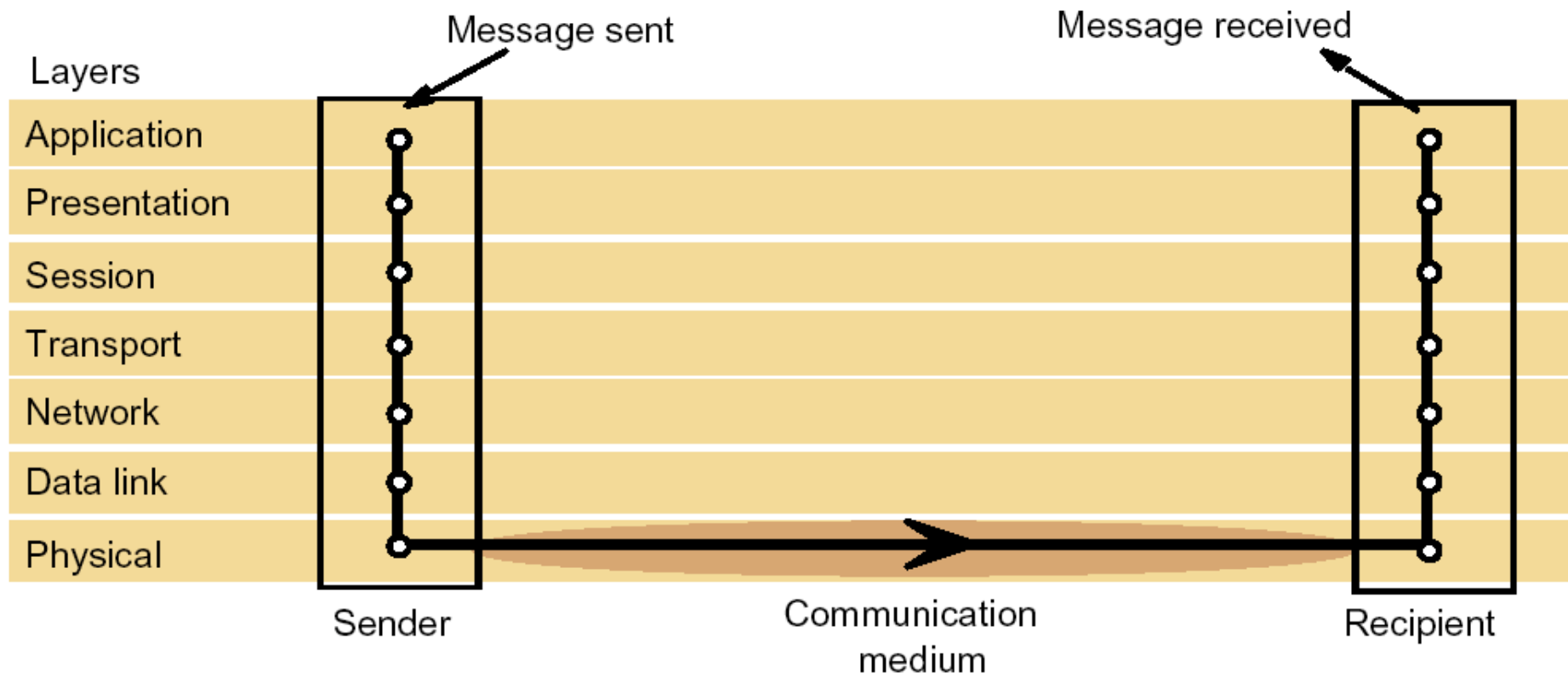
### ■ Increased latency for control (play, pause, fast forward)

### ■ Presentation quality is limited by bandwidth

### ■ Requires connection to server (online only)

# Current Network Layer Technology

## ■ IPv4 is the Standard



# Protocols for Real-Time Data

---

- Real Time Transport Protocol (RTP / RTCP)
  - RTP: open data packet protocol for real time data. – RFC 1889
  - RTCP: control information (resend lost packets etc.) – RFC 2326
- Real Data Transport (RDT)
  - Proprietary data package protocol of Real Networks.
- Real Time Streaming Protocol (RTSP)
  - Controls the streaming process  
(stop, play, fast forward, jump to time index)
  - Enables exchange of descriptions about the media data
  - Used in conjunction with RTP / RTCP or RDT
- Allows switching between different compression levels  
(bit rate)
  - But no real QoS management!

# Streaming Server: Source Data

---

- Base data for streaming presentations is taken from normal digital recordings (WAV, AVI).
- The volume of these base data is too high for real-time transmission over networks.
- Compression necessary:
  - Reduce amount of data to enable real time delivery.
  - Preserve quality of original data as far as possible.

