X-Communicator: Implementing an advanced adaptive SIP-based User Agent for Multimedia Communication

Shakil Siddique, Raimund K. Ege and S. Masoud Sadjadi
School of Computer Science
Florida International University
ssidd003@cs.fiu.edu

Abstract

Our paper describes a SIP and RTP based software component which is platform independent and is adaptable to various network infrastructures. The software component, named "X-Communicator" is capable of multimedia communications including audio, video, text (instant messaging), secure file transfer and desktop streaming with the last three features currently being implemented. The goal is to provide a complete remote-assistance solution and to make the system capable of adjusting to dynamic environment changes. X-Communicator features NAT handling capability and uses intelligent network detection components to identify network infrastructure. For the multimedia RTP data stream transmission it utilizes P2P streaming. The system uses SIP for transferring various control information and implements security. X-Communicator is very user-friendly and can interoperate with other SIP based standard user agents (soft or hard phones) and adjusts its communication capability accordingly. The paper proposes different possible network environments and the software's capability. X-Communicator is currently implemented as a functional prototype and is undergoing further development.

1. Introduction

Voice over IP has become a steady and popular stream of internet community and there are numerous frantic ongoing developments in this area. Session Initiation Protocol (SIP) and H.323 are the backbone for establishing, managing and terminating internet sessions between two or more User Agent Clients (UAC) in these IP based networks. The UACs may be soft-phones (e.g. X-Lite, MSN V 5.x etc.) or hard-phones (e.g. CISCO 7960). In case of soft-phone the scope of development is huge since many services may be incorporated with the system as add-ons. Currently there are quite a few notable development works that are already available for use (e.g. SIPC from SIPquest, SIP-Communicator from Apache Software Foundation) and a number of them are in progress. In SIP based network there is also another important entity, User Agent Server (UAS), but we focus on the UAC soft-phones.

In this paper we discuss about features of a SIP based collaborative tool namely X-Communicator (XC) and compare its functionality and differences with other traditional soft-phones. The system works with any traditional SIP servers and ensures basic soft-phone features, such as call initiation, management, simultaneous call handling (conferencing) and also some other features, such as presence, address book management, desktop streaming, file transfer, text messaging etc are in process of development and integration. The system relies on SIP messages for registration, call management and implements audio video stream transfer using Real Time Protocol (RTP) in a peer to peer architecture, so that there is no RTP traffic in the server side. It is designed to adapt to different network environments, such as open internet traffic, behind router, which would be discussed in detail in section 3 etc. The system is designed in such a manner that it can adapt and implement different corporate policies in the future. XC is also adaptable to network changes, it can handle mobility i.e. network changes. Different existing SIP soft-phones have router and private network handling capability, but to some extent only. It has been observed that most of the systems fail to work properly in the presence of restricted-cone Network Address Translator (NATs), one of the most widely used variant of NAT's. XC is designed to handle this kind of network environments and operate accordingly.

Our effort is to implement most of the features that are available in different SIP based UACs onto XC, enable adaptability in different aspects and some new features of collaborative tools such as calendar systems that are not available in the existing ones.

Next section provides some information about the terminologies used throughout the paper. Section 3 describes how the system should behave in brief and its features. Section 4 explains some important aspects of the implementation techniques and section 5 is the conclusion and future work related discussion.

2. Background and Terminologies

The paper presumes that reader has knowledge about IP based network, client-server model, and concept of sessions. In this section we describe possible network environments and how the terms used in this paper fit into the environment.

We have chosen SIP over UDP as the communication protocol for the XC, since there is less communication

0-7803-8865-8/05/$20.00 ©2005 IEEE.

overhead compared to H.323. We describe the SIP network as an overlay on the IP based network. There are various network configurations possible. For simplicity we choose a simple client-server environment where UACs are of both forms soft-phones and hard-phones. We have chosen SIP Express Router (SER) as our SIP server (UAS), since it provides most of the services to support XC. It is notable that an SER can be used as SIP registration server, a redirect server and as a proxy server. The registration server is used for user registration, where the user location is stored in a location database, which can be trivially implemented using a relational database such as MySQL or an LDAP server. The other functionality of a SIP server as a proxy or a redirect server is beyond the scope of this paper, but interested reader are encouraged and may refer to [3][4] for further information.

