

# 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

December 4, 2001

## Overview

---

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

December 4, 2001

## Internet multimedia (on demand)

---

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

- just one service among many  $\Rightarrow$  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  $\Rightarrow$  lots of content

December 4, 2001

## Internet multimedia

---

Same infrastructure, different delivery modes:

**on demand:** unicast

**near on-demand:** staggered transmission on multicast  $\Rightarrow$  VCR control

**multicast:** niche markets to audience of millions

December 4, 2001

## Applications

---

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

December 4, 2001

## Internet radio

---

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

December 4, 2001

## 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

December 4, 2001

## Internet streaming media requirements

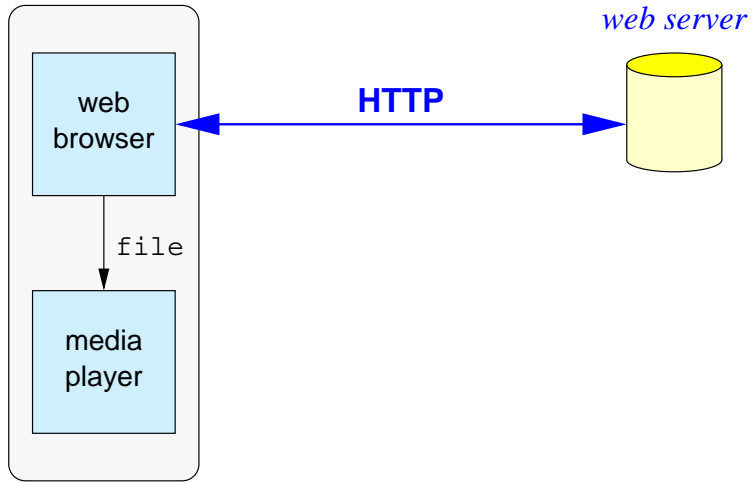
---

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

Have some proprietary protocols, need interoperability

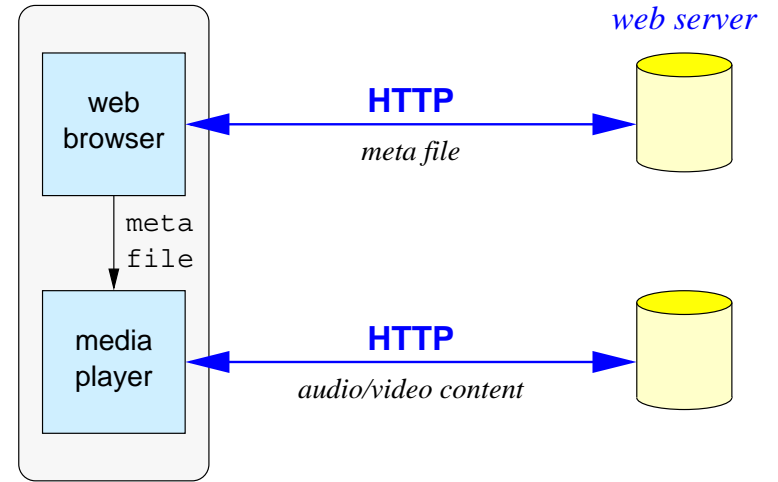
December 4, 2001

# Streaming media: download



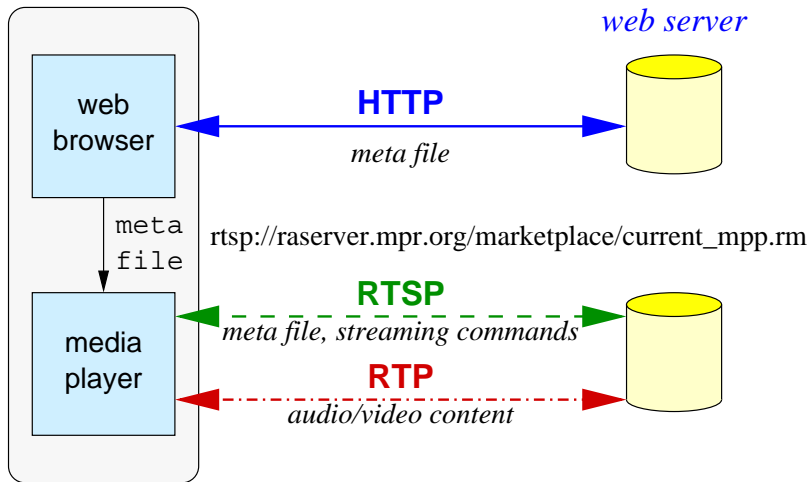
December 4, 2001

# Streaming media: meta files



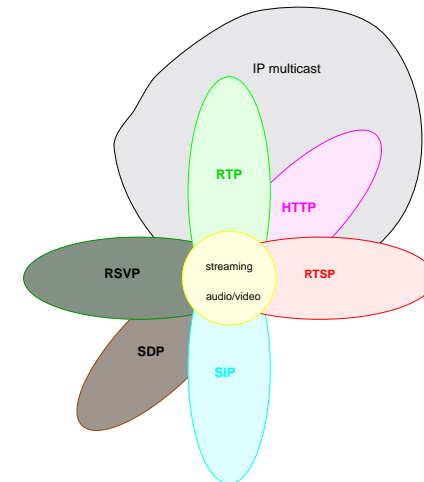
December 4, 2001

# Streaming media: RTSP



December 4, 2001

# Streaming multimedia



December 4, 2001

## Internet real-time & multimedia protocols

---

**resource reservation:** RSVP, YESSIR, ...

**media transport:** RTP

**stream control:** RTSP

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

Related work: DSM-CC, but much simpler

December 4, 2001

## RTSP features

---

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

December 4, 2001

## 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

December 4, 2001

## RTSP sessions

---

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

- one TCP connection per session  $\Rightarrow$  firewalls, bidirectional
- one TCP connection per  $\geq 1$  command  $\Rightarrow$  no server state
- UDP
  - multicast, low latency
  - $\Rightarrow$  “passing around the remote”
  - $\Rightarrow$  limit server connection state (live events!)

December 4, 2001

## RTSP and HTTP: similarities

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

December 4, 2001

## RTSP protocol design

RTSP is not HTTP ⇨

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

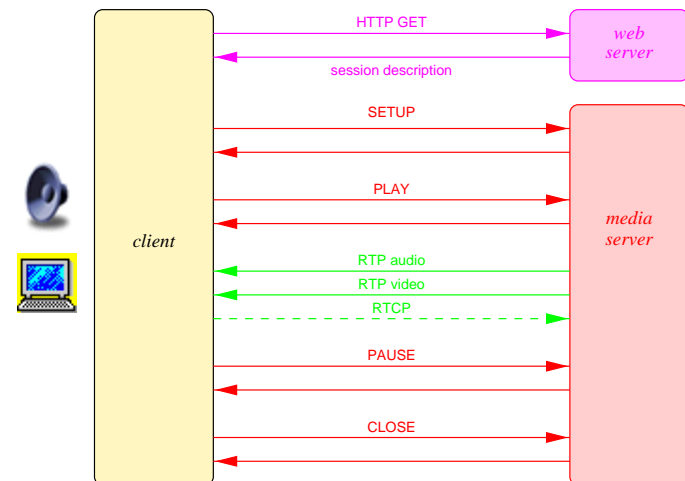
December 4, 2001

## 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

December 4, 2001

## RTSP operation



December 4, 2001

## 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  $\Rightarrow$  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

December 4, 2001

## Control

---