# Streaming Server: Preprocessing

---

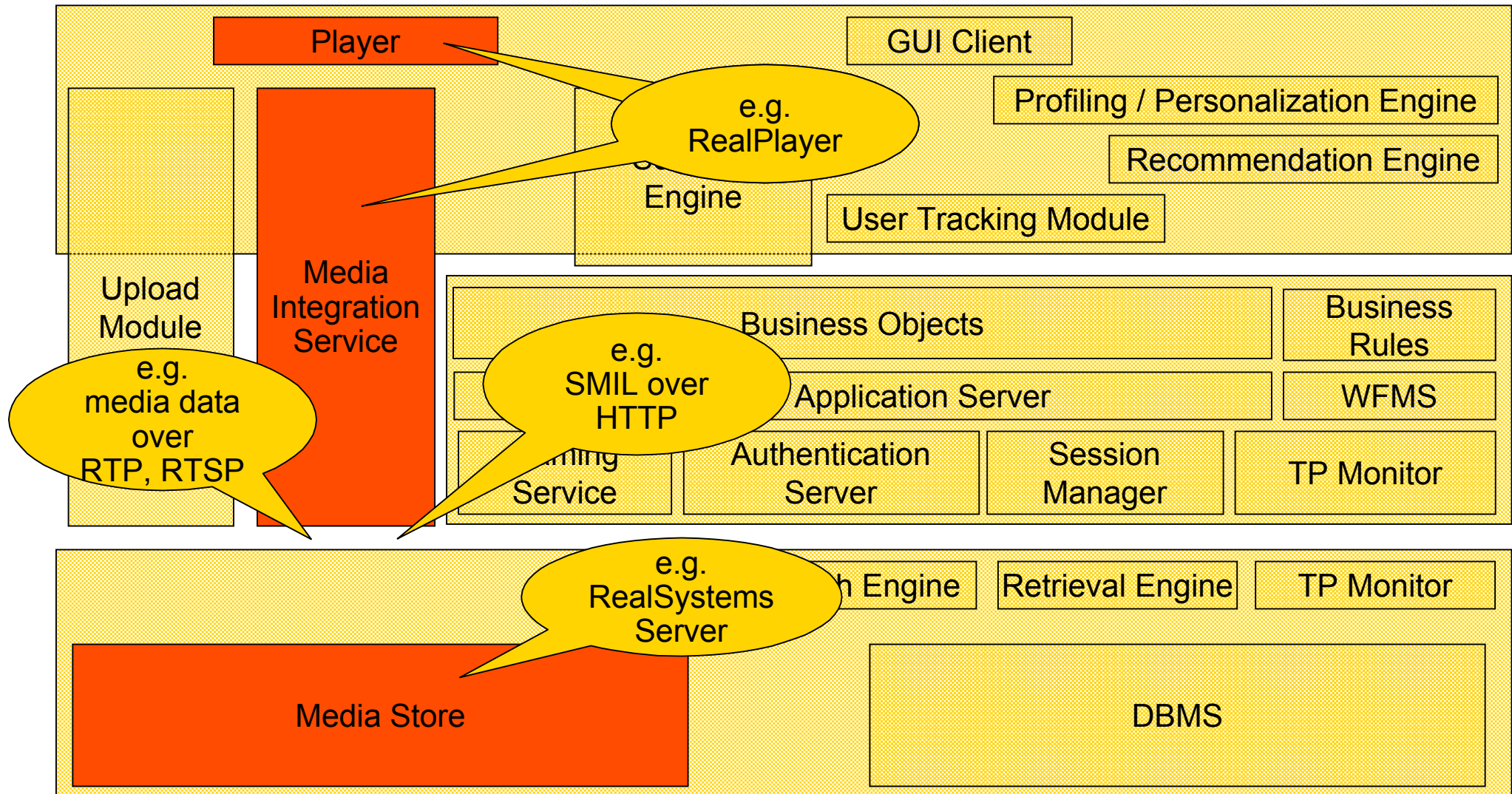
- Choose a set of bandwidths available for transmission over networks
  - e.g. 56 kbps, 64 kbps, 1 Mbps
- Encode data according to each chosen bandwidth
  - Results in alternative representations of the same audio content
  - Consider protocol overhead at all levels – available bit rate can be as low as 60% of hardware connection bandwidth
- Remove “less important” data (e.g. very high or very low frequencies in audio data) and compress the remaining data.
  - E.g. Mpeg audio layer 3 (“mp3”) uses psychoacoustic modeling: remove data that is hardly noticed by humans
  - -> more details about data formats and compression algorithms in a separate session

# Streaming Server: Playout

---

- Streaming server stores encoded data or
- Provides links to online encoding facilities (e.g., for live broadcasting)
- Deliver data to clients using real time enabled protocols (RTSP, RTP, RDT).
- Switch to different encodings on client request (e.g. in case of network congestion)
  
- Additional options for efficient storage access:
  - File allocation on disk optimized for continuous read
    - Requires operating system support / customized OS
  - special hardware
    - e.g. RAID = Redundant Array of Inexpensive (or: Independent) Disks

# Streaming Media Components, Products & Protocols



# Synchronization

---

## ■ Problem:

- A single multimedia presentation can consist of data from many sources (different files or servers).
- The different sources have to be synchronized during presentation to the user.
- Requires a language to specify when to show what (and where)

## ■ Example: A movie containing moving pictures, a soundtrack and subtitles.

## ■ Solution:

- Synchronized Multimedia Integration Language (SMIL, pronounced “smile”)

# Features of SMIL

---

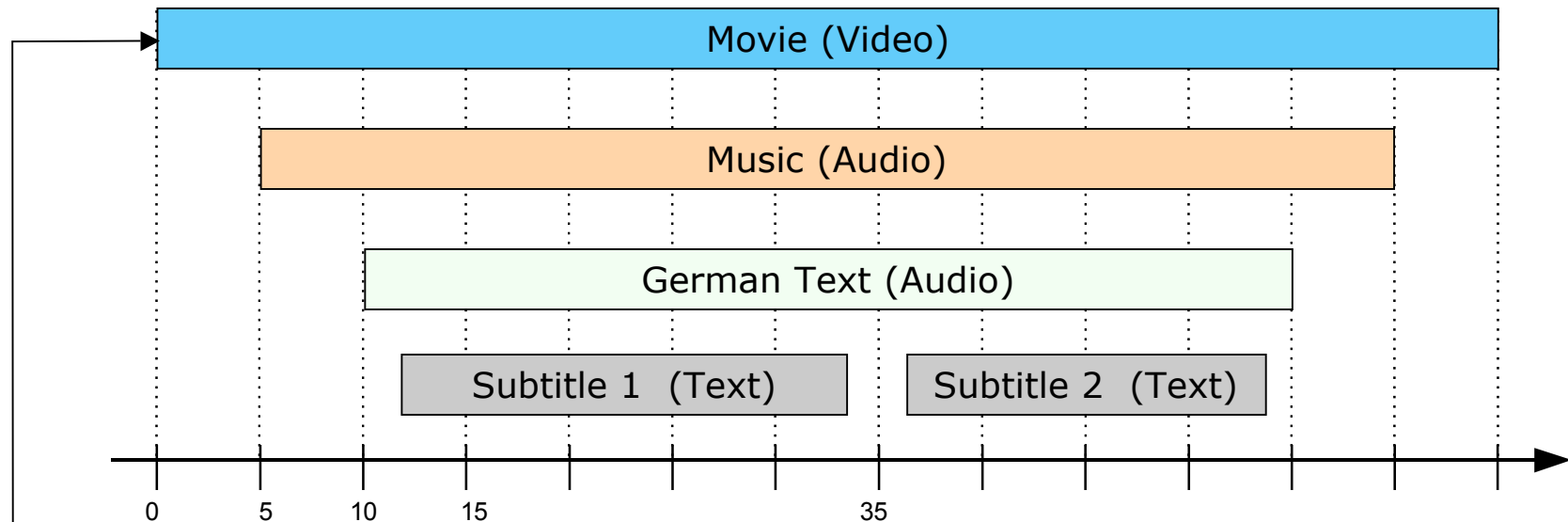
- W3C standard: <http://www.w3.org/TR/REC-smil/>
  - Specify location of different media sources.
  - Group sources (sequential, parallel).
  - Define the layout of the presentation.
  - Modify/synchronize the timing of the sources