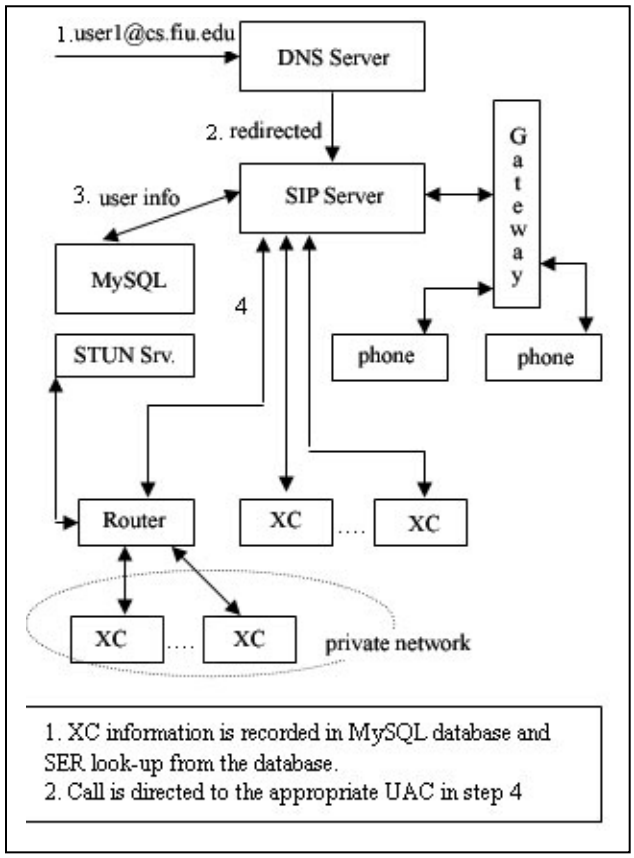

Figure 1: Possible Network Infrastructure (all users are registered to the SIP server.)

SIP [RFC 3261] is being developed by SIP working group, Internet Engineering and Task Force (IETF). It is a text based request-reply signaling protocol used for initiating, terminating and controlling sessions between end-users in the packed-based network.

We also consider the presence of a router in the network as depicted in the figure 1, which is quite common in most of the typical network configurations. The notion of router brings the concept of NAT, where a private IP based network is defined behind the router. This private network identifies its clients through their unique IP addresses that are invalid in the actual internet. The open internet is the network where each client has a unique public IP address. Thus, when a client in a private network want to communicate outside the router in the public internet, the router assigns a particular public IP address for a timed session against the private IP address it has. The process of mapping the public address to private is network address translation (NAT). Once a session is recorded in the router, any incoming reply to the NATted client is routed as mapped in the router sessions. There are different flavors of NAT, namely full-cone, restricted-cone, restricted-port-cone and symmetric NAT. We discard the usage of symmetric NATs since SIP has restrictions and cannot be used if UAC is behind such a translator. More information about NATs can be found in [5][6]. We elaborate our discussion on the other type of NATs later in section 3 where we propose a sample algorithm that helps to handle NAT traversal.

The network configuration also considers STUN [10] servers where the acronym stands for simple traversal of UDP through NATs [6]. If a client issues a STUN message request to such a server, the server reveals the public address assigned by NAT with a message reply. Thus, a UAC may identify any port mappings to public internet, using STUN requests and obtain the public port mapping accordingly. Once the public port mappings are known to the client, the client may use the SIP messages using *contact headers* to inform the other client of its intended port to receive data and be able to send and receive accordingly.

XC uses Java Media Framework (JMF) from Sun Microsystems to send and receive audio and video data stream. The multimedia stream is exchanged using Real Time Protocol (RTP) [7] and its implementation using Java language is trivial.

In the next section we discuss various features available in XC and its implementation design technique.

3. System Features and Implementation

3.1 Common features

The XC offers many services apart from that of available UACs'. The common features include text messaging, audio and video communication, file transfer, authentication and presence information.

The implementation of text messaging is trivial and is implemented using SIP messaging architecture.

The audio and video stream transfer is implemented using JMF and are transferred in a P2P fashion with other UACs. The media stream detection is event-driven, so any

such stream is automatically identified and is compiled accordingly. In case of data stream security, a basic formalism is applied, such that, the port used for audio or video stream is selected during the time of initial call set up using SIP and Session Description Protocol (SDP) messages, so that only intended participants are aware of the information. JMF also allow using different types of data stream from its rich library, so a particular SIP session may be pre-negotiated before data transmission between two or more user agents. The SDP is used in SIP request messages to exchange information about media-stream capabilities for both UACs in a session. For example, in the case of audio data stream, G77 ulaw stream may be chosen between an XC and X-lite UAC. Similarly the video stream may also be chosen as H.261 or JPEG/RTP. The basic system architecture is depicted in figure 2.

