Internet Media-on-Demand: The Real-Time Streaming Protocol

Henning Schulzrinne
Dept. of Computer Science
Columbia University
New York, New York
schulzrinne@cs.columbia.edu

Overview

- Internet media-on-demand
  - why bother – I already have a TV and VCR
  - Internet integrated-services architecture
  - problems
- real-time stream protocol (RTSP) ➠ “Internet VCR”
- session description

Internet multimedia (on demand)

VOD trials not exactly successful. . . Internet MM different:

- just one service among many ➠ reverse economics from VOD
- re-use existing infrastructure
- flexible media: modem, wireless, cable, LAN, . . .
- quality scales from stamp-size flipbook to HDTV – adaptive
- side information easy (closed captioning)
- easy integration with WWW
- easy integration with recording – click-on-page-to-record
- security through encryption
- cheap authoring, service ➠ lots of content

Internet multimedia

Same infrastructure, different delivery modes:

on demand: unicast

near on-demand: staggered transmission on multicast ➠ VCR control

multicast: niche markets to audience of millions

Applications

- lectures, seminars
- on-demand instruction
- entertainment: specialty content
- remote digital editing
- voice mail

Internet radio

- 12,140 U.S. AM and FM radio stations, only 100 in Germany
- FM quality (56 kb/s) ➠ backbone capacity of 680 Mb/s
- New York City: 45 FM stations ➠ 2.5 Mb/s
- DirecTV: 31 audio channels ➠ 1.7 Mb/s
- easy time-shifting, content-labeling ➠ near media-on-demand

Problems

bandwidth: 64–128 kb/s for talking heads, 1.5 Mb/s for movies

quality: packet loss, predictability

reliability: makes CATV look good…

billing infrastructure: pay-per-view?

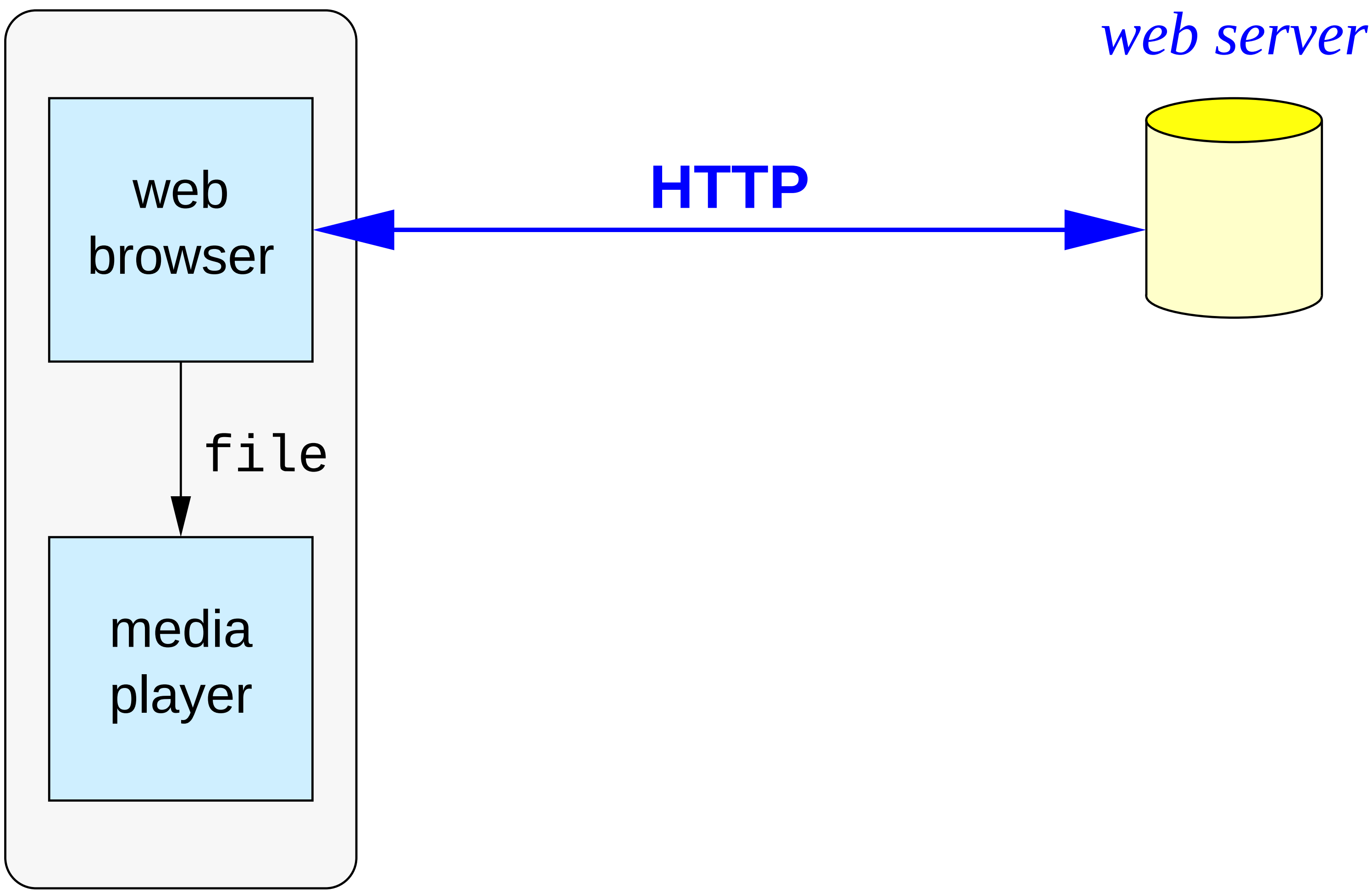
cheap receivers: shouldn't cost more than set-top box