- Example:

```
<smil>
  <body>
    <par>
      <video src="videos/movie.rm" />
      <audio src="soundtracks/music.rm" begin="5s"/>
      <audio id="germ" src="rtsp://example.overdub.de/german.rm" begin="10s"/>
      <seq>
        <textstream src="rtsp://example.overdub.de/subtitles1.rt" begin="id(germ)(2s)"/>
        <textstream src="rtsp://example.overdub.de/subtitles2.rt" begin="4s"/>
      </seq>
    </par>
  </body>
</smil>
```

# Synchronization of Different Media with SMIL

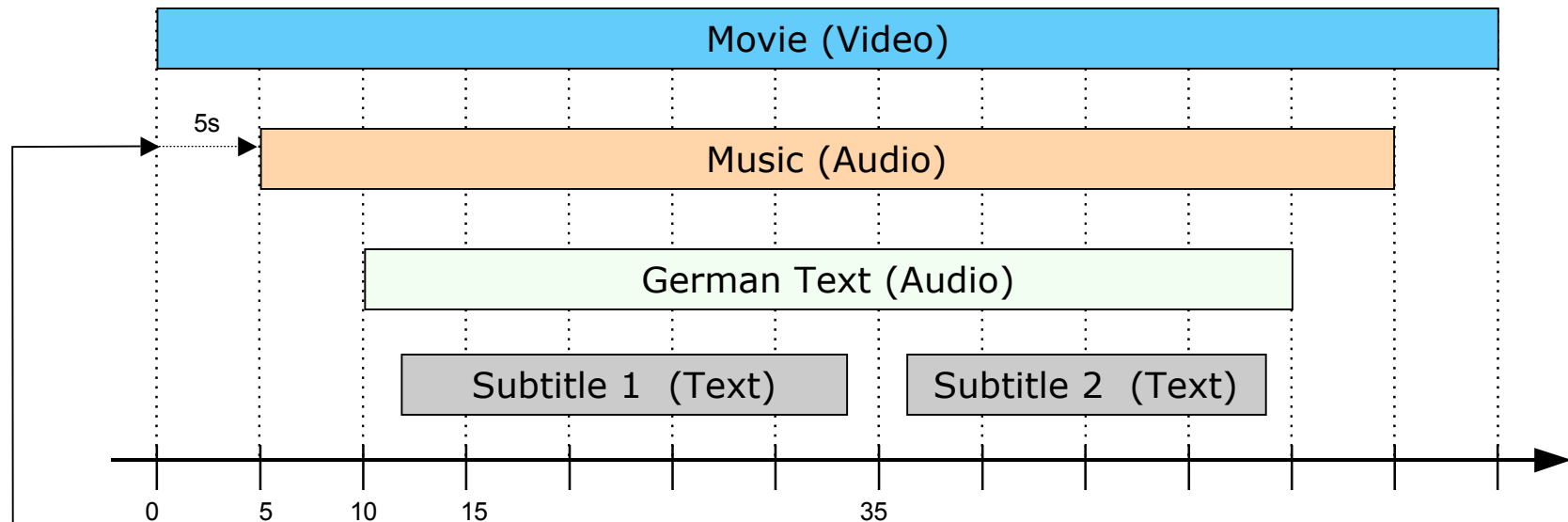
## Different Media Bound to a Common Time Line



```
<video src="videos/movie.rm" />
```

# Synchronization of Different Media with SMIL

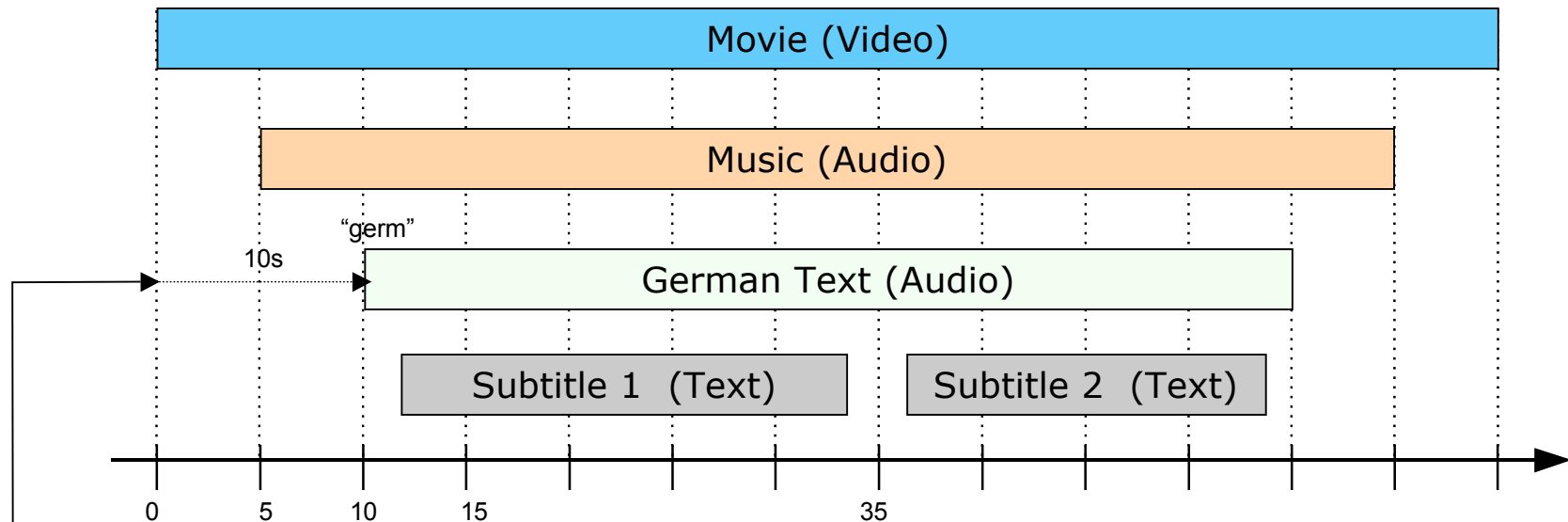
## Different Media Bound to a Common Time Line



```
<audio src="soundtracks/music.rm" begin="5s"/>
```