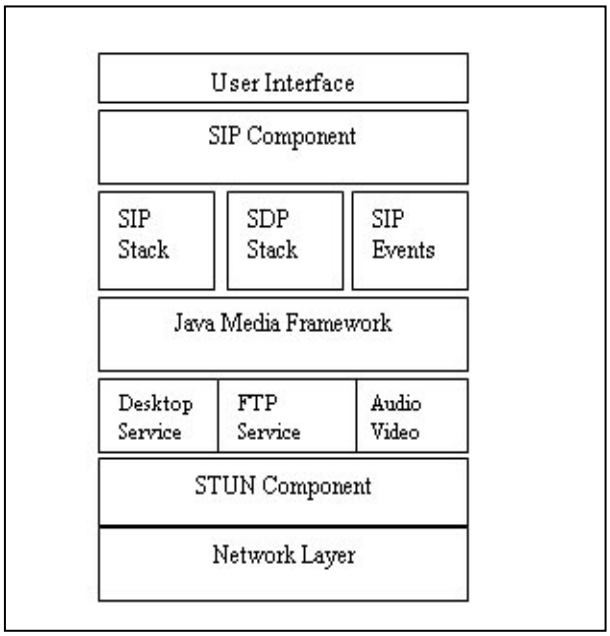

Figure 2: System Architecture

XC is currently using SIP and SDP stack available from NIST (http://www.nist.gov) known as NIST-SIP-API and NIST-SDP-API.

The file transfer operation is currently implemented using pure P2P through sockets, but the implementation can be changed depending upon requirement. For example, JMF may be used to send and receive file using file object as custom data source.

3.2 Remote assistance

The system also offers other services as add-ons, which we discuss below. XC has remote assistance, where a UAC may request to view other UAC's desktop view, so that any information can be shared between the participating agents in a session. For example, if a user needs to have guidance in terms of direction in a location, the other participant may transfer a virtual reality map of the location, and using the remote assistance feature, users may collaborate to a desired solution. This feature is applicable to any other application or document related remote assistance, since a steady video-stream is generated from the desktop screen-shots and is transmitted using the JMF to the other agent. Therefore any information is virtually shared between the end users.

Java API has the *robot* class in AWT package, which may create current screen-shot of desktop. A steady stream of such screen-shots may be generated and fed as a custom data source in the JMF. The media manager can transmit desktop stream as JPEG/RTP or H.261 format. For high quality purpose, the first format may be chosen, but in terms of efficiency, H.261 is widely chosen.

This part of development is in progress. As XC is designed to be modularized, we have developed a prototype that provides desktop streaming service, and would be integrated as an add-on onto XC.

3.3 Adaptability

XC is adaptable to network changes and can detect and manage mobility. In real-world a user may travel from one location to other using a hand-held device, changing different subnets. In this case, the IP address also changes accordingly, and XC is able to handle the environment change.

During the registration process of XC with a SIP server, user credentials are cached into the system, where the information is obtained from authentication process, and a separate thread for re-registration process is launched. During a call establishment and during call management, XC performs network layer check by means of method invocations to STUN compatible component. If the different IP information is returned then the re-registration thread updates the necessary port and IP information to the SIP server and issues SIP and SDP message requests accordingly. Thus the network mobility is transparent to the user and user credential is recorded only once.

3.4 NAT handling technique

The system also is designed to operate behind most of the NATs and is capable of maintaining steady communication without user intervention. As mentioned before, there are different flavors of NAT and we consider all types except the symmetric version.

XC uses different port for each service. For example it may use port 5080 for registration, 80080 for FTP. Let us consider a scenario where one UAC XC 1 communicates with another one, XC 2. To use each port behind a NAT, XC 1 must first find out the public port mapping for each

port in order to send and receive data using that port. Once the public port mapping has been identified, this information is sent to the other client using SIP message, and a dummy session is created using the private port number to the using sample data stream to establish session with the router. The purpose of this dummy session is to record the port mapping in the router and to keep this information in a periodic basis before the actual data stream arrives from the other UAC (XC 2). This is necessary, because if the data is transmitted from XC 2 to XC 1 prior to dummy session establishment (although the receiver knows its public port mapping from the STUN server) the data would be lost since the router does not have the port mapping for the private port (yet), hence cannot re-transmit the data over to the desired client.

In a generic scenario, the SIP server, router and STUN server are in public internet.