Internet streaming media requirements

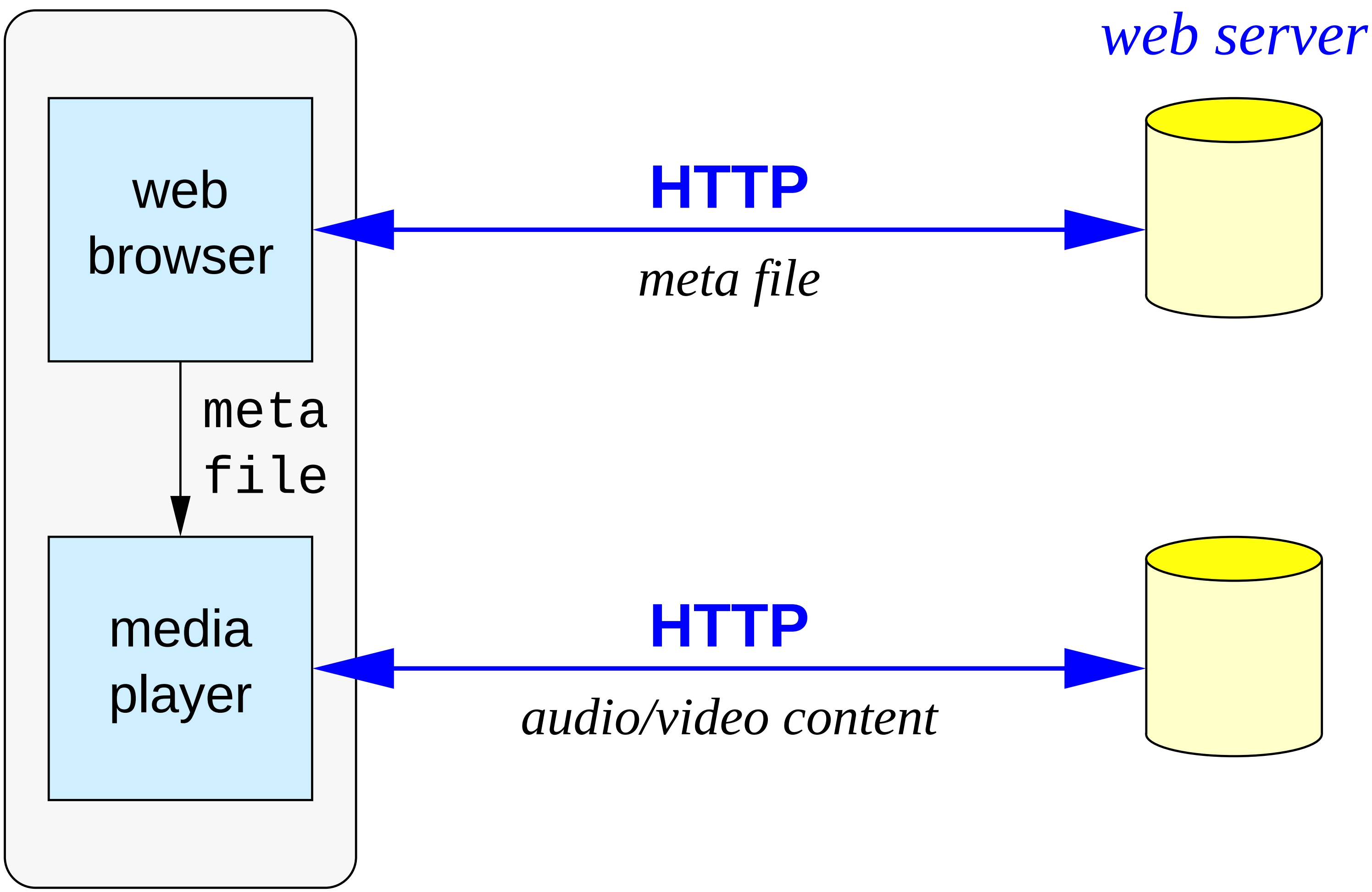
- retrieval of media from media server
  - video-on-demand ➠ unicast
  - near video-on-demand ➠ time-staggered multicast
- live events (Mbone-style) ➠ multicast
- remote digital editing ➠ queued play lists, recording
- remote device control
- integration with conferences
- transport, content, description-neutral

