Efficient Link Utilization of SIP Proxy Server for Reducing Load in FoIP Application

By
Sarvakar Ketan J.
(06MCE014)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY,
AHMEDABAD 382481
MAY 2008

Major Project

On

Efficient Link Utilization of SIP Proxy Server for Reducing Load in FoIP Application

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Sarvakar Ketan J.
(06MCE014)

Under Guidance of

Dr. S.N. Pradhan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY,
AHMEDABAD 382481
MAY 2008

This is to certify that Dissertation entitled

Efficient Link Utilization of SIP Proxy Server for Reducing Load in FoIP Application

Submitted by

Sarvakar Ketan J.

has been accepted towards fulfillment of the requirement

for the degree of

Master of Technology in Computer Science and Engineering

Prof. (Dr.) S. N. Pradhan	Prof. D. J. Patel
Professor	Head of The Department

Prof. A. B. Patel
Director, Institute of Technology

CERTIFICATE

This is to certify that the work presented here by **Mr. Ketan Sarvakar (06mce014)** entitled "**Efficient Link Utilization of SIP Proxy Server for Reducing Load in FoIP Application"** has been carried out at **Institute Of Technology, Nirma University** during the period **September 2007 – May 2008** is the bonafide record of the research carried out by him under my guidance and supervision and is up to the standard in respect of the content and presentation for being referred to the examiner. I further certify that the work done by him is his original work and has not been submitted for award of any other diploma or degree.

Dr. S. N. Pradhan
Professor,
Department of Computer Science and Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

Date: / / 2008

ACKNOWLEDGEMENT

The success that I have got in the accomplishment of the project work is not only due to my efforts. In fact, I stand on the shoulders of many people. So, with great pleasure I take this opportunity to express my gratitude towards all the individuals who have helped and inspired me in my project work. It was a great experience working on this project, about a totally new concept. The project taught me many things and added knowledge to my memory bank.

First of all, I would like to express my earnest gratitude to my project guide **Dr. S N Pradhan**, (P.G. Coordinator, CSE) Department of Computer Science and Engineering, Nirma University for their constant guidance, encouragement and moral support which helped me to accomplish the project.

I would like to convey my sincere thanks to external project guide, **Mr. Deepak Kakadia**, CTO & Vice President of Data Centrica (Advanced IT Technologies Pvt. Ltd.) for providing me such a good project definition and continuous support.

I would like to thank **Prof. A. B. Patel**, Director, Institute of Technology, Nirma University of Science & Technology, Ahmedabad for providing me the facilities in Nirma campus.

I am thankful to all the faculty of Computer Science and Engg department, Nirma University of Science & Technology, Ahmedabad for providing suggestions with crucial feedback that influenced me to complete this work.

The blessings of God, my Guruji and my family members make the way for completion of major project. I am very much grateful to them.
Last but not the least, I am equally thankful to all my friends for everything.

Finally, I would like to thank my friends for their coordination.

Sarvakar Ketan J.

(06MCE014)

Abstract

It’s been some time now that expressions like “Voice over IP”, “Fax over IP” and the likes are heard extensively in the telecommunications industry. The idea is utilizing data networks to deliver telecommunications services which are currently provided by the PSTN. The incentive is pretty straightforward: cutting costs and yet being able to provide the previous services, not to mention the added capabilities to deliver a multitude of other services, hardly imagined feasible with the PSTN.

When considering the implementation of the aforementioned objective, one faces a lot of difficulties. Simply put, the current data networks, e.g. the Internet, have not been designed with telecommunications services in mind. They have been optimized to carry data which is bursty in nature. This design is in obvious contradiction to the requirements of the telecommunications services, one of which is fax. In this dissertation, Fax over Internet protocol (FoIP) is being considered which has two possible approaches to be accomplished: Real-time and Store-and-Forward. Real-time approach is the ultimate goal since it is the real-time faxing which makes the transition from the PSTN to the Internet-based architecture smooth.

Signaling comprises initiation, management and tear-down of sessions examples of which are fax, voice, video and the like. Currently there are two protocols that can provide an end-to-end solution: H.323 and Session Initiation Protocol (SIP). SIP is the protocol of choice among other standards in the voice and fax transmission domains due to its numerous advantages.

