Outline of the presentation

- 1- the context
- 2- the RTP/RTCP protocols
- 3- the RTSP protocol
- 4- selected bibliography
PART 1:
The context

A general view of the real-time protocols

- the protocols and their application field...
  - stream description: SDP, SMIL...
    describe the session and content
  - stream control: RTSP
    remote control the session
  - media transport: RTP
    send data and metadata

- resource reservation (if any!): RSVP, DiffServ
  make sure the communication path offers appropriate guaranties...
  ...otherwise Best-Effort transmissions!
PART 2:
The RTP/RTCP protocols

- 2.1 Overview
- 2.2 RTP generalities
- 2.3 header format
- 2.4 mixers vs. translators
- 2.5 RTCP generalities
- 2.6 RTP profiles
- 2.7 some implementations
2.1- Overview

- IETF Audio/Video Transport WG
  - RTPv1 RFC 1889 (January 1996)
  - RTPv2 draft-ietf-avt-rtp-new-09.txt (March 2001)

- Real-Time Protocol (RTP)
  - understand: « a framing protocol for real-time applications »
  - does not define any QoS mechanism for real-time delivery!

- Real-Time Control Protocol (RTCP)
  - its companion control protocol, just here to get some feedback
  - does not guaranty anything either!

Overview… (cont’)

- Design goals:
  - flexible provide mechanisms, do not dictate algorithms!
    - instantiations for H261, MPEG1/2/...
  - protocol neutral UDP/IP, private ATM networks...
  - scalable unicast, multicast, from 2 to ∞
  - separate control/data some functions may be taken over by conference control protocol (e.g. RTSP)
2.2- RTP generalities

- data functions (RTP)
  - content labeling
  - source identification
  - loss detection
  - resequencing

- timing
  - intra-media synchronization: remove jitter with playout buffers
  - inter-media synchronization: lip-synchro between audio-video

Typical use

- Usually...
  - uses UDP (TCP is not for real-time!!!)
  - no fixed UDP port negotiated out of band
  - UDP port for RTCP = UDP port for RTP + 1
  - usually one media per RTP session (i.e. port pair)
**Time management**

Two times...

- **RTP time**
  - Random initial offset (for each stream)
  - RTP timestamp present in each data packet
  - Increases by the time « covered » by a packet

- **NTP time (or wallclock time)**
  - Absolute time (use Network Time Protocol format)
  - NTP timestamp present in each RTCP Sender Report
  - Enables inter-stream synchronization