Have some proprietary protocols, need interoperability

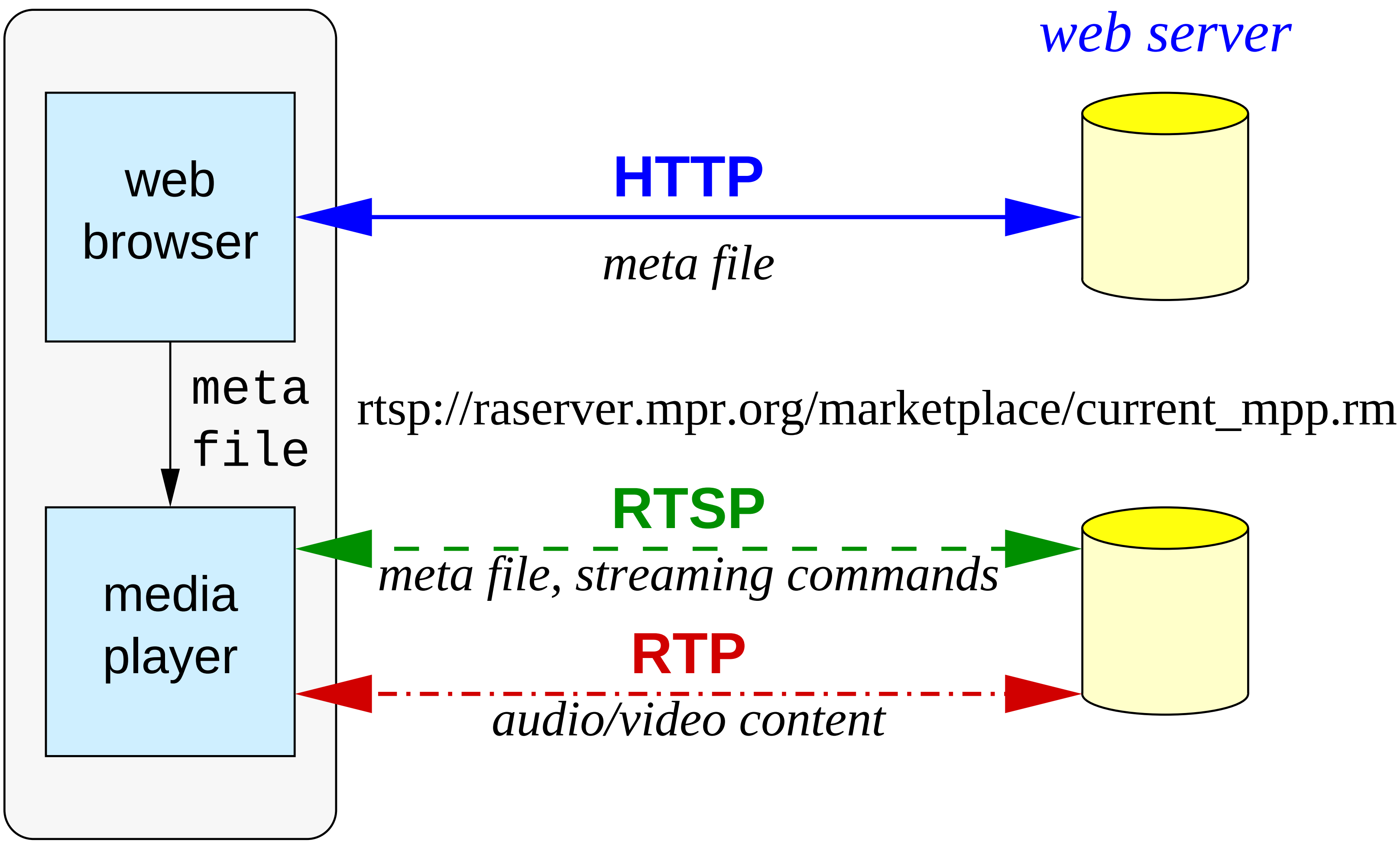
Streaming media: download



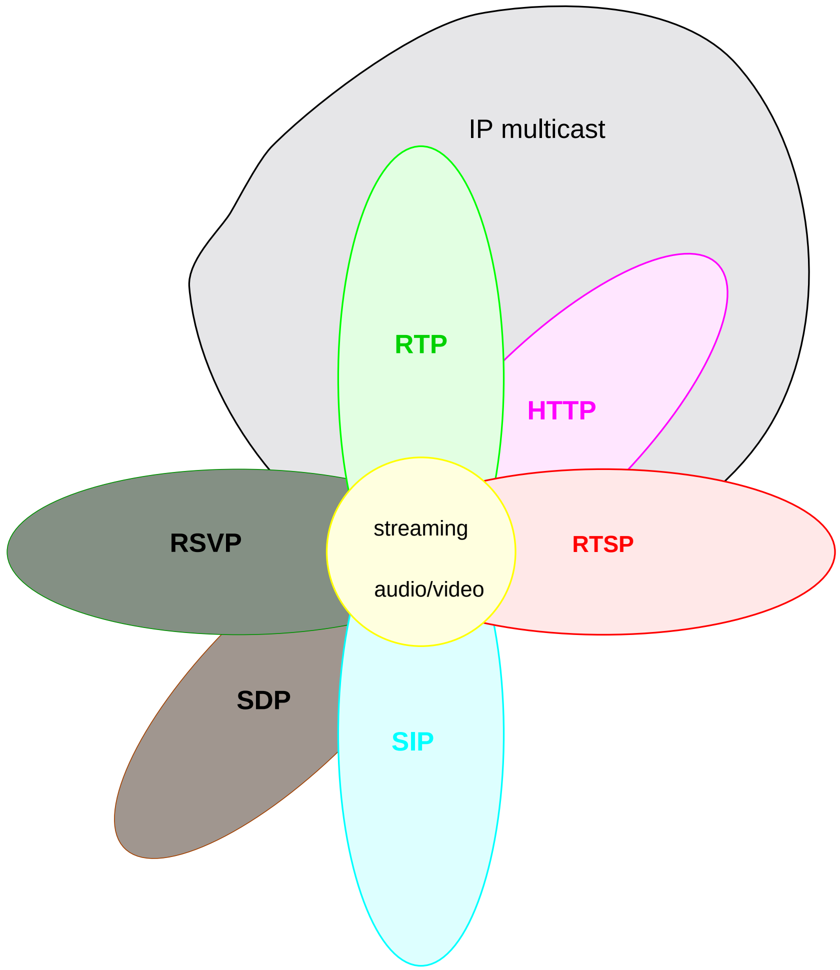
Streaming media: meta files



Streaming media: RTSP



Streaming multimedia



Internet real-time & multimedia protocols

resource reservation: RSVP, YESSIR, ...

media transport: RTP

stream control: $\boxed{\text{RTSP}}$

stream description: SDP, SMIL (W3C), RTSL, ...

