RTP and VoIP

Real-Time Protocol (RTP)
Real-Time Transport Protocol (RTP)

- A protocol for carrying generic real-time data (e.g., multimedia)
- RFC 1889

An application-layer protocol

- RTP runs at the App layer (“Application-Level Framing”)
  - The idea of ALF: “The application knows best how to handle its own needs”
- Normally implemented as a library called by user code
- Designed to run over UDP
Why not just use HTTP or some other TCP-based protocol?

- TCP
  - Emphasizes reliability without regard to real-time needs
  - Limits data rates with an eye to network performance, not datastream requirements

RTP functions

- Multiplex multiple streams of real-time data onto a stream of UDP packets
  - UDP stream can be unicast or multicast
- Note: Normally, the network just treats these as standard UDP packets – no special handling
### RTP packet format

<table>
<thead>
<tr>
<th>Ver</th>
<th>P</th>
<th>X</th>
<th>CC</th>
<th>M</th>
<th>Payload type</th>
<th>Sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Synch source identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>payload</td>
</tr>
</tbody>
</table>

- **Ver**: RTP version number
- **P**: 1 if packet is padded (last pad byte tells how many added)
- **X**: 1 if an Extension Header is present (allows for future extension of the protocol)
- **CC**: Number of contributing sources (if more than one, sources are id'ed at end of payload)
- **M**: Application-specific marker bit (e.g., to mark the start of a video frame)
- **Payload type**: Type of encoding used (uncompressed, MP3, …)
- **Timestamp**: In “ticks” — RTP does not specify the units, but leaves that to the application
- **Synch source**: Identity of the sending node

---

### Synch Source

- Uniquely identifies one sender
- When setting up the RTP connection, each sender randomly chooses a 32-bit synch source number (apps must detect and resolve the rare conflicts)
- Why not just the IP address?
  - One node may have several streams
  - Designers wanted to be independent of lower-level protocols
RTP provisions for error control

- Very limited – Error recovery decisions are left to the application
- Seq number allows receiver to detect lost packets
  - Usually no time for re-trans in a real-time stream – receiver often ignores or interpolates
- Timestamp helps receiver adjust for jitter

RTP’s control protocol

- Real-time Transport Control Protocol (RTCP)
  - End-to-end clock synch
  - Feedback of quality data from receiver to sender (e.g., jitter rate)
  - Source identification (e.g., Who’s talking now)
RTSP

- An emerging RFC (RFC 2326) for controlling multimedia streams
- Goal: “VCR-like” controls
  - SETUP / PLAY / PAUSE / CLOSE
  - Synchronized playback from multiple senders
  - Remote control (e.g., Camera panning)

Voice Over IP (VoIP)
Voice over IP (VoIP)

- The idea: Carry telephone-class voice traffic over IP networks
- Two approaches:
  - H.323 (telco-designed)
  - SIP (IETF-designed)
H.323

• Overarching standard is ITU H.323 (1996)
  – Defines an architecture:
    • Internet and POTS coexist with gateway between them
    • LANs can have a “gatekeeper” to manage user terminals (computers) – the set of terminals managed is called a "zone"

The H.323 protocol stack

<table>
<thead>
<tr>
<th>speech</th>
<th>control</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.7xx</td>
<td>RTCP</td>
</tr>
<tr>
<td>RTP</td>
<td>H.225 (RAS)</td>
</tr>
<tr>
<td></td>
<td>Q.931 (Call signaling)</td>
</tr>
<tr>
<td>UDP</td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>IP</td>
</tr>
</tbody>
</table>
Speech encoding/decoding (G.7xx)

- Basic codec is G.711 (any H.323 system must support)
- G.711:
  - Single voice channel
  - 8K samples/s x 8 bit/sample => 64Kbps
  - POTS quality

Alternate codecs

- Other encoding/decoding protocols may be used
- Example:
  - G.723.1
    - Takes 30msec of speech (240 samples)
    - Uses predictive coding to compress to either 24 or 20 bytes (compression factor of 10 or 12)
    - Data rate = 6.4 Kbps or 5.3 Kbps
    - Close to POTS quality
Control protocols

• H.245
  – Sets connection parameters (e.g., which codec to use)

• Q.931
  – Establishes and releases connections

• H.225 (RAS)
  – Manages the channel between the PC and gatekeeper (e.g., join/leave the zone)

H.323 channels

• 5 logical channels support a call:
  1. Call signalling channel (Q.931)
  2. Control channel (H.245)
  3. Forward data channel (RTP)
  4. Reverse data channel (RTP)
  5. Data control channel (RTCP)
A timeline of a VoIP call (PC to phone)

1. PC discovers gatekeeper by broadcasting a UDP "Gatekeeper Discovery" packet (pt 1718)
2. Gatekeeper responds with IP address
3. PC registers with gatekeeper by sending a RAS message
4. Gatekeeper ACKs registration
5. PC sends gatekeeper a RAS message to request bandwidth
6. Gatekeeper ACKs bandwidth request
7. PC sets up TCP connection to gatekeeper for call setup
8. PC dials by sending gatekeeper a SETUP message with the number to be called*
9. Gatekeeper responds: CALL_PROCEEDING and forwards SETUP to Gateway
10. Gateway makes a POTS call to the desired phone number
11. End office sends ALERT when ringing starts – forwarded back to calling PC
12. End office sends CONNECT when connected – forwarded back to calling PC
13. Ends negotiate call parameters (via H.245)
14. Forward and reverse data channels set up
15. Call proceeds

* If the destination was another PC, this would be the IP address

VoIP: SIP
Session Initiation Protocol (SIP)

- RFC 3261
- Designed by IETF as a lightweight alternative to H.323
- Not just VoIP – general session management protocol for multimedia

What SIP is

- A single-module protocol for setting up, managing and terminating sessions (data transport is left to other protocols, such as RTP/RTCP)
- App-layer
- Runs over UDP or TCP
Some SIP features

- Phone numbers = URLs (e.g., sip:hostname@cs.uah.edu)
- One-to-one, conference, and multicast calls
- Locate user if not at “home”
- Call waiting
- Call screening
- Encryption
- Authentication

SIP communication

- SIPs communicate using ASCII text messages
- First line of message identifies method:
  - INVITE – Request a session
  - ACK – Confirm session
  - BYE – Terminate
  - OPTIONS – Query a host about its capabilities
  - CANCEL – cancel pending request
  - REGISTER – Inform Redirection server of user location
- Succeeding lines of message pass parameters
A timeline of a SIP session (PC-to_PC using UDP)

1. Caller queries callee using OPTIONS to see if it is SIP-capable
2. Caller sends INVITE in UDP packet
3. Callee responds with a reply code (“180” – “ringing”)
4. Callee responds with a reply code (“200” – “accepted”)
5. Caller responds with ACK (3-way handshake)
6. Call proceeds

User location

- SIP “Location Servers” keep track of user’s locations
- When a user moves to a different location, the user registers the new location with the Location Server
- A caller makes an INVITE to a proxy, which looks up the callee’s location on the Location Server
- Proxy can then set up call to user’s current location – proxy acts as relay as call proceeds
H.323 / SIP Interoperability

- TDM Peering
  - One format downconverted to standard Telco TDM, then up converted to the other format

- Session Border Controllers
  - Single device with both H.323 and SIP
  - Direct translation between formats