In this dissertation, It is intend to closely examine some aspects of the new architecture and its implementation feasibility. Different components of the real-time Fax over IP architecture are analyzed and attention has been paid to the signaling part. Utilization of SIP in conjunction with SDP, as companion protocol to

SIP for capabilities to exchange for fax transmission has been studied. What It is intend to do is exploring whether fax parameters details can be negotiated using SIP/SDP. Session establishment, starting a sample file transfer, which can act on behalf of real-time fax transfer, and the subsequent session tear-down, after file transfer is complete, are demonstrated. This simulation scenario and its results exhibit the potential success of the proposed SIP/SDP combination for real-time fax session establishment, management and tear-down.

Another important analysis carried out in this dissertation is the utilization of SIP contact header for reducing the load on proxy servers which is a highly desirable feature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 INTRODUCTION

“Voice over IP”, “Fax over IP” and the likes are gradually becoming the next big cutting-edge technologies in the telecommunications industry. These are meant to replace the traditional method of delivery of telecommunications services by Public Switched Telephone Network (PSTN) through utilization of data networks e.g. the Internet. By doing so, both the telecommunication service providers and the users can save fortunes, not to mention the newly-presented capabilities to deliver a multitude of other services. Venturing into the actual implementation has proved to be a hard-to-overcome challenge and a plethora of standards are still being considered to make the new architecture a reality.

1.2 EXISTING PROBLEM

When considering the implementation of the aforementioned objective, one faces a lot of difficulties. Simply put, the current data networks, e.g. the Internet, have not been designed with telecommunications services in mind. They have been optimized to carry data which is bursty in nature. It means a discrete series of packets of data which travel through the net from a source to a destination with frequent idle times in transmission. It is not strictly continuous and generally the users don’t mind the jitter and extended delays of the data packets. These are in obvious contradiction to the requirements of the telecommunications services. Specifically, they stipulate the existence of a network infrastructure which is either connection-oriented in nature or at least can resemble its behaviors and therefore is able to guarantee a stream of data free of any kind of interruption. One of these telecommunication services is Fax. Fax over Internet protocol (FoIP) is being considered in this dissertation but needless to say that most of the technologies are shared with Voice over IP (VoIP). There are two possible approaches to accomplish faxing over the internet protocol:

Real- time and Store-and-Forward. Store-and-Forward or non-real-time usually uses E-mail capabilities to transfer fax between the end-points. In real-time approach, as the name suggests, fax is transferred in real-time manner and without delay; like the way it currently sends a fax using the PSTN. Real-time approach is the ultimate goal since it is the real-time faxing which makes the transition from the PSTN to the Internet-based architecture smooth. Signaling as one can imagine is the most important part of any session initiation, management and tear-down whether it is fax, voice, video or the like. Currently there are two protocols that can provide an end-to-end solution: H.323 and Session Initiation Protocol (SIP). H.323 is a binary protocol which consists of a complex suite of protocols that reuse many older services and methods borrowed from Integrated Services Digital Network. Being binary, among other short-comings of H.323 in comparison with SIP which is text-based, makes H.323 a platform dependant protocol. SIP, in addition to other advantages which are further discussed in the coming chapters, is capable of supporting user mobility by proxying and redirecting requests to the user's current location. Hence, SIP is the protocol of choice among other standards in the voice and fax transmission domains. But SIP has yet to prove its capabilities to be chosen as the ultimate solution for providing fax services in the Internet.

1.3 DISSERTATION OBJECTIVES

In this dissertation , It is intend to explore the issues related to Fax over IP in their entirety and closely examine some aspects of the new architecture and its implementation feasibility. Different components of the real-time Fax over IP architecture are analyzed and attention has been paid to the signaling part. Utilizing SIP in fax transmission is studied and feasibility of implementing Fax over IP architectures using this protocol is discussed. Based on the reasons briefly mentioned earlier, SIP is better suited to the task in comparison with H.323. Computer simulations are utilized for the analysis; specifically, a simplified version of the SIP protocol and network components are developed to study whether fax parameters details can be negotiated between the end-points. These simulation scenarios and their results exhibit the potential success of proposed implementation approach.