Related work: DSM-CC, but much simpler

RTSP features

- “rough” synchronization (fine-grained ➠ RTP sender reports)
- virtual presentations = synchronized playback from several servers ➠ command timing
- load balancing using redirection at connect, during stream
- supports any session description
- device control ➠ camera pan, zoom, tilt
- caching: similar to HTTP, except “cut-through”

RTSP protocol design

- similar design as HTTP (TCP + UDP, HTTP, . . .)
- HTTP = “the Internet RPC protocol”
- supports any session description
- control “tracks” (audio, video) and “presentation” (movie)
- remote digital editing

RTSP sessions

TCP connection $\neq$ RTSP session ➠ session maintained by identifier

- one TCP connection per session ➠ firewalls, bidirectional
- one TCP connection per ≥ 1 command ➠ no server state
- UDP
  - multicast, low latency
  - ➠ "passing around the remote"
  - ➠ limit server connection state (live events!)

RTSP and HTTP: similarities

- protocol format: text, MIME-headers
- request/response = request line + headers + body
- status codes
- security mechanisms
- URL format
- content negotiation

RTSP protocol design

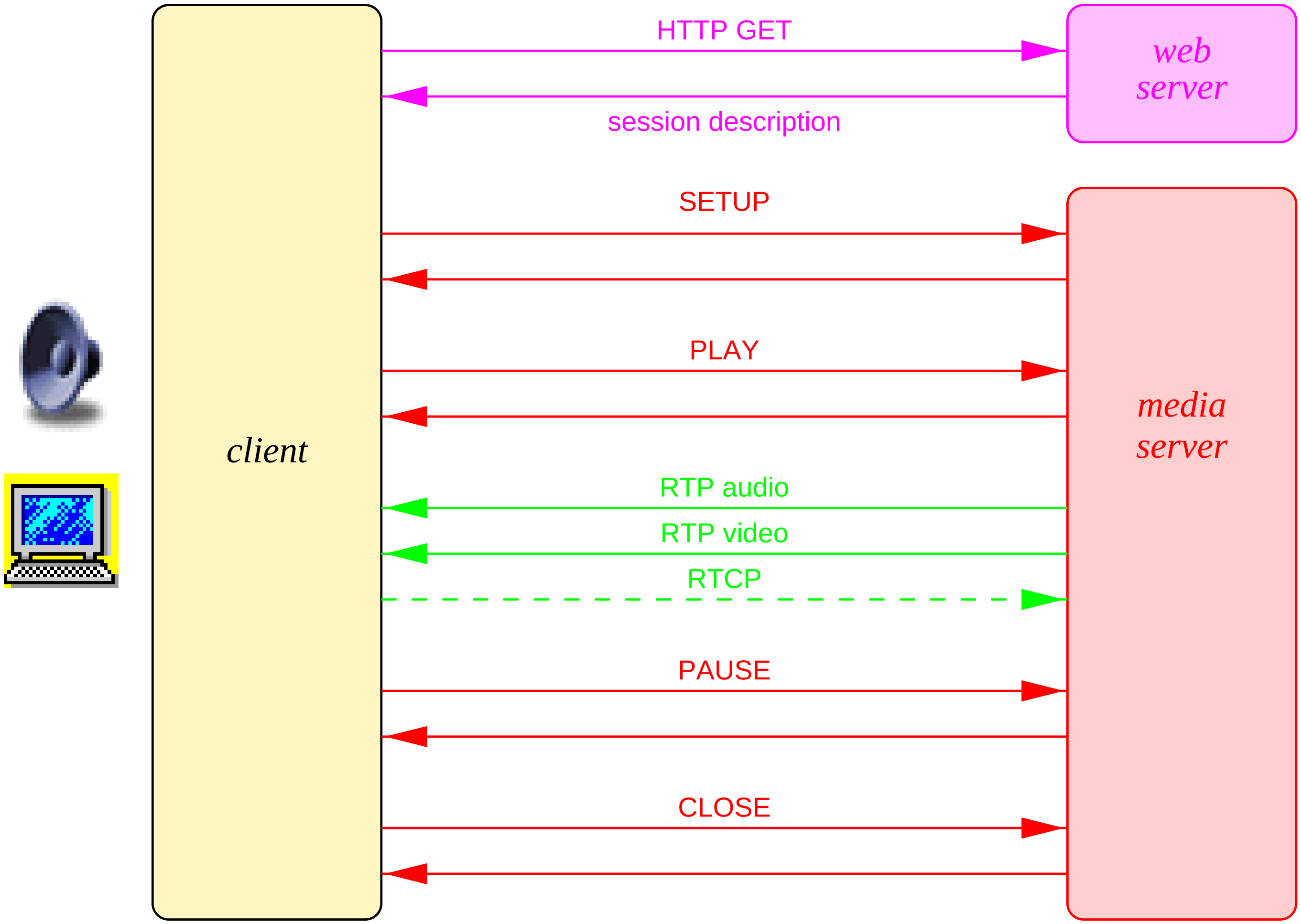
RTSP is not HTTP ➠

- server state needed
- different methods
- server $\longrightarrow$ client
- data carried out-of-band
- avoid HTTP mistakes:
  - relative request paths
  - no extension mechanism
  - 8859.1 coding

RTSP: HTTP inheritance

- simple servers are easy, Apache for industrial-strength
- re-use HTTP extensions:
  - authentication (basic, digest, . . .)
  - PICS = content labeling
  - JEPI = electronic payments
  - PEP = protocol extensions
- SSL for security

RTSP operation



RTSP functionality

retrieval: media-on-demand for continuous media

- first, get presentation description
- unicast
- multicast, client chooses address
- multicast, server chooses address (NVOD)
- independent of stream file format ➠ subsets or combinations of files

conference participant: “invite” to conference, controlled by several people

live streaming: ability to add media

one session = single time axis

Control

Aggregate control: one command ➠ control several streams

- content may be in *container file* (QuickTime, .wav, ASF, MPEG systems stream, rtpdump, . . .)
- on single server

Per-stream control: each stream has own command

- across container files