---

**2.3- RTP header**

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--------------------------+--------------------------+--------------------------+--------------------------+
| V=2 | P | X |  CC   | M |     PT      |       sequence number     |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           |                           |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           | timestamp                |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           |                           |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           |                           |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           |                           |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           |                           |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           |                           |                           |
+--------------------------+--------------------------+--------------------------+--------------------------+
|                           | padding | count                |
+--------------------------+--------------------------+--------------------------+--------------------------+
| version (V)              | CSRC count (CC)          |
| padding (P)               | marker (M)               |
| extension (X)             | payload type (PT)        |
```
RTP header... (cont’)

- Sequence number field
  - Incremented for each RTP packet

- Synchronization Source (SSRC) field
  - Uniquely identifies the source in the session
  - Chosen randomly ⇒ detect and solve collisions

- Contributing Source (CSRC) and CC fields
  - Used by a mixer to identify the contributing sources
  - Size of the list given by the CSRC Count (CC) field

2.4- Mixers versus Translators

- Mixers
  - Combines several flows in a single new one
  - Use new SSRC; original SSRCs in a CSRC list
  - Appears as a new source
  - E.g. bandwidth reduction for dial-up networks

- Translators
  - Passes through the various flows separately
  - Keep original SSRCs
  - E.g. protocol translation, firewall
**Mixer**

- A mixer may change the data format (coding) and combine the streams in any manner.
- Example: video mixer (~MCU)

**Translator**

- Data may pass through the translator intact or may be encoded differently.
- The identity of individual flows remains intact.
- Example: going through a firewall.
2.5- RTCP generalities

- periodic transmission of control packets
- several functions
  - feedback on the quality of data distribution
  - let everybody evaluate the number of participants
  - persistant transport-level canonical name for a source, CNAME
    - usually: user@host
    - will not change, even if SSRC does!
    - provides binding across multiple media tools for a single user

RTCP generalities… (cont’)

- Five RTCP packets
  - SR sender reports
    tx and rx statistics from active senders
  - RR receiver reports
    rx statistics from other participants, or from active senders if more than 31 sources
  - SDES source description, including CNAME
  - BYE explicit leave
  - APP application specific extensions
**RTCP generalities… (cont’)**

● **distribution**
  - use same distribution mechanisms as data packets (n→m multicast)
  - multiple RTCP packets can be concatenated by translators/mixers
    ⇒ compound RTCP packet

● **scalability with session size**
  - RTCP traffic should not exceed 5% of total session bandwidth
    requires an evaluation of number of participants
    RTCP tx interval = f(number of participants)
  - at least 25% of RTCP bandwidth is for source reports
    let new receivers quickly know CNAME of sources!

---

**SR RTCP packets**

● **includes**
  - SSRC of sender: identify source of data
  - NTP timestamp: when report was sent
  - RTP timestamp: corresponding RTP time
  - packet count: total number sent
  - octet count: total number sent
  - followed by zero or more receiver report…

● example:

  source 1 reports, there are 2 other sources

  ![RTCP packet diagram]
**RR RTCP packets**

- includes
  - SSRC of source: identify the source to which this RR block pertains
  - fraction lost: since previous RR (SR) sent
    \[= \text{int}(256 \times \frac{\text{lost}}{\text{expected}})\]
  - cumul # of packets lost: long term loss
  - highest seq # received: compare losses
  - interarrival jitter: smoothed interpacket distortion
  - LSR: time when last SR heard
  - DLSR: delay since last SR

---

**SDES RTCP packet**

- may contain:
  - a CNAME item (canonical identifier/name)
  - a NAME item (real user name)
  - an EMAIL item
  - a PHONE item
  - a LOC item (geographic location)
  - a TOOL item (application name)
  - a NOTE item (transient msg, e.g. for status)
  - a PRIV item (private extension)
Example of RTCP compound packet

- Aggregation of two packets (e.g. by a mixer)

2.6- RTP profiles

- RTP is generic… define a profile for each target media!
  - Example: MPEG1/2 video packetization
  - Must follow general guidelines
    - RFC 2736, December 1999
  - Goal:
    - « every packet received must be useful »

- Potential problems
  - Over standard Internet packets may be
    - Lost
    - Reordered
    - Fragmented by IP if size > MTU (max tx unit)
**RTP profiles... (cont')**

- **Example of what must not be done!!!**
  - Loss multiplication effect due to bad framing

```
application
 application data unit

RTP
 fragment 1
 fragment 2
 fragment 3

network tx

RTP
 fragment 1
 fragment 2

application
 uncomplete data!!!

