Government Controlled Mobile VOIP

Rossi Kamal
Department of CSE
University of Dhaka
Email: rossikamal@gmail.com

Mosaddek Hossain Kamal
Department of CSE
University of Dhaka
Email: tushar@cse.univdhaka.edu

Dr Kazi Muheymin
Institute of Information Technology
University of Dhaka
Email: muheymin@yahoo.com

***Abstract*—VOIP (Voice Over Internet Protocol) is becoming popular day by day. With VOIP, people can make local or internet phone calls with little cost, can take part in audio or video conferences. At the same time, government is concerned about the anti-social activities like terrorism or threats going in phone conversations. There should be a control of government over voice traffic going in IP packet data based VOIP networks. Government Controlled Mobile VOIP presents such a software solution where a government agent can monitor and control the voice data passing between two mobile VOIP clients.**

I. Introduction

VOIP based communication network is taking place of circuit switch based traditional phone system. People can make an audio or video call to home or abroad at the lowest cost using internet. They can enjoy video conference or any kind of real time multimedia based streaming technology. With the expansion of wireless network and mobility, these advantages are encouraging people to think about VOIP based solutions.

Government has become worried in a social security issue that is correlated with threats or similar antisocial activities performed against any individual. People often deploy VOIP services like IP calls or video conferencing in such harmful activities. In some cases government cannot track or record voice conversations between two mobile users. A government server (VOIP gateway server) can set the VOIP session between two mobile users but it might not know what conversation is going there. For example, A Symbian C++ mobile client can talk to other mobile client in real time without the involvement of any interim server in a media session.

Government Controlled Mobile VOIP gives governments control over IP based mobile phone conversations. In our developed solution, a VOIP gateway server is acting as a government agent. This agent can monitor, control or record conversations of two mobile VOIP clients. These J2ME based clients can talk to each other in a VOIP media session.

Our software has separate solutions for both HTTP and RTP based government agents. In first solution, a J2ME client captures voice of caller and sends to an HTTP based government agent. The receiver J2ME client plays it from HTTP agent and receiver can listen to sender. In second solution, the agent is a RTP streaming server that streams senders voice. The receiver is a J2ME RTSP client who listens to server using RTSP protocol. Second solution enables receiver to listen almost in real time. The agent can track and record users, conversations and call related detail information (bit rate, data sent etc).

II. Detailed Description

When two VOIP clients want to talk to each other, they first go through a signaling session with the presence of an interim VOIP server. Then VOIP clients talk to each other in media session with HTTP or RTP. VOIP protocols differ in their signaling sessions those set a secured communication path between two clients. Just after the signaling session, VOIP clients transfer media data to each other.

Let us consider that a caller has established a VOIP session with a receiver by a signaling session. Now they are ready to talk to each other in media session. Both of them are using GOVERNMENT CONTROLLED MOBILE VOP client. First, let us see first solution based on HTTP based Government agent. Then we will move on to RTP based approach.

A. HTTP Based Solution

Our HTTP based solution ensures government agents' control over mobile clients in a HTTP-based VOIP media session.

The solution includes software components at both mobile and server end. Mobile-end components are MySender (J2ME application at caller end to capture voice and to upload to server), MyListener (J2ME application at receiver end to receive and play callers voice). The server end component is MyUploader (PHP script to upload voice to HTTP server). We have used open source apache web server as the HTTP server.

1) Voice capturing at caller end: When a caller talks on phone using internet, GOVERNMENT CONTROLLED MOBILE VOP client will capture and record the voice data [Figure 1]. Let us see what happens inside.

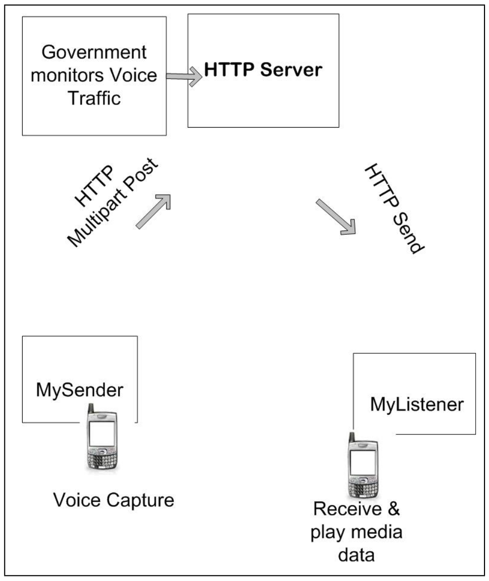


Figure 1: Two Government Controlled Mobile VOIP clients are talking in media session of VOIP with presence of a government agent.

MySender application captures the voice data by creating a player using Manager class of MMAPI (Mobile Media API) [Figure 2].

```
p=Manager.createPlayer(”capture://audio?
        encoding=pcm rate=8000”);
```

Figure 2: Capturing voice with MySender

The voice is captured with pcm encoding, a baud rate of 8000.

MySender then records the voice data by getting control of record and then by setting the record stream

```
rc=(RecordControl)p.getControl(”RecordControl”);
```

Figure 3: Recording voice at MySender.