- several servers

RTSP URLs

whole presentation:

`rtsp://media.example.com:554/twister`

track within presentation:

`rtsp://media.example.com:554/twister/audiotrack`

but: name hierarchy $\neq$ media hiearchy $\neq$ file system

RTSP: Web integration

1. web page with "program guide"
2. contains pointer to presentation description (say, SMIL):

```
<session>
  <group>
    <track src="rtsp://audio.mtv.com/movie">
    <track src="rtsp://video.mtv.com/movie">
  </group>
</session>
```

3. RTSP sets up and controls delivery
4. RSVP reserves resources
5. RTP delivers data

RTSP methods

OPTIONS	get available methods
SETUP	establish transport
ANNOUNCE	change description of media object
DESCRIBE	get (low-level) description of media object
PLAY	start playback, reposition
RECORD	start recording
REDIRECT	redirect client to new server
PAUSE	halt delivery, but keep state
SET_PARAMETER	device or encoding control
TEARDOWN	remove state

commands may be pipelined

RTSP time

- normal play time (NPT): seconds, microseconds
- SMPTE timestamps (seconds, frames)
- absolute time (for live events)

allow absolute timing of events: ➠ "start playing movie at 10:05.34, at NPT = 10 s" ➠ synchronize distributed servers

- DSM-CC: single pending command
- RTSP: edit list (play 10-12, play 15-20, . . .) ➠ editing

Request headers

Accept	media description formats
Accept-Encoding	encoding of media format
Accept-Language	human language
Authorization	basic and digest authentication
Bandwidth	client bandwidth available
Conference	conference identifier
From	name of requestor
If-Modified-Since	conditional retrieval
Range	time range to play
Referer	how did we get here?
Scale	(play time)/(real time)
Speed	speed-up delivery
User-Agent	software

Response headers

Location	redirection
Proxy-Authenticate	authenticate to proxy
Public	methods supported
Retry-After	busy; come back later
Server	server software
Vary	cache tag
WWW-Authenticate	request authorization

RTSP reliability

- if TCP, send request once
- if UDP, retransmit with RTT (estimate: 500 ms)
- CSeq for request sequence
- Timestamp for RTT estimation
- atomicity: may pack requests into PDU
- kludge: data interleaving for TCP

RTSP descriptions

contains streams + initialization information [+ network info]:

- RTSP DESCRIBE
- http, email, ...
- command line
- updated via ANNOUNCE; both C-to-S and S-to-C