1.4 DISSERTATION ORGANIZATION

This dissertation comprises 7 chapters. Chapter 2 has been devoted to introducing the fundamental concepts behind every network. Readers with sufficient background in network technology can readily bypass these chapters and move on to Chapter 3. Specifically, in Chapter 2 the discussion has been started by inspecting the Public Switched Telephone Network (PSTN) infrastructure and architecture and then move forward to study Internet Protocol will be examined fully in the third section, because of their prime importance in this dissertation. The chapter is concluded with a thorough treatment of the Internet protocol itself and its transport layer protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

In Chapter 3 The issues are treated directly involved in this dissertation: issues related to V/FoIP. In this chapter, the first section, Fax over IP is treated by first introducing the conventional Group 3 T.30 fax transmission, then T.37, ITU-Transfer of facsimile data via store-and-forward on the Internet recommendation, and T.38, ITU-T real-time Group 3 fax communication over IP networks recommendation, are explained. Various approaches to provide QoS in IP networks are discussed next. At the end of the chapter, a discussion regarding the FoIP trends and economics is presented.

Chapter 4 is devoted entirely to presenting and exploring the Session Initiation Protocol (SIP): The protocol of choice in this dissertation. In Chapter 5, after an introduction, SIP clients and servers, request and response messages and the protocol headers are introduced. Session Description Protocol (SDP), a companion protocol to SIP, is introduced in Section 5. A thorough treatment of SIP programming is presented afterwards in Section 6. The chapter has been concluded with a discussion covering T.38 and SIP utilization for FoIP.

In Chapters 5 and 6, a detailed account of computer simulations is presented. In Chapter 5, J-Sim simulator, a powerful Java-based network simulation tool is introduced. The source code of this simulator is in the public domain, so It takes a close look at inner workings of different components of it, how it operates and how new modules can be developed and added to its set of supported protocols. The simulation scenario creation method has been examined in the simulator.

In Chapter 6, as a major implementation part of this dissertation, the development and testing of the SIP protocol is explained. SIP network architecture components i.e. SIP user agents and proxy servers, among other components, have been developed and added to the simulator. The SIP components are put together to build a network simulation scenario to further analyze the behavior of them. As is explained in Chapter 6, building simulation scenario, node initialization and configuration, addition of measurement tools to track packets in a real-time manner and performing measurements of parameters such as throughput, packet sequence numbers and congestion are made possible with the tool command language (TCL) scripting language. A detailed analysis of the specific chosen simulation scenario is given and measurement results are discussed.

In Chapter 7, it has been wrapped up the whole discussion and summarizes the main points from concept-presenting chapters. Some important analysis results and future work directions are also presented.

Chapter 2

Fundamentals

2.1 INTRODUCTION

In this chapter, fundamental concepts related to this thesis have been introduced and discussed. First it has been started by inspecting the Public Switched Telephone Network (PSTN) infrastructure and architecture and then move forward to study Internet Protocol will be examined fully in the third section. In this Section, transport layer protocol has been discussed very briefly. Then the chapter ends with introducing materials for further studying. Having built the necessary knowledge base in Chapters 2, issues specific to V/FoIP networks will be introduced and examined in Chapter 3.

2.2 THE PSTN INFRASTRUCTURE AND ARCHITECTURE

My views about what a network should be designed to support and what the infrastructure should be comprised of have changed quite a bit over the years, as applications and technology have changed. This section takes a look at how the PSTN infrastructure evolved and where it is today. The traditional PSTN is associated with highly developed, although not necessarily integrated, operational support systems (such as billing systems, provisioning systems, network management systems, customer contact systems, and security systems). These systems have very well-developed business processes and techniques for managing their environments. But the various systems' databases cannot yet all speak to one another to give one comprehensive view.

PSTN Architecture

The PSTN includes a number of transmission links and nodes. There are basically four types of nodes: CPE nodes, switching nodes, transmission nodes, and service nodes.

CPE Nodes

CPE nodes generally refer to the equipment that's located at the customer site. The main function of CPE nodes is to transmit and receive user information. The other key function is to exchange control information with the network. In the traditional realm, this equipment includes PBXs, key telephone systems, and single-line telephones.