2) Uploading voice data to server: MySender uploads recorded data to HTTP server using HTTP-multipart post. [Figure 4]

```
HttpMultipartRequest req = new HttpMultipartRequest
   ( ”http://localhost/MyUploader.php”, params,
 ”uploadfield”, ”my.wav”, ”audio/wav”, fileBytes );
```

Figure 4: Uploading from MySender

Algorithm:

1. Create byte array from the recorded audio data.
2. Add header parts to the start and end of the data.
3. Create HTTP Connection to the server.
4. Define multipart-property of the data to be sent [Figure 5]

```
hc.setRequestProperty(”Content-Type”,
  ”multipart/form-data; boundary=” +
       getBoundaryString()); );
```

Figure 5: Setting multipart property of data.

5.Define HTTP post type

```
hc.setRequestMethod(HttpConnection. POST);
```

Figure 6: Defining HTTP Posttype

3) Receiving playing media data from server with HTTP : GOVERNMENT CONTROLLED MOBILE VOIP at receiver end has an application MyListener that receives audio data sent from the HTTP server and plays it to the receiver. So, receiver can listen to what sender is saying to him. MyListener creates a player specifying the url of HTTP server. [Figure 6].

```
player=Manager.createPlayer
  ( http://localhost/my.wav);
```

Figure 6:Creating player at receiver end by MyListener..

Algorithm:

1. Set an alert with a delay as we are loading media data from HTTP server. 2. Create a player of Manager class of MMAPI with the url of HTTP server as locator argument 3. Create event handler to control (start, stop etc) player.

But, streaming over HTTP is an inefficient solution. Because, HTTP is based on Transmission Control Protocol (TCP). TCP is concerned about whether media data has reached destination reliably, it does not consider when the data has been delivered from source or has reached destination. On the other hand, RTSP is based on both User Datagram Protocol and TCP. UDP is a connectionless protocol that is concerned about faster transmission of media data rather than reliability. RTSP has built in control mechanism that lets mobile client to play, resume or stop data streamed from server. A RTSP session involves both RTP and RTCP. RTSP is somewhat like a control protocol, but RTP is related with streaming of that data. RTCP, co-related with RTP indicates quality of steaming data to server and receiver.

B. RTSP Based Solution

In RTP based solution,a RTP streaming-based government agent is in control of VOIP media session of two mobile clients.

This solution has complex software component than the previous one. This includes MyRTSPListener (J2ME client at receiver that initiates RTSP session with RTP streaming server). Server software component is a RTP streaming server(In our solution,we have used Darwin Streaming Server).MySender and MyUploader of previous solution are kept in their positions.

MySender captures data and uploads voice data to a Darwin Streaming Server. MyRTSPListener at receiver end initiates a RTSP session with that server. A government agent on streaming server stores voice data and keeps log of conversation. [Figure 7]. As,MySender is same as in previous session, lets move onto MyRTSPListener in detail.

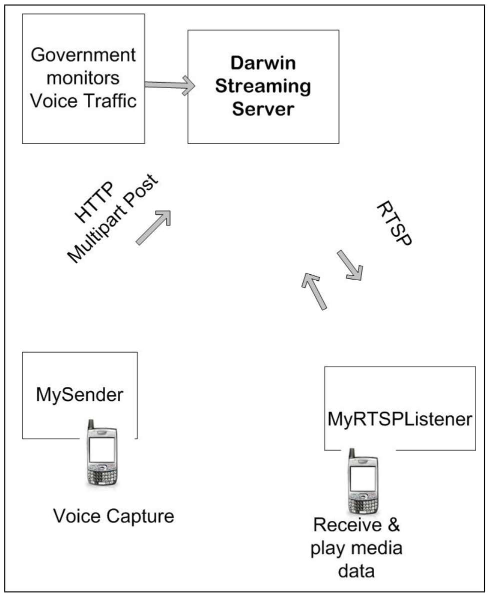


Figure 7:RTP based Government Controlled Mobile VOIP.

MyRTSPListener initiates a RTSP session with the Darwin streaming server by sending a DESCRIBE command. The request is intended to know about information of the media file on server. The URL of the media file contains RTSP version that MyRTSPListener is using, CR/LF. The next line is the sequence number of the request and it increments with each subsequent request made to streaming server. All RTSP commands are terminated by a single line. The session initiation command DESCRIBE is terminated by a single line.[Figure 8]

```
DESCRIBE
rtsp://localhost:554/My.3gp rtsp/1.0
CSeq: 1
```

Figure 8:MyRTSPListener initiates a RTSP session by DESCRIBE command.

MyRTSPListener, on a successful attempt gets a response from server. This response, starting with RTSP/1.0 200 OK, contains several important parameters. As My.3gp is streamable, response contains information of any track of that file. The control string with track id (a=control: trackID) is used to initiate next request to the server.