**Aggregate control:** one command  $\Rightarrow$  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

December 4, 2001

## 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 hierarchy  $\neq$  file system

December 4, 2001

## 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

December 4, 2001

## 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

December 4, 2001

commands may be pipelined

December 4, 2001

## RTSP time

---

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

allow absolute timing of events:  $\Rightarrow$  “start playing movie at 10:05.34, at NPT = 10 s”  $\Rightarrow$  synchronize distributed servers

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

December 4, 2001

## 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

December 4, 2001

## 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

December 4, 2001

## 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

December 4, 2001

## 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

December 4, 2001

## 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
```

December 4, 2001

## 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
```

December 4, 2001

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

December 4, 2001

## 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-
```

December 4, 2001

```
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
```

December 4, 2001

## 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
```

December 4, 2001

## PLAY and PAUSE

---

- several ranges ( $\geq 1$  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

December 4, 2001

## 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-
```

December 4, 2001

## RECORD

---

- may use URL or create own  $\Rightarrow$  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
```

December 4, 2001

## 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

December 4, 2001

## 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  $t \rightarrow$  join appropriate multicast group for  $t$
- easy in Internet: IP multicast groups  $\Rightarrow$  no network signaling
- may be able to “catch up” with group

December 4, 2001

## RTSP caching

---

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

December 4, 2001

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

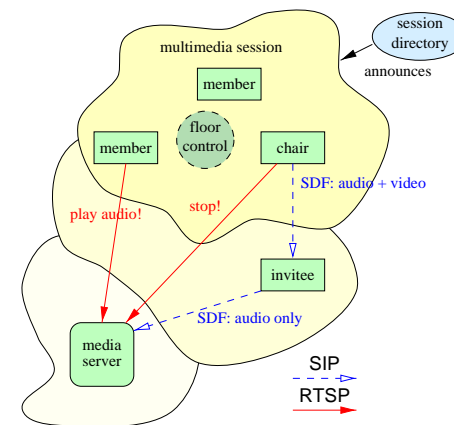
December 4, 2001

## RTSP extensions

- add headers, methods
- Require header for must-understand extensions:  
Require: org.ietf.rtsp.foobar  
501 Not implemented

December 4, 2001

## SIP and RTSP integration



- provide transport parameters to RTSP explicitly

December 4, 2001

- H.323 needs introductions → conference identifier

December 4, 2001

## 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

December 4, 2001

## 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

December 4, 2001

## Summary

---

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

December 4, 2001