Switching Nodes

Switching nodes interconnect transmission facilities at various locations and route traffic through a network. They set up the circuit connections for a signal path, based on the number dialed. To facilitate this type of switching, the ITU standardized a worldwide numbering plan (based on ITU E.164) that essentially acts as the routing instructions for how to complete a call through the PSTN. The switching nodes include the local exchanges, tandem exchanges (for routing calls between local exchanges within a city), toll offices (for routing calls to or from other cities), and international gateways.

Transmission Nodes

Transmission nodes are part of the transport infrastructure, and they provide communication paths that carry user traffic and network control information between the nodes in a network. The transmission nodes include the transmission media as well as transport equipment, including amplifiers and/or repeaters, multiplexers, digital cross-connects, and digital loop carriers.

Service Nodes

Service nodes handle signaling, which is the transmission of information to control the setup, holding, charging, and releasing of connections, as well as the transmission of information to control network operations and billing.

2.3 INTERNET TRANSPORT SERVICE CLASSES

The Internet model basically assumes that the network can only provide an unreliable connectionless service, and only provides two transport classes, TCP, which equates to the ISO Class 4 service, and UDP, which is connectionless. Both these protocols operate over Internet Protocol.

2.3.1 User Datagram Protocol

The User Datagram Protocol (UDP) provides a simple connectionless mechanism for applications to exchange messages. While the fact that no connection is established means that the protocol has very low signaling overheads, it also means that there is no error or flow control. For some real-time services with very low delay requirements like voice transmission, a lack of flow control is an advantage, since any lost data would not be repeated anyway. UDP is also used for broadcast messages since a connection-orientated approach is not then appropriate, and for periodic messages like routing table updates where if the data is lost, it does not matter since the existing data can be retained until the next update. Some services, like DNS, which could use TCP, usually use UDP for efficiency. Rather than wasting the time for setting up a connection, as well as adding to the load of the host, a connectionless UDP request is made. If the request or its response is lost, another DNS server will be tried after a timeout [3][4].

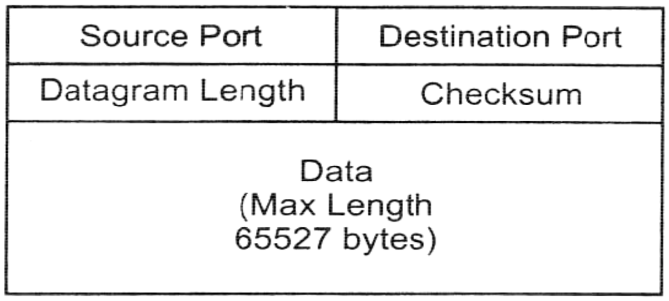


Figure 2.1 UDP packets [3]

The UDP PDU has four 16 bit fields, shown in Figure 2.1, with the source and destination ports referring to application processes on local and remote hosts. The source port is optional; it is set to zero if it is not used. The length field refers to the total number of octets in datagram including the header.

Note that the UDP segment does not include the address of the recipient, only the port number. This is because UDP is designed for transport over IP, and the IP header holds that information. There is still the problem that since the UDP header does not contain that information directly, a UDP datagram could be delivered to the wrong host and the transport layer would be unaware of this fact. To avoid this, the source address, destination address, protocol and length of the IP packet header are considered to form a pseudo-header which is added to the UDP datagram for the purposes of calculating the frame check sequence. If the datagram is delivered to the wrong host, the checksum will fail. The checksum is optional and is set to zero if not required, but if it is used it can be checked by intermediate routers which can drop corrupted packets to save on network load. IP is not restricted to carrying UDP.

2.3.2 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) provides a connection-orientated communications protocol designed to work over IP. Like UDP, TCP allows communication between specific processes on each host. So there can be many different connections between two hosts simultaneously. However, since the identification of a connection is done on the basis of ports on each host, there can only be one connection between a given source port and a given destination port on a pair of hosts. Since IP only provides an unreliable transfer mechanism, and TCP is connection-orientated, TCP must provide mechanisms to ensure that any lost or corrupted data is replaced before delivery to the upper layers. This is done using ARQ mechanisms. TCP has three phases of operation: connection establishment, data transfer and connection termination. A three-way handshake is used in the connection phase because of the unreliable network service. Data is passed to TCP from an upper layer protocol in a continuous fashion. It is then blocked arbitrarily into segments. Being full duplex, the protocol allows data to be sent in either direction between processes.