MyRTSPListener initiates streaming from server by SETUP command using media files' track information. Command includes the transport properties of RTP stream. Our mobile VOIP client, for example, requests server to stream trackid 1 of My.3gp file, to send RTP packets over UDP, to send to our mobile device port 8080. [Figure 9]. RTSP uses 8080 as client port where RTCP uses 8081 for the same.

```
SETUP
rtsp://localhost:554/My.3gp/trackID=1rtsp/1.0
CSeq: 2
TRANSPORT:UDP;unicast;client
```

Figure 9:RTSP SETUP request from MyRTSPListener

Darwin Streaming Server, on successful SETUP request, responses with the session information. An OK response from server means that MyRTSPListener can now initiate PLAY [Figure 10].

SETUP rtsp://localhost:554/My.3gp/trackID=1
rtsp/1.0
CSeq: 3
TRANSPORT: UDP;unicast;client$_{port} = 8080, 8081$

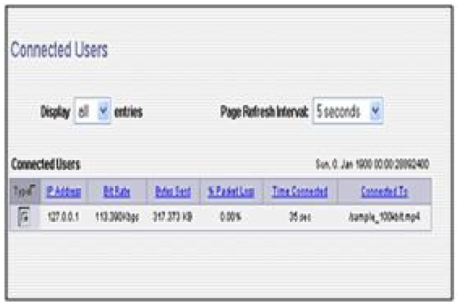
Figure 10:Response from Darwin Streaming Server on SET UP request

Now Mobile VOIP client requests server to start sending RTP packets with PLAY command. This command is issued on media file My.3gp file, not on individual tracks.[Figure 11]. Same is applicable for PAUSE or TEARDOWN request. They are similar to PLAY except the command.

```
PLAY rtsp://localhost:554/My.3gp rtsp/1.0
CSeq: 3
Session: 556372992204
```

Figure 11:MyRTSPListener initiates PLAY command to play My.3gp in RTP.

Government agent on streaming server can watch call information of connected RTSP clients. Information include bit rate of media packets, bytes sent, packet loss , connection time etc. [Figure 12]


Connected Users

Display all entries Page Refresh Interval: 5 seconds

Connected Users Sun, 0. Jan 1900 00:00:20092400

Type	IP Address	Bit Rate	Bytes Sent	% Packet Loss	Time Connected	Connected To
	127.0.0.1	113.390Kbps	317.373 KB	0.00%	35 sec	/sample_100kbit.mp4

Figure 12:Tracking of voice traffic of GOVERNMENT CONTROLLED MOBILE VOIP clients on RTP Streaming Server.

III. Future Work

Government Controlled Mobile VOIP will overcome the overhead of interim server by implementing distributed shared objects. The overhead includes 'keeping huge voice data in storage' and 'streaming of data in HTTP or RTP'. Monitoring agents will be able to share their huge voice data in a distributed system. Streaming workload will be shared by exploring distributed shared objects CORBA, RMI or SOAP.

All media data transfer of our J2ME based software will be in RTP in coming stages. Sending from mobile to server in RTP has not yet been possible in J2ME, but research is going on. As soon as it arrives, our sender mobile client will implement it. Sender will stream his voice to government agent in real time. Our RTSP J2ME client will be updated to hear voice from to RTP server. At present it is performing RTSP protocol with the media data on Darwin Streaming Server. RTSP J2ME client of our software runs on RTSP enabled expensive phone sets. But, we are thinking about possible low cost alternative so that mass people can enjoy RTSP feature with java enabled limited profile- mobiles.

IV. Conclusion

VOIP on mobile is a relatively recent technology considering there are VOIP solutions for web based or desktop platform. Enabling people to make call from remote areas using internet is a cheaper solution. It can be handful for abroad call or distant call. Even in areas where GSM network is not available, this software can be useful for video or audio call using wireless network. Government or any monitoring authority can use this system to look over the voice traffic within its network

Solution thought for overhead problem on monitoring server has moved us toward a distributed approach for the system. We have thought about distributed monitoring system with storage and streaming servers. Why we have to give extra workload on servers? Can't we provide solution on mobile clients? Are the mobile devices capable of storing data or history? If not what can be the alternative? Can't they be used for VOIP protocol stack for the whole system? Going with the questions we have reached a solution. A middleware on mobile should have support for VOIP protocols exploring distributed shared objects SOAP, CORBA and RMI. This will decrease workload on server implementing VOIP protocols. As mobile devices cannot contain massive storage and it will not be trust-worthy, distributed monitoring servers will handle storage of media data.

References

[1] Request for Comments: 2616, *Hypertext Transfer Protocol – HTTP/1.1*, . Network Working Group, IETF.

[2] Request for Comments: 1889 , *RTP: A Transport Protocol for Real-Time Applications*, Audio-Video Transport Working Group,IETF.

[3] Request for Comments: 2326 , *RTSP: Real Time Streaming Protocol* , Network Working Group ,IETF .

[4] JSR 135, *MMAPI:Mobile Media API* , . Sun Developers Nework, Sun Microsystems

[5] Vikram Goyal, *Pro Java ME MMAPI, Mobile Media API for Java MicroEdition*, . Appress.