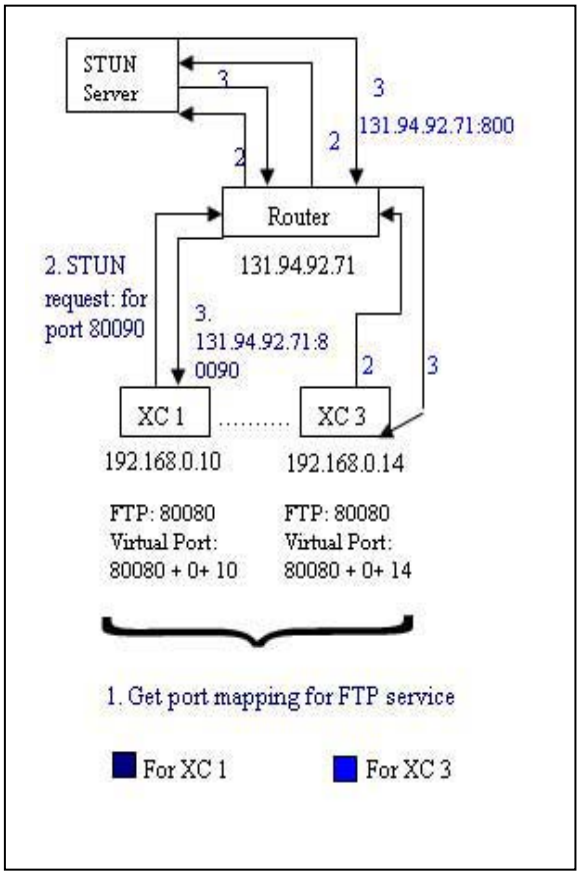

Figure 3: NAT handling by X-Communicator in port-restricted-cone version.

In case of full-cone NAT, each private port is assigned a unique public port by the router. In an actual implementation let us consider XC 1 and XC 3 behind the same router whose IP is 131.94.92.71, a public IP address. If the private IP address of XC 1 and 3 are 192.168.0.10 and 192.168.0.14 respectively and they have FTP service enabled, which uses port 80080, then a typical public port mapping returned by the STUN server could be 131.94.92.71:62962 and 131.94.92.71:64964 respectively, where the initial IP address of the public port mapping is that of the router. Therefore, using dummy session, both these UACs can operate properly.

However, in the case of restricted-cone NAT, the scenario is different. The router does not provide unique public port for each private port number, instead it maps its own public IP and the private port of each client and returns the information as a public port mapping for both the clients. Using the above example, the practical output would be 131.94.92.71:80080 for both the clients, which is not desirable. Since the later port mapping would override the previous port mapping, and the router would only know the later port mapping information. The ultimate result would be that the data intended for XC 1 would be transmitted wrongly to XC 3.

To handle this problem, we propose to provide an algorithm that would use the private IP address and port of a UAC as function parameters and generate private ports that would be unique in public domain, but different for the same service in the UACs in same private network. This approach would solve the problem of port overriding in the restricted-cone NAT and would still be applicable and valid in other contexts, such as full-cone and open internet.

Figure 3 illustrates a sample technique using the example given above.

If we consider the case of randomly generating port numbers for different services, then there might be a situation, where two or more UACs behind same router might get same port mappings, resulting communication flaw. However, using this scheme as described above, we can avoid such situation. The approach described above is only a sample technique, but illustrates the concept of using internal IP address and port numbers to generate unique public port mapping. This feature is being implemented currently.

3.5 Multiple terminal usage

We also propose multi-device communication in this architecture, where a single user may use UAC hard-phones, such as CISCO 7960 for audio and XC for video and other features simultaneously. The idea is to provide flexibility of using audio conversation using VoIP phones to user during a single virtual session.

During an ongoing SIP session between two UACs, user 1 may want to use available phones. Using address book features user can connect to a VoIP phone in a separate SIP session with user 2 having the hard-phone's information in the contacts, so even though actually there are two separate session, user would get the flavor of using one virtual session utilizing multiple terminals. Once the XC connects

establishes the session with a hard-phone of user 2, the soft-phone would disconnect audio session and keep the existing session with user 1 for other features. This feature is not yet implemented.

4. Functional Flow

In this section we present the brief functional flow of different processes in the XC implementing the features. We start with the registration process and follow through establishment towards termination of a call.

XC uses XML based system configuration and the address book or contact information can be stored locally or in the server, depending upon environment. For example SER provides contact information to be saved in the server side, where as VOCAL, another SIP server from vovida.org do not support this feature.

Once the system is initialized, the STUN component (as shown in figure 1) detects network configuration, and obtain necessary port mappings using a STUN server for all the services that are enabled. XC then registers itself with user provided authentication credentials to desired SIP server. At the same time, the presence module is initialized and using the contact list necessary presence messages is transmitted and received from the presence module. The multimedia component that comprises of JMF is then initialized and system becomes ready for communication.