Chapter 3

Voice/Fax over IP

3.1 INTRODUCTION

In this chapter, the concepts has been discussed and introduced directly related to this dissertation: Voice/Fax over IP architectures and standards. the first section, Fax over IP is treated by first introducing the conventional Group 3 T.30 fax transmission, then T.37, ITU-Transfer of facsimile data via store-and-forward on the Internet recommendation, and T.38, ITU-T real-time Group 3 fax communication over IP networks recommendation, are explained. At the end of the chapter, a discussion regarding the FoIP trends and economics is presented.

3.2 FAX OVER IP

3.2.1 Introduction

Facsimile transmission presents a major implementation challenge in the new packet networks. Voice support is difficult enough, with its own stringent requirements for maintaining the quality of service and user experience of the PSTN, but fax transmission escalates the problem to a higher level. Simply speaking, for the new VoIP networks to succeed, it is mandatory to support the legacy facsimile equipment, which attaches to plain old telephone system (POTS) service. The installed base of fax equipment is so large that the unspoken requirement is for complete support for Group 3 fax at a minimum in any service deployment. A large percentage of homes with access to the Internet have at least simple PC-based fax capability, and as the number of telecommuters grows, this number is expected to increase.

In contrast to the simple transport of voiceband data on the PSTN, packet networks come with many complexities regarding this transmission. Voiceband data is interpreted by machines, and they are far less tolerant than the human ear when things are not according to the highest quality. This is exactly the issue with fax support over packet networks. While the loss of a single packet during a human conversation may go unnoticed, the same lost packet during the handshake procedures of a fax call can result in a dropped call or lengthy recovery procedures with possible drop in speed as the outcome. All this is annoying and disturbing to users, especially if the affected call incurs long distance or international charges and must be repeated.

ITU-T Recommendation T.30 defines the procedures for the transmission of facsimile over the PSTN, which includes end-to-end capabilities negotiation between fax terminals and sending of image data encoded in a standard format. Transmission of images for Group 3 fax is specified in ITU-T Recommendation T.4. The need for an expedient solution to support facsimile in packet networks has resulted in additional specifications that preserve a local nature of Group 3 signaling and transmission. Local signaling executes between a fax terminal and the network equipment to which it connects. The packet infrastructure on the network side of the gateways is then used to transport the image data and re-create fax signaling sequences at each gateway toward their local fax terminal.

There are currently two methods that accomplish this operation and both are standards specified by the ITU. Recommendation T.37 specifies store and forward, non-real time techniques for sending facsimile with legacy equipment connected via a gateway to a packet network. Recommendation T.38 specifies a real-time operation which does not use voiceband data transport, but instead uses packetized information to carry both the handshake sequences and the digitized image data itself between the terminals. Discussions of both those techniques are done in this section.

Both the real-time and non-real-time methods for facsimile support over packet networks have their own benefits and shortcomings, from both a technical and an economic perspective. The availability of two methods to send faxes over packet networks has opened a new market in business segments, where previously there was no choice with POTS service. Both fax transmission techniques are expected to continue to enjoy success in the IP networks and find their way into VoIP service offerings as the technologies mature and their acceptance in the commercial sector increases.

Let's look now at the details of the most common fax transmission methods. The call flow diagram in Figure 3.2 shows a typical error-free fax transmission between an auto-calling and an auto-answer terminal, such as two PCs, or a PC and a fax machine, or two fax machines. This diagram has been used as a basis for explaining the basic fax operation and issues.

A fax call in conventional Group 3 facsimile is completed in five phases:

