Real-Time Audio Streaming over IP

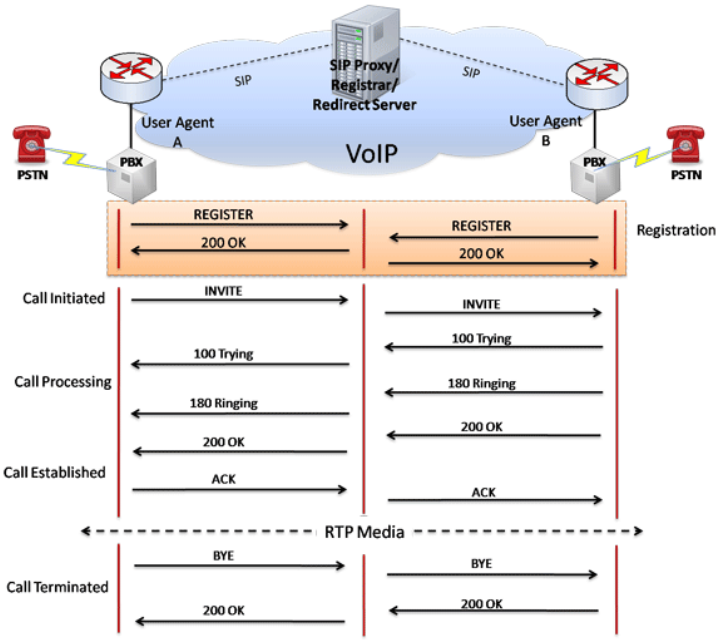
NSCOM01 – MCO2 (Term 2, AY 2025-2026)

Overview

The **Session Initiation Protocol (SIP)** is a widely adopted signaling protocol used to establish, manage, and terminate real-time communication sessions—such as voice or video calls—over IP networks. In a typical VoIP call, SIP is responsible for creating and negotiating session parameters (e.g., codecs, IP addresses, ports), while the actual media (audio, video) flows via a separate protocol called **RTP (Real-time Transport Protocol)**. Alongside RTP, **RTCP (RTP Control Protocol)** provides statistics and control messages (like packet counts, jitter, and periodic sender or receiver reports).

In practical deployments, **SIP** runs over **UDP or TCP**, and **RTP/RTCP** generally run over **UDP**. For many lightweight or real-time voice applications, **SIP over UDP** is common due to its low overhead and simplicity in NAT scenarios. Once a SIP session is established, the two endpoints send RTP packets containing audio or video payloads to each other, while RTCP reports track quality metrics. In this project, students will focus on the audio portion by sending a pre-recorded audio file from one client to another, demonstrating how SIP session negotiation and RTP media streaming integrate into a complete VoIP call flow.

Specifications



SIP Signaling (Over UDP)

- Students must create or utilize a SIP stack that sends INVITE messages, receives 200 OK, and sends ACK to confirm a call.
- The SIP messages should include an SDP body specifying media details (port, codec, etc.).

- At the end of the session, a BYE message will gracefully terminate the call.

RTP & RTCP Media (Over UDP)

- After SIP negotiation, **Client 1** sends audio data (from a local file) to **Client 2** via RTP.
- **Client 2** receives the RTP stream and plays it in real-time (or near real-time) using a basic audio player or a library.
- Periodically exchange **RTCP** packets for minimal statistics (e.g., packet count).

Audio Requirements

- **Client 1**: Must read an audio file (e.g., .wav) or a G.711-encoded file, packetize frames, and send them via RTP.
- **Client 2**: Must receive the RTP packets, decode them if necessary, and play them (or store them, then play).

Basic Error Handling

- If a client receives a SIP error (4xx, 5xx), it should log or handle gracefully.
- No complex recovery is required, but do not crash on unexpected packets.

Additional Notes

- If a client receives a SIP error (4xx, 5xx), it should log or handle gracefully.
- No complex recovery is required, but do not crash on unexpected packets.
- **No SIP Proxy Required**: This project does **not** mandate the deployment of a SIP proxy (e.g., Kamailio, OpenSIPS). The two clients can directly communicate with each other using SIP over UDP.
- **No Registration Phase**: There is no need for a SIP Registrar or user registration process in this simplified project. Both clients explicitly know each other's IP addresses.
- **Local Network Assumption**: The project is assumed to run on a **local network**, avoiding complexities of NAT traversal or public IP routing. If both clients are on the same LAN (or using localhost for testing), direct SIP and RTP communication should work seamlessly.
- Up to two (2) students per group. Sign up for your group at the People tab.

Bonus Points Enhancements

- Real-time microphone capture
- Two-way communication - microphone capture

Deliverables

- April 8, 2025 07:59 AM (HARD DEADLINE, NO LATE ALLOWED)
- Demonstration Date: TBA
- Source Code: The complete source code for this project in a zip archive file.
- README file containing:
    - Instructions for compiling and running the program.
    - A description of the implemented features and error-handling mechanisms.
    - Test cases and sample outputs demonstrating functionality.

Rubric

	Description	Points
SIP HANDSHAKE	Sends INVITE	3
	Processes 200 OK	2
	Sends ACK	2
	All required SIP headers are included	3
SDP Negotiation	Correctly embeds SDP in **INVITE**	3
	Parses **SDP** from **200 OK** to obtain remote IP/port and codec info.	2
RTP Packet Generation	Builds valid RTP headers (sequence, timestamp, SSRC).	5
	Sends audio frames from a file in a timely manner.	5
RTP Receiving & Playback	Accurately receives incoming RTP packets.	5
	Plays the audio (or outputs it properly).	5
RTCP Usage	Periodically sends RTCP (e.g., Sender Reports) and optionally processes Receiver Reports (stats, etc.).	5
Call Teardown	Uses a BYE message to gracefully end the call. Closes sockets and frees resources.	5
Documentation & Clarity	Includes a clear README, usage instructions, and comments.	3
	Code is logically structured, well-documented.	2
	Total	50

Bonus Features

Real-time Microphone Feature	Uses microphone input as the audio source file. Play the captured audio signals on the other client successfully.	5
Two-way communication	Both sides uses microphone audio capture file as source.	5

Useful References

- RFC 3261: SIP: Session Initiation Protocol
- RFC 4566: SDP: Session Description Protocol
- RFC 3550: RTP: A Transport Protocol for Real-Time Applications
- RFC 3551: RTP Profile for Audio and Video Conferences with Minimal Control