# Synchronization of Different Media with SMIL

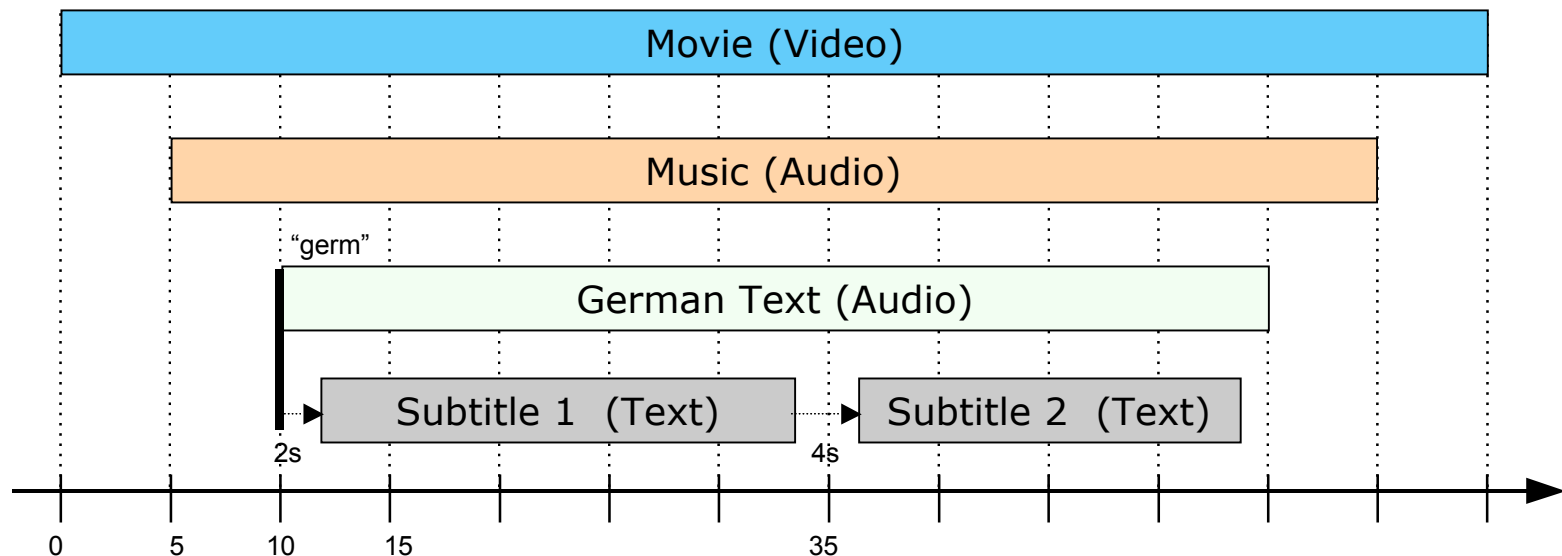
## Different Media Bound to a Common Time Line



```
<audio id="germ" src="rtsp://example.overdub.de/german.rm" begin="10s"/>
```

# Synchronization of Different Media with SMIL

## Different Media Bound to a Common Time Line



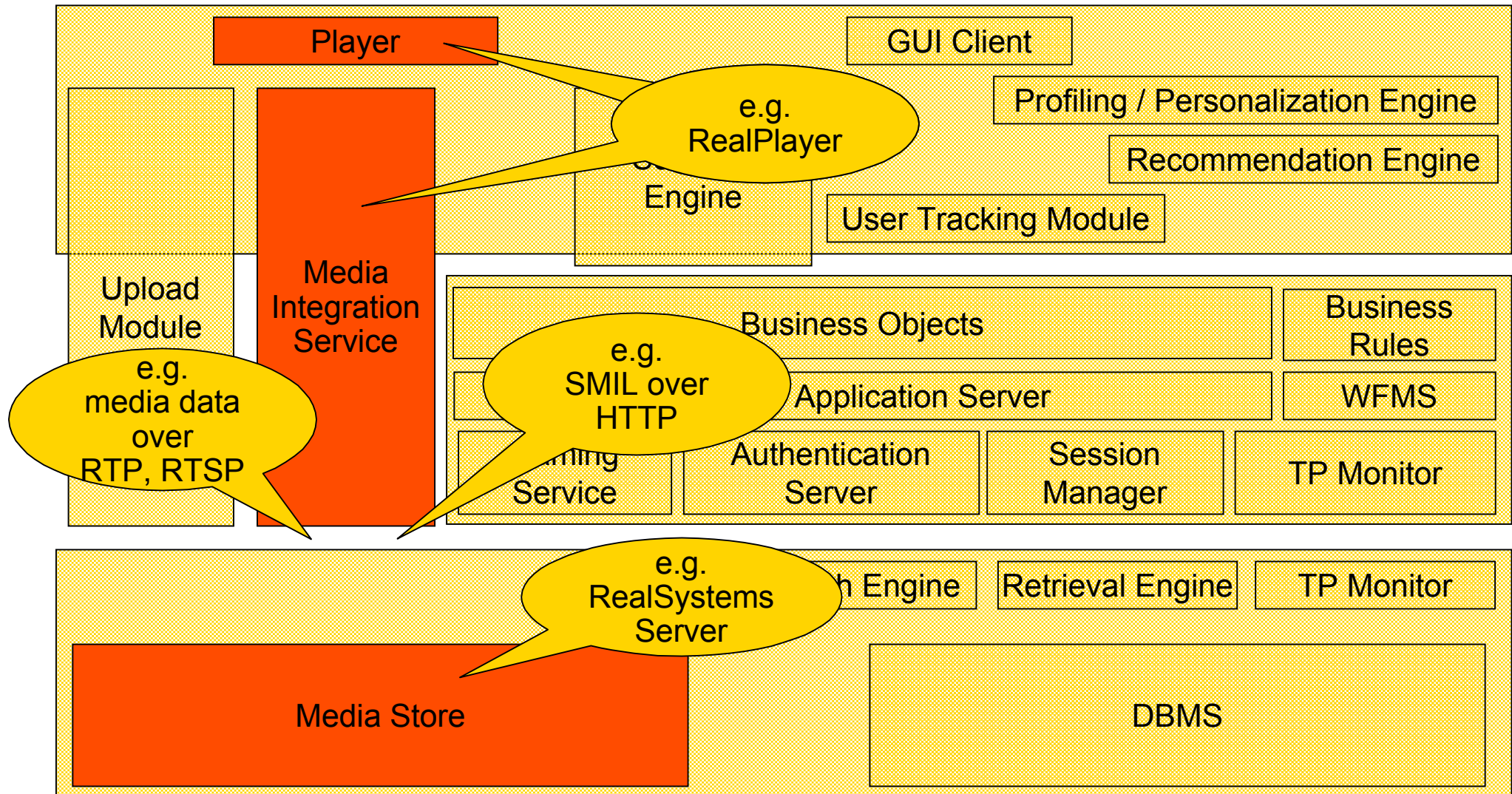
```
<seq>  
  <textstream src="rtsp://example.overdub.de/subtitles1.rt" begin="id(germ)(2s)"/>  
  <textstream src="rtsp://example.overdub.de/subtitles2.rt" begin="4s"/>  
</seq>
```

# Typical client/server interaction

---

1. Client (e.g. Real Player) request metadata (SMIL file).
2. Server (e.g. Web Server) delivers metadata using HTTP.
3. Client request media data mentioned in the SMIL file from the different sources using RTSP.
4. One or many media servers deliver the media data using RTP or RDT.
5. Client assembles the different streams according to the SMIL data to one single presentation.
6. Client controls playback via RTSP using the synchronization data from the SMIL file.