1. Phase A - Call establishment

2. Phase B - Attributes, capabilities, and control signaling

3. Phase C - Single-page fax transmission

4. Phase D - End-of-page signaling and multipage notification

5. Phase E - Call termination

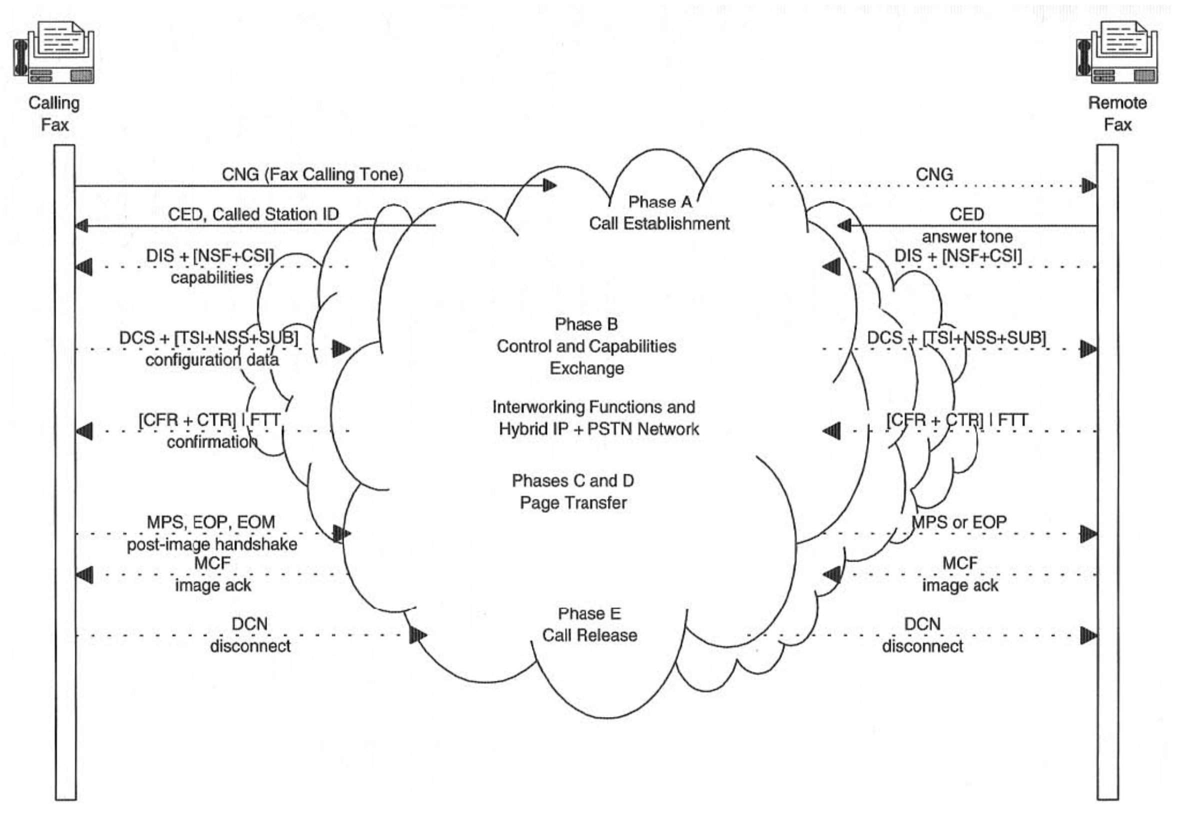

Calling Fax
Remote Fax
CNG (Fax Calling Tone)
CED, Called Station ID
DIS + [NSF+CSI] capabilities
DCS + [TSI+NSS+SUB] configuration data
[CFR + CTR] | FTT confirmation
MPS, EOP, EOM post-image handshake
MCF image ack
DCN disconnect
Phase A Call Establishment
Phase B Control and Capabilities Exchange
Interworking Functions and Hybrid IP + PSTN Network
Phases C and D Page Transfer
Phase E Call Release
CNG
CED answer tone
DIS + [NSF+CSI]
DCS + [TSI+NSS+SUB]
[CFR + CTR] | FTT
MPS or EOP
MCF image ack
DCN disconnect

Figure 3.1 Conventional Group 3, T.30 Fax Transmission Call Flow [5].

After dialing the remote fax number, an auto-calling fax terminal plays a CNG tone, which is an 1100 Hz tone ON for 0.5 secs and OFF for 3 secs. The called terminal answers the phone and plays the CED (Called Station ID) tone, which is a 2100 Hz tone with phase reversals, for 4 secs. The called station also sends a DIS (Digital Identification Signal) carrying the station capabilities and optionally two additional signals, NSF (Non Standard Facilities) and CSI (Called Subscriber Identification). NSF is used to identify requirements for the stations that are not explicitly covered by the ITU-T T series of recommendations. CSI provides the identity of the called subscriber in the form of a telephone number. This signaling exchange completes the call establishment—Phase A—of the process.