User may select SIP addresses from address book that is displayed in the presence pane or provide SIP URI in the system and initiate call. During call establishment XC obtains information about the other party's capability, for example audio and video codec, file transfer capability, port information etc using SDP descriptions. Once a call is established XC obtains all the information using SIP messages about service requirement. For example, if both the UACs are XC and remote assistance service is required, XC obtains port information using STUN component and sends information to the other party. Similarly the port information is obtained from the other party. Once this information is obtained, XCs establish peer to peer communication to transfer data stream in both direction. However, the control information is exchanged using SIP.

When a call is terminated, the media manager component terminates the media session and then the SIP component terminates the SIP session.

5. Conclusions

As mentioned before there are numerous possibilities to extend services in this architecture. We have two versions of above implementation namely *X-Communicator* and *Oghma* which are still undergoing development for full-functionality. Current systems provide the basic functionality and remote assistance service and are adaptable to different network environments.

In the future versions, we intend to implement several more functionality for optimization and quality of service (QoS). For QoS transferring the video stream using compression technique would make the system more efficient for congested networks.

Similar technique can be applied to the desktop stream transfer. Also we plan to provide *transparent-board* application integrated into the remote assistance service so that both users can draw on the stream pane for better means of communication. This feature can be achieved by drawing objects on one client side on the display pane and the generate SIP messages to record and transfer co-ordinate information for each points of drawing and transfer the data to the other side, so that the image can be re-drawn on the display pane. The functionality of this feature is similar to MSN's whiteboard application, except it would provide drawing objects on image stream with different colors for different users.

In case of FTP service, there could be an option of selecting multiple files and compress them onto a single file for better network utilization. At the same time, file can be encrypted, so that multiple files can be transferred in a secured way between XCs.

Another possible implementation is calendar system, another collaborative tool. This feature is popular in web-based systems such as Yahoo or MSN. The idea is to incorporate this feature in a P2P level using XC. Having proper calendar data entry, users may collaborate for scheduling conversations or other related resource exchange.

The XC supports conferencing capabilities, but to some extent. Multiple user may connect to a single XC with audio-visual data streams, but all participating users do not have each other's data stream except for the originating XC's in consideration. The other version of XC, Oghma supports conferencing and do not have this problem. However, a more efficient implementation can be achieved using minimum data stream cloning and redirecting the stream to participating users. Where each user may support this feature, and stream can be distributed efficiently. If stream compression technique is applied then we can achieve higher QoS.

More information may be obtained from

- http://rocksteady.cs.fiu.edu/voip
- http://rocksteady.cs.fiu.edu/ssa/oghma/

6. Acknowledgements

This material is based upon work supported by the National Science Foundation under grant No. HRD-0317692.

7. References

[1] Henning Schulzrinne and Elin Wedlund "Application-Layer Mobility using SIP" *Mobile Computing and Communications Review (MC^2R)*, Volume 4, Number 3, July 2000.

[2] Phelim O'Doherty "Identify the specifications to the Session Initiation Protocol (SIP) defined through the Java Community Process (JCP)" *Sun Microsystems*, SIP Specifications and the Java Platforms White paper. http://www.cs.columbia.edu/sip/Java-SIP-Specifications.pdf

[3] RADVision "What is SIP" *RADVision white paper*. http://www.radvision.com/ResourceLibrary/WhitePapers/

[4]Network Working Group "SIP RFC" *http://www.faqs.org/rfcs/rfc3261.html* Internet RFC/STD/FYI/BCP Archives

[5]http://www.newport-networks.com/whitepapers/nat-traversal.html

[6]Network Working Group "NAT RFC" http://www.faqs.org/rfcs/rfc1631.html, Internet RFC/STD/FYI/BCP Archives

[7] Xiaotao Wu, Henning Schulzrinne **"**SIPC, a multi-function SIP user agent", Columbia University, Department of Computer Science

[8] Sun Microsystems "JMF API guide" available from http://www.sun.com

[9] http://snad.ncsl.nist.gov/proj/iptel/

[10] Network Working Group IETF "STUN RFC", http://www.faqs.org/rfcs/rfc3489.html, Internet RFC/STD/FYI/BCP Archives

[11] Henning Schulzrinne, Jonathan Rosenberg "The Session Initiation Protocol: Internet-Centric Signaling"

[12] M. Handley and V. Jacobson "SDP RFC (RFC2327)" *http://www.faqs.org/rfcs/rfc2327.html*