# Streaming Media Components, Products & Protocols



## Presentation – Client's Player

---

- Use player which can process data while the “download” is still in progress (e.g. Real Player)
- Player observes network bandwidth and chooses appropriate stream quality
- Player processes layout and synchronization data (SMIL) and sends corresponding requests (RTSP) to the media servers.

# Product overview – Real Networks

---

## ■ RealSystem Producer:

- | Encoding software for creation of streaming formats from various sources.

## ■ RealSystem Server:

- | Media Store implementation which
  - | can respond to RTSP requests,
  - | can change connection properties dynamically during transmission

## ■ RealPlayer:

- | Player and Media Integration Service implementation which can
  - | receive streams in various formats
  - | process metadata encoded with SMIL
  - | combine streams from many sources to one presentation

# QoS-Management-Systems

---

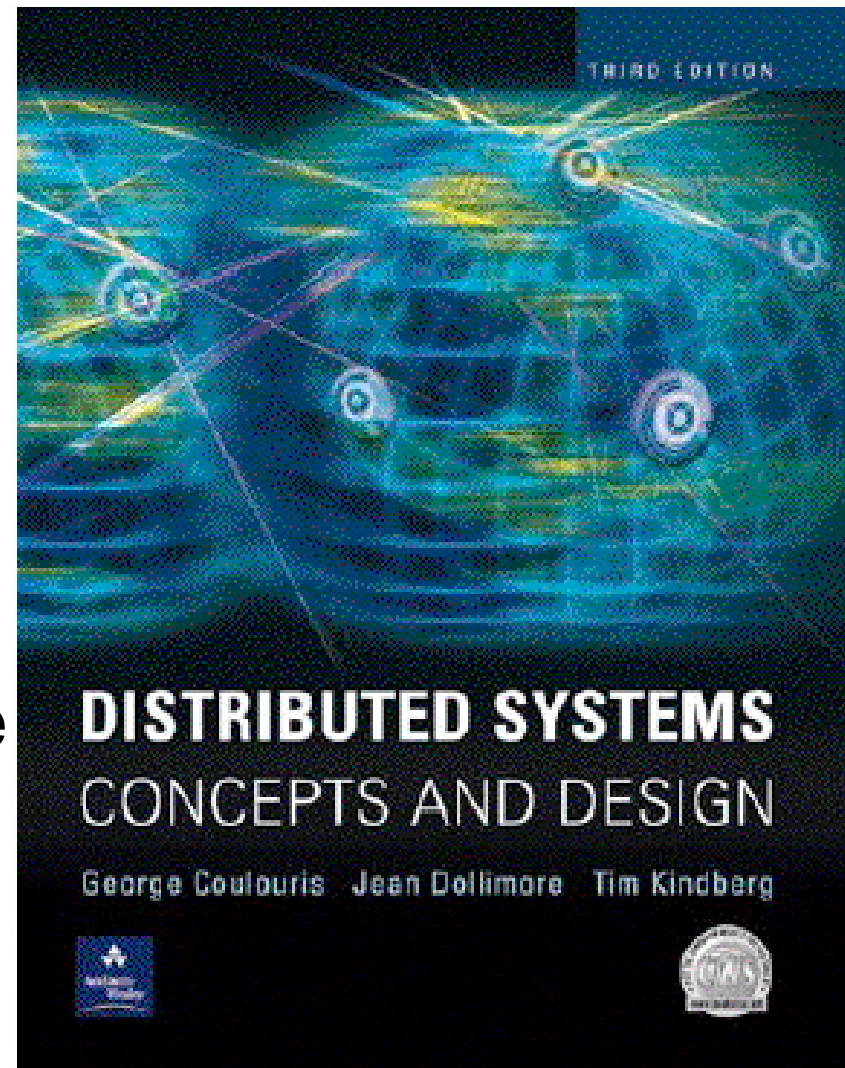
- Management of computation and communication resource and their allocation
- "Resource bandwidth"
  - Network bandwidth, memory bandwidth, processor cycles, memory or buffer capacities
- Openness (open distributed multimedia system)
  - Applications can be started without prior agreement
  - Dynamic service requests
- Negotiation of QoS guarantees