Src

LOST

Rx

useless!!!
```

**RTP profiles... (cont')**

- **Principles**
  - **ALF (Application Level Framing)**
    - Unit of transmission == unit of control
    - Each unit is self-sufficient and can be processed as soon as it is received
  - An RTP/UDP/IP packet should consist of one or more complete codec frame
  - If a codec frame size is larger than MTU, the application must define its own fragmentation mechanism
### 2.7- Some implementations (by Schulzrinne)

<table>
<thead>
<tr>
<th>Organization</th>
<th>product name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bell Labs, Columbia Univ., Massachusetts Univ.</td>
<td>RTP Library</td>
</tr>
<tr>
<td>Version: 3.0 Beta (4/1997)</td>
<td></td>
</tr>
<tr>
<td>An RTP implementation by the standard authors, source code available.</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.bell-labs.com/topic/swdist/">http://www.bell-labs.com/topic/swdist/</a></td>
<td></td>
</tr>
<tr>
<td>Limburgs Universiteits Centrum</td>
<td>JVOPLIB</td>
</tr>
<tr>
<td>Version: 1.0 11/2000</td>
<td></td>
</tr>
<tr>
<td>JVOPLIB is an object-oriented Voice over IP (VoIP) library written in C++.</td>
<td></td>
</tr>
<tr>
<td>JavaSoft / Sun Microsystems</td>
<td>Java Media Framework</td>
</tr>
<tr>
<td>RTP application and library for audio/video session playback over IP networks.</td>
<td></td>
</tr>
<tr>
<td><a href="http://java.sun.com/products/java-media">http://java.sun.com/products/java-media</a></td>
<td></td>
</tr>
<tr>
<td>Limburgs Universiteits Centrum and Universiteit Maastricht</td>
<td>JRTPLIB</td>
</tr>
<tr>
<td>Version: 2.4 7/2000</td>
<td></td>
</tr>
<tr>
<td>JRTPLIB (Jori's RTP library) is a C++ library.</td>
<td></td>
</tr>
<tr>
<td>Live.com</td>
<td>LIVE.COM Streaming Media</td>
</tr>
<tr>
<td>Version: 1.0 (October 2000)</td>
<td></td>
</tr>
<tr>
<td>This C++ code, distributed under a LGPL licence, forms a set of libraries that can be used within RTP/RTCP streaming applications, or can be extended to support additional media types and codecs (in addition to MP3 audio).</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.livesourcesforge.net/">http://www.livesourcesforge.net/</a></td>
<td></td>
</tr>
<tr>
<td>UCL</td>
<td>Common Multimedia Library</td>
</tr>
<tr>
<td>Version: 1.2.8 (May 2001)</td>
<td></td>
</tr>
<tr>
<td>The UCL.common multimedia library implements a number of algorithms and protocols (including RTP) needed by a number of our applications.</td>
<td></td>
</tr>
<tr>
<td><a href="http://www-cs.cs.ucl.ac.uk/multimedia/software/common/index.html">http://www-cs.cs.ucl.ac.uk/multimedia/software/common/index.html</a></td>
<td></td>
</tr>
</tbody>
</table>

---

**PART 3:**

The RTSP protocol

- 3.1 generalities
- 3.2 an example
- 3.3 a bit more in details
- 3.4 some implementations
3.1- RTSP generalities

- IETF standard
  - RFC 2326

- Real-Time Streaming Protocol
  - acts as a « network remote control »

- supports the following operations:
  - retrieval of a media from a server
  - invitation of a media server to a conference
  - recording of a conference

RTSP generalities… (cont’)

- Protocol design
  - text-based protocol
  - transport protocol independant
  - supports any session description (sdp, xml, etc.)
  - similar design as HTTP with differences yet!
    - client → server and server → client requests
    - server maintains a « session state »
    - data carried out-of-band
  - works either with unicast or multicast
RTSP URL

- whole presentation
  rtsp://media.example.com:554/twister

- a track within the presentation
  rtsp://media.example.com:554/twister/audio

- name hierarchy ≠ media hierarchy ≠ file hierarchy

RTSP messages

- a request (client → server or server → client)
  method to apply: PLAY
  URL: rtsp://video.example.com/twister/video
  RTSP version: RTSP/1.0
  CSeq: 2
  Session: 23456789
  Range: smpte=0:10:00-

- and its response