Unicast session: get description

```
C->W: GET /twister.sdp HTTP/1.1
      Host: www.example.com
      Accept: application/sdp

W->C: HTTP/1.0 200 OK
      Content-Type: application/sdp

      v=0
      o=- 2890844526 2890842807 IN IP4 192.16.24.202
      s=RTSP Session
      m=audio 0 RTP/AVP 0
      a=control:rtsp://audio.com/twister/audio.en
      m=video 0 RTP/AVP 31
      a=control:rtsp://video.com/twister/video
```

Unicast session: open streams

```
C->A: SETUP rtsp://audio.com/twister/audio.en RTSP/1.0
      CSeq: 1
      Transport: RTP/AVP/UDP;unicast
        ;client_port=3056-3057

A->C: RTSP/1.0 200 OK
      CSeq: 1
      Session: 12345678
      Transport: RTP/AVP/UDP;unicast
         ;client_port=3056-3057;
         ;server_port=5000-5001

C->V: SETUP rtsp://video.com/twister/video RTSP/1.0
      CSeq: 1
      Transport: RTP/AVP/UDP;unicast
         ;client_port=3058-3059
```

```
V->C: RTSP/1.0 200 OK
      CSeq: 1
      Session: 23456789
      Transport: RTP/AVP/UDP;unicast
         ;client_port=3058-3059
         ;server_port=5002-5003
```

Unicast session: play

```
C->V: PLAY rtsp://video.com/twister/video RTSP/1.0
      CSeq: 2
      Session: 23456789
      Range: smpte=0:10:00-

V->C: RTSP/1.0 200 OK
      CSeq: 2
      Session: 23456789
      Range: smpte=0:10:00-0:20:00
      RTP-Info: url=rtsp://video.com/twister/video
        ;seq=12312232;rtptime=78712811

C->A: PLAY rtsp://audio.com/twister/audio.en RTSP/1.0
      CSeq: 2
      Session: 12345678
      Range: smpte=0:10:00-
```

```
A->C: RTSP/1.0 200 OK
      CSeq: 2
      Session: 12345678
      Range: smpte=0:10:00-0:20:00
      RTP-Info: url=rtsp://audio.com/twister/audio.en
        ;seq=876655;rtptime=1032181
```

RTSP session teardown

```
C->A: TEARDOWN rtsp://audio.com/twister/audio.en RTSP/1.0
      CSeq: 3
      Session: 12345678

A->C: RTSP/1.0 200 OK
      CSeq: 3

C->V: TEARDOWN rtsp://video.com/twister/video RTSP/1.0
      CSeq: 3
      Session: 23456789

V->C: RTSP/1.0 200 OK
      CSeq: 3
```

PLAY and PAUSE

- several ranges (≥ 1 PLAY) are queued
- PAUSE intercepts first matching time point
- PLAY parameters:

  Scale: NPT speed $\updownarrow$

  Speed: delivery bandwidth $\updownarrow$

  Transport: for near-video-on-demand
- mute vs. pause
- implementation: calendar queue

REDIRECT

- server tells client: go elsewhere
- Location header contains URL
- load balancing
- needs to do TEARDOWN and SETUP

```
S->C: REDIRECT rtsp://example.com/fizzle/foo RTSP/1.0
      CSeq: 732
      Location: rtsp://bigserver.com:8001
      Range: clock=19960213T143205Z-
```

RECORD

- may use URL or create own ➠ return new URL in Location

```
C->S: RECORD rtsp://example.com/meeting/audio.en RTSP/1.0
      CSeq: 954
      Session: 12345678
      Conference: 128.16.64.19/32492374
```

Interaction with RTP

- PLAY response announces RTP timestamp and sequence number
- allow discarding of packets before break

RTP-Info: url=rtsp://foo.com/bar.avi/streamid=0;seq=45102,
url=rtsp://foo.com/bar.avi/streamid=1;seq=30211

Near video-on-demand

- in wide area, *video*-on-demand not scalable
- near on-demand, with positioning, pause
- popular content delivered every 5 minutes
- RTSP PLAY $t \longrightarrow$ join appropriate multicast group for t
- easy in Internet: IP multicast groups ➠ no network signaling
- may be able to “catch up” with group