# Literature

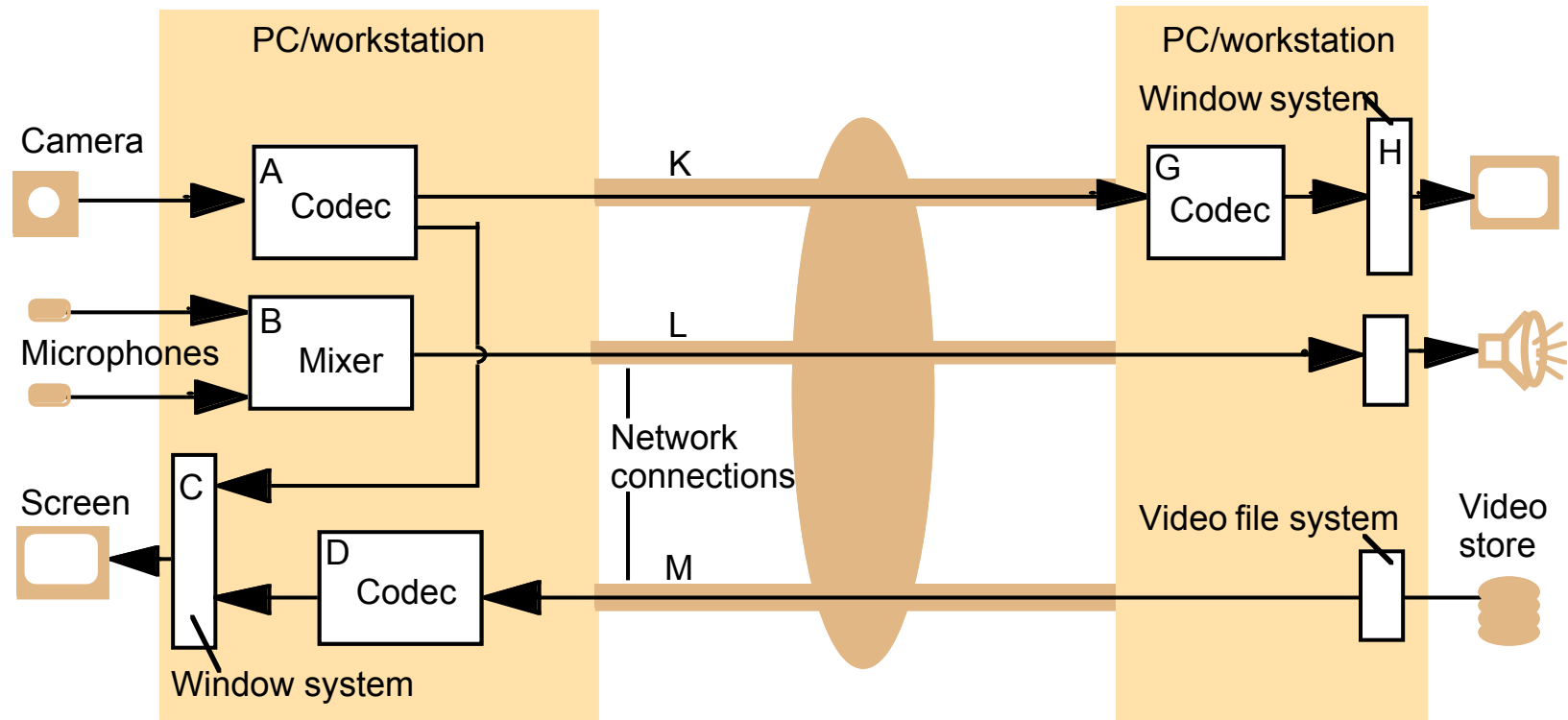
---

Chapter 15 from:

Many graphics are taken from this book



# Typical Components in Multimedia Infrastructures



—▶ : multimedia stream  
White boxes represent media processing components, many of which are implemented in software, including:  
codec: coding/decoding filter  
mixer: sound-mixing component

# QoS Specifications for Components

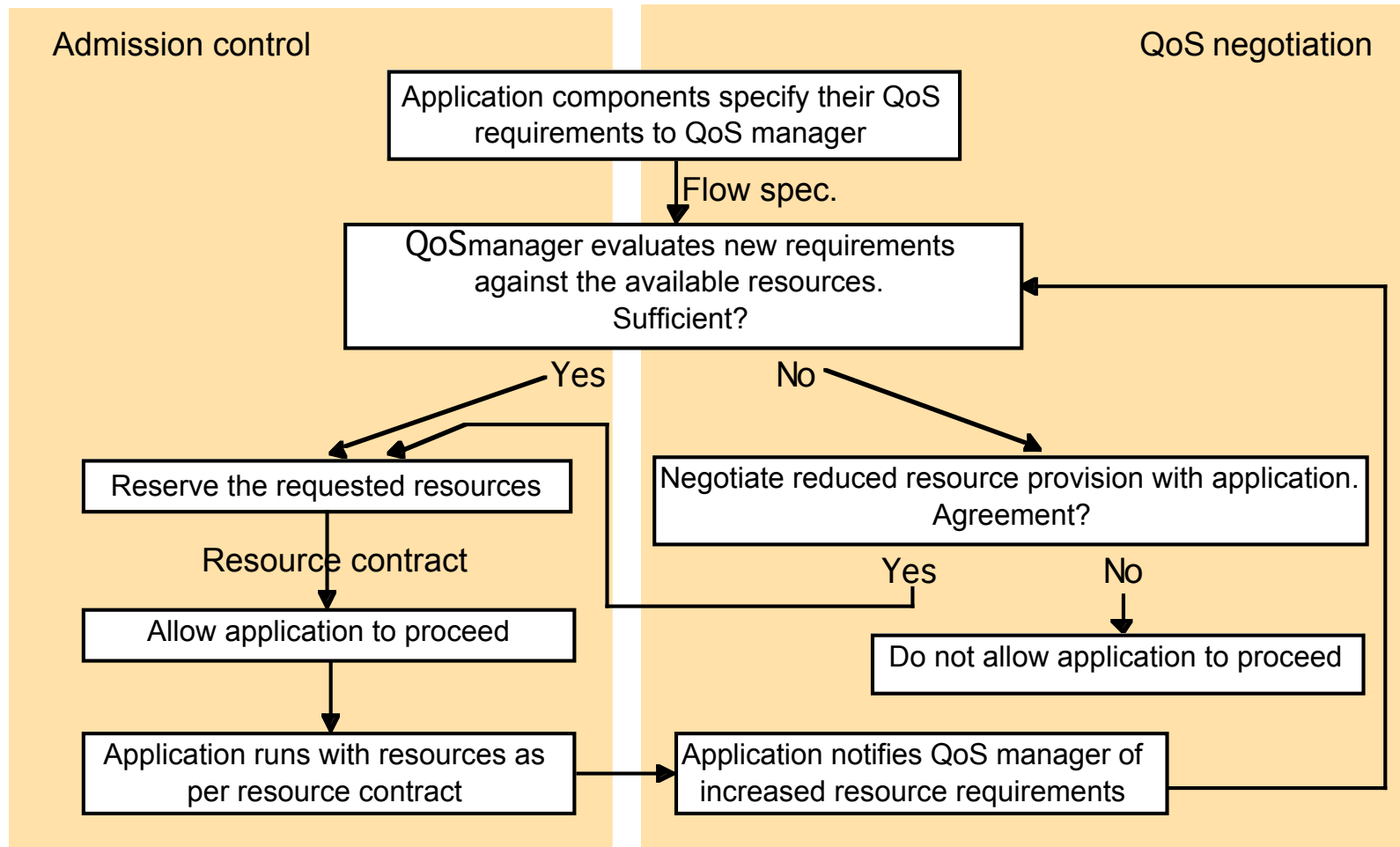
<i>Component</i>	<i>Bandwidth</i>	<i>Latency</i>	<i>Loss rate</i>	<i>Resources required</i>
Camera	Out: 10 frames/sec, raw video 640x480x16 bits		Zero	
A Codec	In: 10 frames/sec, raw video Out: MPEG-1 stream	Interactive	Low	10 ms CPU each 100 ms; 10 Mbytes RAM
B Mixer	In: 2 44 kbps audio Out: 1 44 kbps audio	Interactive	Very low	1 ms CPU each 100 ms; 1 Mbytes RAM
H Window system	In: various Out: 50 frame/sec framebuffer	Interactive	Low	5 ms CPU each 100 ms; 5 Mbytes RAM
K Network connection	In/Out: MPEG-1 stream, approx. 1.5 Mbps	Interactive	Low	1.5 Mbps, low-loss stream protocol
L Network connection	In/Out: Audio 44 kbps	Interactive	Very low	44 kbps, very low-loss stream protocol