Attributes, capabilities, and control signaling exchange is performed using the 300 bps modulation mode of V.21. Messages are preceded by a preamble consisting of one second of HDLC flags to condition the line for each turnaround. Phase B begins with the calling terminal sending DCS (Digital Command Signal) to send configuration data, complete the digital setup, and respond to the DIS command. The TCF (Training Check Field) is sent by the sender, and the called station responds with CFR (Confirmation to Receive). At that time page transmission is ready to begin. The sender sends training flags to turn the line around and begins transmission, Phase C, in accordance to ITU-T Recommendation T.4.

At the end of each page the sender sends post-image handshakes (Phase D), which are either EOP (if that was the last page), EOM (End of Message), or MPS (MultiPage Signal) if there are multiple pages to send. MPS results in re-entering Phase C. If the sending session wishes to return to Phase B at the end of a page, it sends an EOM command. An MPS signal must be acknowledged by the receiver. This process continues until the last page has been transmitted, at which time the sending station disconnects by sending a DCN command to the receiver. This terminates the fax call.

3.2.2 Fax over Packet Networks

The support of fax over packet networks has major business drivers. Fax is a major revenue earner for service providers and a major expenditure for corporations. More than half of the fax transmissions are long distance calls and therefore the desire to reduce costs is great. The number of installed fax machines continues to grow. The problem with the fax scenario has been described in the previous section is that it is a POTS telephone call incurring toll charges, just like any voice call. The desire of the business sector to reduce the cost of fax has led to advances in packet technology to support facsimile transport.

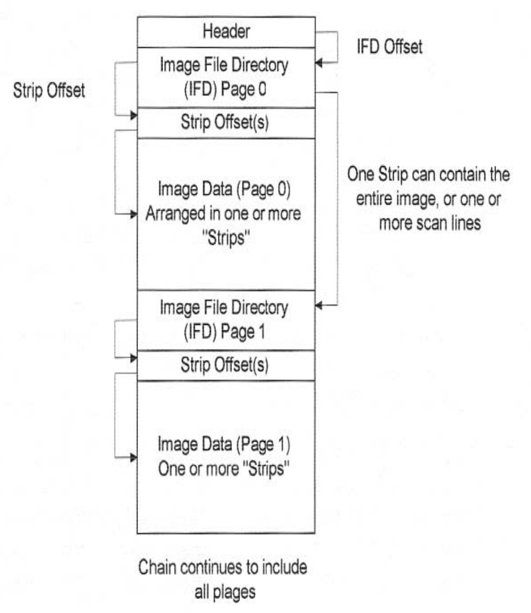
It has been briefly mentioned the two ITU-T specifications for fax over packet networks, and it has been visited in this section. Recommendation T.37 defines procedures for store-and-forward fax transmission over the Internet. The functionality provided through this specification is simple facsimile transmission with non-real-time requirements. Simply stated, the fax begins and ends with a local device emulating Group 3 facsimile operation, and the actual transmission of the image to the intended final destination fax machine occurs in a second step, a short time later. Recommendation T.38 on the other hand, defines real-time procedures for fax support over IP networks, and this is the standard that has been accepted for enterprise and wide area networks. Both standards address legitimate commercial needs with slightly different business drivers. The major attributes have been examined, and some major requirements they impose on the underlying packet networks.

3.2.3 Store-and-Forward Fax over IP Networks-T.37

Recommendation T.37 defines two modes for non-real-time fax, simple and full. Simple mode supports plain transmission of data, but capabilities negotiation between terminals may not take place and is undefined in the specification. All fax terminals must support simple mode. Image data is sent in TIFF format, specified in RFC 2301, Profile S with Modified Huffman Compression. It supports Group 3 standard and fine image resolutions. The fundamental element for T.37 fax operation is the Internet Fax Gateway, which emulates Group 3 operation toward the attached stations, and has a direct connection to a packet network, acting as a host or router. In Figure 3.2, the Internet Fax Gateway provides the protocol mapping between the standard Group 3 fax terminal on one side and the IP network on the other.

On the network side, the gateway interfaces to the IP network conforms to RFC 2305 for errors in handling and delivery of the fax, information to trace the origin of the fax, ensuring MIME compliance at both ends of the IP fax gateways, sending notification to the originator of the fax regarding reception problems, and optionally using TIFF profiles for other fax types. The gateway must also implement the Simple Mail Transfer Protocol (SMTP). The Internet Fax Gateway functionality can be implemented inside the fax terminal, thus making it Internet-aware.

