RTSP Prime (RTSP')

- semantics very similar to RTSP-00
- textual encoding; but: RTSP' would work text or binary
- works with TCP and UDP
- re-use of HTTP authentication
- better fit with MMUSIC conference control, session description
- allow evolution of session description
- syntax is strawman – don't take too seriously

Media Server Operation

“Traditional” media-on-demand server – live or stored media:

1. RTSP’, HTTP: client retrieves session description, including multicast address (if applicable)
2. client picks desirable stream combination, indicates unicast port
3. RTSP’: client sets port, speed, …
4. RTSP’: client issues PLAY, … requests for individual media or session
5. server can announce new media (for live sessions)
6. RTSP: client or server says BYE

Media Server in Conference

1. S*IP?: conference participant (CP) invites server
   e.g.: play audio into existing multicast session
   session establishment outside scope (e.g., H.323)
   session description in invitation
   server joins group and waits for commands

2. RTSP’: GET description (language, content, but not address)

3. RTSP’: CP sets parameters, controls playout

Naming

- URL-like name for session and individual components:
  host/session/media

```
movies.microsoft.com/twister/audio_en#11:12
```

Example: join a broadcast

```
C->S  GET twister

S->C  200 OK
      session description
      (media (m audio 224.2.0.1 3456) (id audio_en))

C->S  PLAY twister/audio_en
```

Example: unicast media server

```
C->S GET twister

S->C 200 OK
     (media (m audio)

C->S PORT 48484
     PLAY twister/audio_en
```

denial-of-service attack ➠ only to address of request

Example invitation to conference

Client $\rightarrow$ server:

```
S*IP invitation -> conference id

GET twister 17
Conference: conference identifier
```

S $\rightarrow$ C:

```
200 17 OK
Content-type: application/sdf
...
(media (audio ...) (language en/us) (id audio_en)
(media (video ...) (id video)
```

C $\rightarrow$ S:

```
PLAY twister/audio_en
Content-range: 0:5:6.0
```

```
PLAY twister/video

SET_SPEED twister
Speed: 0.5
```

Camera Control

➠ just more commands
difficulty: naming (several cameras attached to host)

```
PAN camera1
Angle: 125
```

RTSP’ (textual) over TCP

- simply delineate requests: blank line after header, content-length for session description, etc.
- embed RTP, RTCP, text, . . . packets using content-length

RTSP' over UDP

- several commands in one packet
- sequence number (per packet) for reliability
- RTT ➠ retransmission = different seq. #

Other RTSP Issues

- parameter settings: advantage of second level not clear
- separate issue: who gets to control ➠ floor control or social protocol
- request for report should be in RTCP (with sampling option for large groups)
- retransmission should be in RTCP ➠ already there for H.261!
- difference between PAUSE and STOP? RSVP?

Session Description

- current SDP is compact, but not good for structure
- ➠ re-use much of SDP coding, but add structure
- easy inheritance of parameters

all: play all list elements concurrently

one-of: pick one the list elements

sequence: play list elements sequentially

Parameters: at any level; globally unique names?

SDF

```
(session (v 0)
  (o mhandley 2890844526 2890842807 IN IP4 126.16.64.4)
  (s Sd seminar)
  (i A seminar on the session description protocol)
  (u http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.01.ps)
  (e M.Handley@cs.ucl.ac.uk (Mark Handley))
  (c IN IP4 224.2.17.12/127)(t 2873397496 2873404696)(a recvonly)
  (all
    (media (m audio 3456 RTP (
      (one-of
        ((enc PCMU) (id Foo))
        ((enc DVI)  (id Bar))
     )
    (media (m video 2232 RTP H261) (id Pix))
    (media (m whiteboard 32416 UDP WB)(orient portrait))
  ))
```