# Quality of Service Parameters

---

- QoS negotiation (possibly per “media channel”)
  - Bandwidth
  - Latency (jitter, first derivative)
  - Loss rate (z.B. 1%)
    - Less important: bit errors
    - More important: buffer overflows, late arrivals
  - Required computational resources (CPU cycles)
- QoS admission control

# Tasks of a QoS Manager



# Specification of QoS Parameters

---

## ■ Bandwidth

### ■ "Peaks" / Traffic Patterns

#### ■ Example : 3 Streams with 1Mbps:

- Stream with 1Mbit frame per second
- Stream with computer-generated animation (asynchronous) with 1Mbps
- Stream with 100 bit sound sample per microsecond

### ■ Burstiness:

- Maximum number of media elements that can come too early
- Maximum number of messages per time interval  $t$ :  $P = Rt + B$   
R = frame rate, B = burst

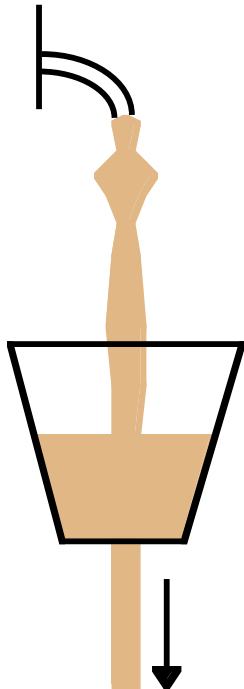
## ■ Latency

## ■ Loss rate

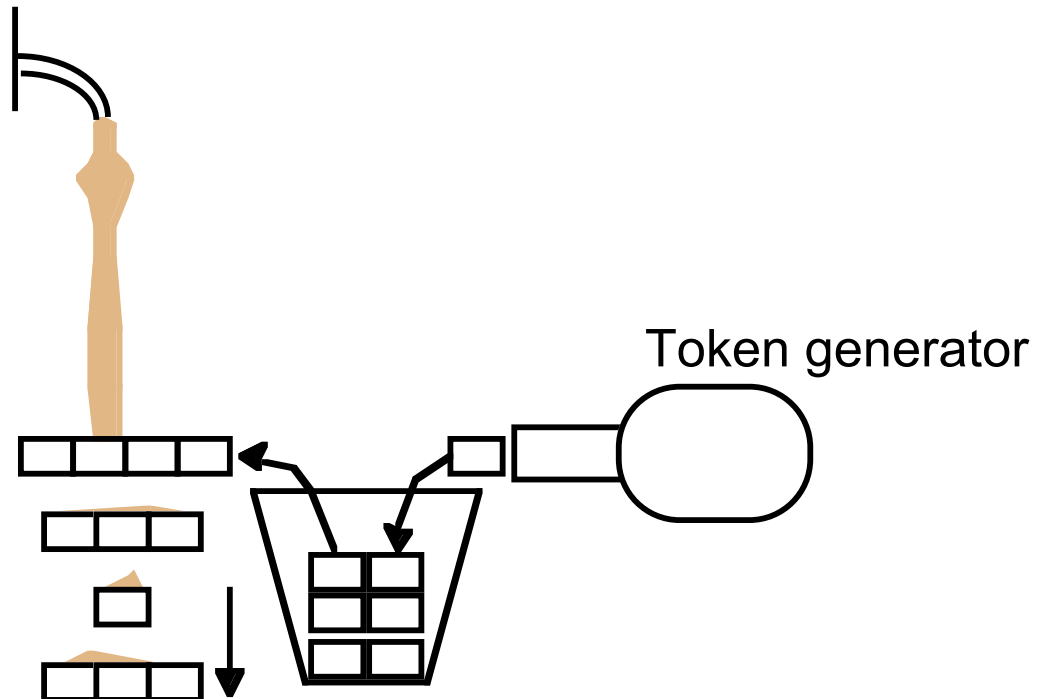
# Traffic Shaping Algorithms

- Use of buffers to reduce the effects of peaks

(a) Leaky bucket



(b) Token bucket



- Token bucket: maximal  $Rt + B$  data in system

# Flow Specifications

---

## ■ Collection of QoS Parameters (e.g., RFC 1363)

	Protocol version
Bandwidth:	Maximum transmission unit
	Token bucket rate
	Token bucket size
	Maximum transmission rate
Delay:	Minimum delay noticed
	Maximum delay variation
Loss:	Loss sensitivity
	Burst loss sensitivity
	Loss interval
	Quality of guarantee

# Negotiation Procedures: Sender-Initiated (1)

---

## ■ Simple approach:

- Trace data flow through network from source to target
- Send flow specs from intermediate node to intermediate node
- Each intermediate node checks locally if requirements can be met...
- ... and propagates request
- Final results will be propagated back from the destination node (target) to the source node

## Negotiation Procedures: Sender-Initiated (2)

---

- Use min/max specs instead of absolute values
- But: parameters interact
  - Specify two, minimize third one
- In case of multiple targets propagate worst-case values
- Maybe in this case receiver-initiated QoS management is advantageous

# Admission Control

---

- Reservation of bandwidth
  - Aggregation of Max-Bandwidth specification leads to a waste of resources
- Statistical multiplexing with overbooking
  - Guarantees valid only with certain probability
  - Based on assumption that not all participants require max specs at the same time
  - Relies on the assumption that aggregate bandwidth remains constant even parts exhibit bursts (known to be wrong)