  version: RTSP/1.0
  status code: 200
  reason phrase: OK
  CSeq: 2
  Session: 23456789
  Range: smpte=0:10:00-0:20:00
  RTP-Info: url=rtsp://video.example.com/twister/video;
  seq=12312232; rtptime=78712811
RTSP methods

- major methods
  - SETUP: server allocates resources for a stream and starts an RTSP session
  - PLAY: starts data tx on a stream
  - PAUSE: temporarily halts a stream
  - TEARDOWN: free resources of the stream, no RTSP session on server any more

- additional methods
  - OPTIONS: get available methods
  - ANNOUNCE: change description of media object
  - DESCRIBE: get low level descr. of media object
  - RECORD: server starts recording a stream
  - REDIRECT: redirect client to new server
  - SET_PARAMETER: device or encoding control

3.2- Example: media on demand, unicast

- Scenario...

  - Client (C) communicates with media描述 (W), audio server (A), and video server (V)
  - Step 1: get description (in SDP format)
  - Step 2: open streams with RTSP
  - Step 3: play
  - Step 4: teardown
Example... (cont’)

Another view

Example: step 1, 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.example.com/twister/audio.en
m=video 0 RTP/AVP 31
a=control:rtsp://video.example.com/twister/video
Example: step 2, open streams

C->A: SETUP rtsp://audio.example.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.example.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

Example: step 3, play

C->V: PLAY rtsp://video.example.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.example.com/twister/video;
              seq=12312232;rtptime=78712811
Example: step 3, play… (cont’)

C->A: PLAY rtsp://audio.example.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.example.com/twister/audio.en;
            seq=876655;rtptime=1032181

NB: use standard RTP synchronization methods for lip-sync!

Example: step 4, teardown

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

V->C: RTSP/1.0 200 OK
    CSeq: 3
3.3- A bit more in details: time management

- **NPT** (normal play time)
  - gives stream absolute position relative to start of the presentation
  - two formats:
    - hours:minutes:seconds.subseconds
    - seconds.subseconds
  - this is the info displayed on a viewer!
  - example:
    - npt= 10:07:00- 10:07:33
      - range start
      - range end
      - (hh:mm:ss)
    - npt=121.45-125
      - range start
      - range end
      - (sec.sub)
      - (sec)

---

Time management… (cont’)

- **SMPTE**
  - relative to start of clip
  - format:
    - hours:minutes:seconds:frames.subframes
  - example:
    - smpte-30-drop=10:07:00-10:07:33:05.01
      - range start
      - range end
      - (hh:mm:ss)
      - (hh:mm:ss:fr.sub)

- **absolute time**
  - follows ISO 8601, using UTC/GMT
  - example:
    - current time is 20010829T114501.00Z
      - date
      - time
      - (yyyyymmdd)
      - (hhmss.sub)
Time management… (cont’)

- Allows absolute timing of events
  - Example:
    start playing movie at 20010829T114501.00Z, at npt= 10:07:00
  - Enables distributed server synchronization

- Allows media editing
  - Frame precision

A bit more in details: RTSP communications

- Three working modes
  - Persistent transport connections (TCP) for several requests/responses
  - A connection (TCP) per request/response
  - Connectionless mode (UDP)

- Indicated by the URL:
  - « rtsp » assumes persistent connection
  - « rtspu » assumes connectionless

- Server to client requests require the « persistent connection » mode
  - The only reliable way to reach a client!
A bit more in details: Invitation

- goal
  - invite a server to add a media in an existing presentation

- client uses SETUP method + transport/conference params
  - the client indicates to the media server the transport and conference parameters to use

Invitation… (cont’)

- example: server M is invited to add a sound track to the conference