RTSP caching

- proxy caching of *content*, not RTSP responses
- except: DESCRIBE
- parameters similar to HTTP:

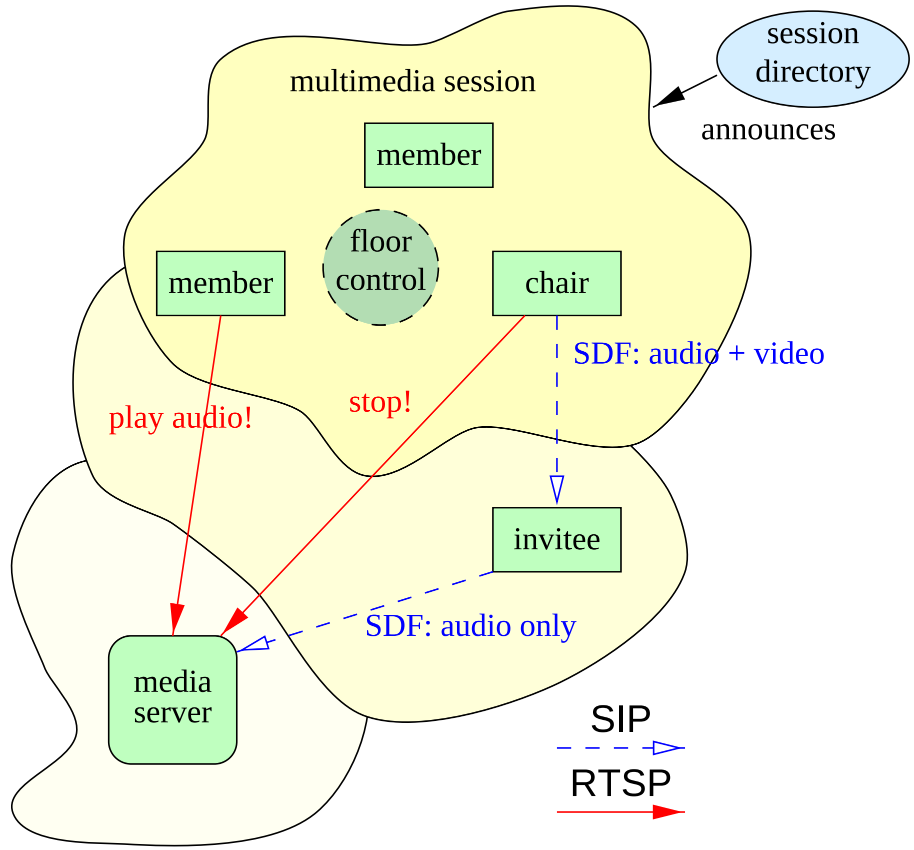
no-cache	don’t cache
public	anybody may cache
private	only end-user may cache
no-transform	conversion disallowed
only-if-cached	only if proxy has content
max-stale	except beyond expiration date
min-fresh	shelf life left
must-revalidate	ask first, proxy later

RTSP extensions

- add headers, methods
- Require header for must-understand extensions:

```
Require: org.ietf.rtsp.foobar
501 Not implemented
```

SIP and RTSP integration



- provide transport parameters to RTSP explicitly

- H.323 needs introductions ➠ conference identifier

RTSP status

- IETF MMUSIC working group ➠ RFC 2326
- active contributors: Columbia University, Netscape, RealNetworks; IBM, INRIA, Microsoft, ...
- implementations in progress:
  - Columbia University (NT, Unix)
  - IBM
  - Lucent
  - Netscape
  - RealNetworks (G2)
- may use existing Mbone tools

RTSP implementation

Example: Columbia `rtspd`

- share parser with SIP (Internet telephony) server
- basic UDP and TCP (per connection) threads: listen for RTSP requests, assign to session
- thread that picks up timed PLAY and PAUSE request
- thread that cycles through multimedia file
- RTP packetizer

Summary

- Internet multimedia-on-demand ➠ integrated services Internet
- building block for virtual reality systems
- conferencing $\stackrel{?}{\longleftrightarrow}$ telephony ➠ same tools, formats, network
- WebTV as VOD, Internet telephony terminal?
- digital TV: specialized protocols ➠ IP over the air
- Columbia *MarconiNet* for TV/radio network architecture
- 18 GB disk ➠ download movie at night?