# Stream adaptation

---

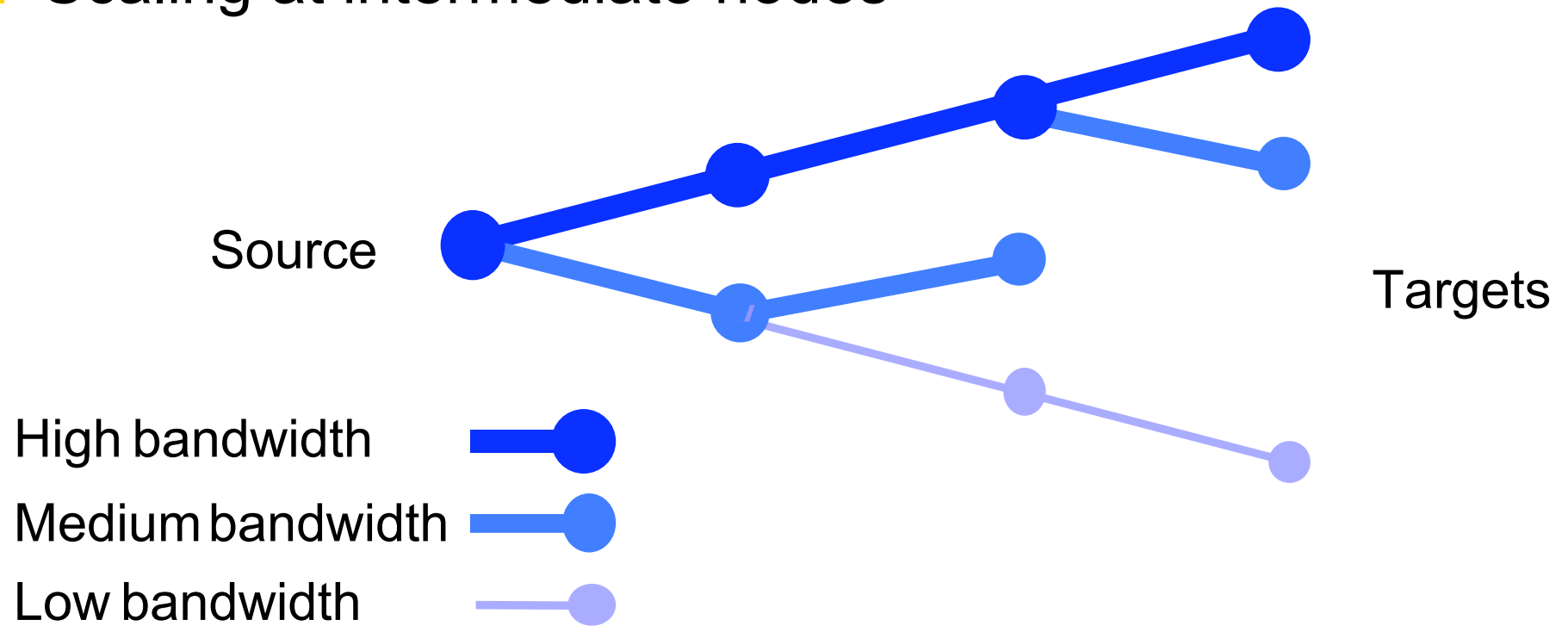
## ■ Scaling

- If a bottleneck occur in a network, reduce input at source
- Transmit less frames
- Mostly applied if data are directly generated
- More complex if stored data are just “played” out (decoding, processsing and reencoding required)
- Dimensions:
  - Temporal (reduce number of frame)
  - Spatial (reduce number of pixels)
  - Frequency (reduction of quality through compression)
  - Amplitude / Color space (reduce number of entries in color space)

# Filtering

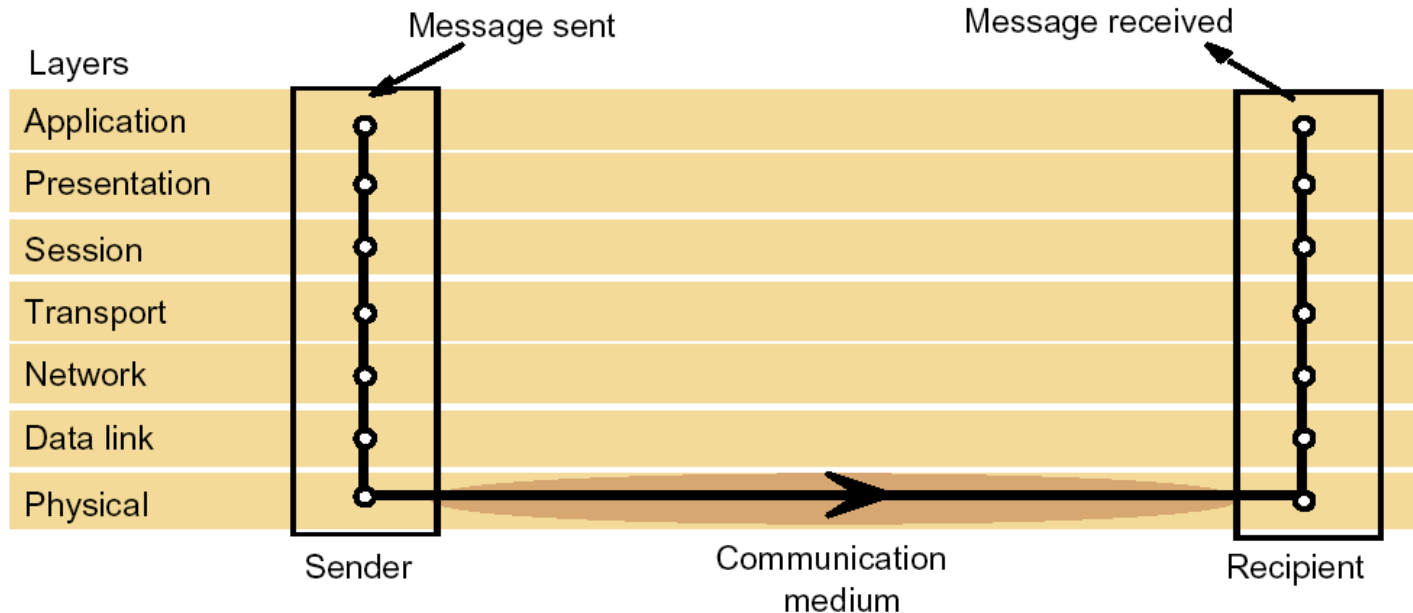
---

## ■ Scaling at intermediate nodes



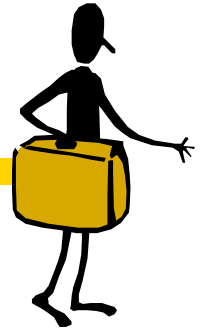
## ■ Encoding of information into substreams

# TCP/IPv6



- QoS built into network layer?
- Priority and flow labels
- Specify which packets are to be handled more rapidly and with more reliability than others
- Specify which packets can be dropped if they are too late (late arrival has no value)

# Summary: Streaming



- Resource Management
- QoS Management, negotiation, admission control
- Resource utilization planning
- Buffers and traffic shaping to avoid peaks

# What comes next?