  C->M: DESCRIBE rtsp://server.example.com/demo/548/sound
  RTSP/1.0
  CSeq: 1
  Accept: application/sdp

  M->C: RTSP/1.0 200 1 OK
  Content-type: application/sdp
  Content-Length: 44

  v=0
  o=- 2890844526 2890842807 IN IP4 192.16.24.202
  s=RTSP Session
  i=See above
  t=0 0
  m=audio 0 RTP/AVP 0
**Invitation... (cont’)**

C->M: SETUP rtsp://server.example.com/demo/548/sound RTSP/1.0
CSeq: 2
Transport: RTP/AVP;multicast;destination=225.219.201.15;
port=7000-7001;ttl=127
Conference:
199702170042.SAA08642@obiwan.arl.wustl.edu%20Starr
M->C: RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP;multicast;destination=225.219.201.15;
port=7000-7001;ttl=127
Session: 91389234234
Conference:
199702170042.SAA08642@obiwan.arl.wustl.edu%20Starr

C->M: PLAY rtsp://server.example.com/demo/548/sound RTSP/1.0
CSeq: 3
Session: 91389234234
M->C: RTSP/1.0 200 OK
CSeq: 3

---

**A bit more in details: Recording**

- **goal**
  - ask a server to record an active session

- **client uses ANNOUNCE and RECORD methods**
  - Announce method enables the client to give (e.g. the sdp) presentation description
  - Record method enables the client to record the presentation for a given time interval
A bit more in details: Cache control

- goal
  - control the way the media stream may be cached

- example of directives
  - no-cache: never cache
  - public: stream cacheable by any cache
  - private: don’t cache unless by private cache
  - no-transform: never alter the stream content
  - only-if-cached: e.g. play a stream only if cached
  - max-stale: only if expired from less than that
  - min-fresh: only if won’t expire after that value
  - must-revalidate

3.4- Some implementations (by Schulzrinne)

<table>
<thead>
<tr>
<th>Organization type</th>
<th>OS</th>
<th>product name</th>
<th>session description</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Darwin</td>
<td>QuickTime Streaming Server</td>
<td>open source, full RTP/RTSP server</td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>QuickTime 4</td>
<td>SDP</td>
<td>QuickTime 4 also supports importing SDP files that describe multicasts, with standard decoders.</td>
<td></td>
</tr>
<tr>
<td>Columbia University</td>
<td>Wrtp</td>
<td>SDP</td>
<td>no container files</td>
<td></td>
</tr>
<tr>
<td>Columbia University</td>
<td>rtspd</td>
<td>SDP</td>
<td>MPEG-1 System; alternative RTSP server for IBM’s VideoCharger</td>
<td></td>
</tr>
<tr>
<td>Columbia University</td>
<td>KOM-URI</td>
<td>SDP</td>
<td>Delivers MPEG-2 video over ATM, QAM, or DVB-ASI; no RTP delivery.</td>
<td></td>
</tr>
<tr>
<td>Columbia University</td>
<td>rtpserv</td>
<td>SDP</td>
<td>MPEG-1 Systems, MPEG-2 Transport, AVI codecs. ATM, QAM, IP, DVB-ASI. No RTP delivery.</td>
<td></td>
</tr>
<tr>
<td>Oracle Corporation</td>
<td>Real Networks</td>
<td>SDP</td>
<td>Supports RTP for RTP-based clients</td>
<td></td>
</tr>
<tr>
<td>Oracle Corporation</td>
<td>Real Networks</td>
<td>SDP</td>
<td>Streams .au and .wav files over RTP when configured for multicast.</td>
<td></td>
</tr>
<tr>
<td>Real Networks</td>
<td>Solaris</td>
<td>SDP</td>
<td>RTP</td>
<td></td>
</tr>
<tr>
<td>Real Networks</td>
<td>Solaris</td>
<td>SDP</td>
<td>RTP</td>
<td></td>
</tr>
<tr>
<td>Real Networks</td>
<td>Solaris</td>
<td>SDP</td>
<td>RTP</td>
<td></td>
</tr>
<tr>
<td>Real Networks</td>
<td>Solaris</td>
<td>SDP</td>
<td>RTP</td>
<td></td>
</tr>
<tr>
<td>Real Networks</td>
<td>Solaris</td>
<td>SDP</td>
<td>RTP</td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>JMF 2.1</td>
<td>SDP</td>
<td>RTP-based servers</td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>MediaCentral server</td>
<td>SDP</td>
<td>RTP-based servers</td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>Open source, RECORD and PLAY</td>
<td>SDP</td>
<td>RTP-based servers</td>
<td></td>
</tr>
</tbody>
</table>
Selected bibliography

- **General**

- **RTP**

- **RTSP**