The TIFF specification defines the method for describing, storing, and interchanging image data, such as facsimile and scanned documents. It defines a core set of fields, shown in Figure 3.3, along with the method to arrange the image data in a file which includes all the document pages in chained fashion. The exact header field definition for the image encoding is specified in RFC 2301, "File Format for Internet Fax".


Header
IFD Offset
Strip Offset
Image File Directory (IFD) Page 0
Strip Offset(s)
Image Data (Page 0) Arranged in one or more "Strips"
One Strip can contain the entire image, or one or more scan lines
Image File Directory (IFD) Page 1
Strip Offset(s)
Image Data (Page 1) One or more "Strips"
Chain continues to include all plages

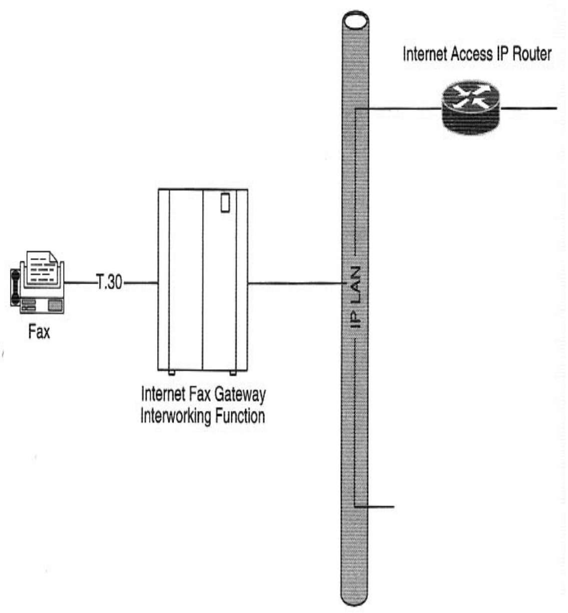

Internet Access IP Router
T.30
Fax
IP LAN
Internet Fax Gateway Interworking Function

Figure 3.2 Internet Fax Gateway, working Function [5] **Figure 3.3 Basic TIFF image Inter format [5]**

Images in both the T.37 simple and full operating modes are sent to the remote gateway as MIME-encoded email messages containing the image as the attachment. T.37 full mode adds the requirement to confirm that the fax was received properly, and negotiation of the capabilities of the fax terminals. Any Internet mail transport protocol can be used in full mode to carry the image data.

Delivery confirmations are returned to the sender as MIME-encoded Delivery Status Notifications (DSNs) for gateways, as described in RFC 1894. Senders and receivers require the use of Message Disposition Notifications (MDN), defined in RFC 2298.

3.2.4 Real-time Internet Fax-T.38

Store-and-forward fax is a rather primitive approach to support non-realtime facsimile, with some fairly obvious limitations in functionality. Even so, the business side of the argument points to substantial interest for this type of service and it is experiencing continuing growth in the industry. Several service providers already offer Internet fax, sometimes in package deals with other services.

The alternative to non-real-time fax over IP networks is Recommendation T.38. The T.38 protocol gives the "look and feel" of real-time facsimile by emulating the handshake activities of the T.30 protocol on the packet network side. Its basic idea is fax demodulation by a T.38 gateway at the source, packetization of all relevant handshake exchanges, sending of the IP packets across the network, and remodulation of the analog line by the receiving T.38 gateway from the information carried in the packet data. All this is accomplished with the simplicity of just two types of messages (packets), T30_INDICATOR (indicator packets) and T_DATA (data packets), which are part of the Internet Fax Protocol (IFP) recommendation of the ITU.

Indicator packets carry information to the far end about the presence of a CNG/CED tone, modem modulation training, or preamble flags each time the line is turned around. Data packets carry the Phase C data and HDLC control frames. Packets may carry one or more HDLC control frames, or a complete image. Adherence to the timing restrictions of T.30 is thus critical during the handshaking procedures between the terminals.

The IFP allows either TCP or UDP to be used as the transport protocol. When TCP is used, the IP payload is simply the TCP header and the concatenated indicator or data packet. It has been figure out in figure 3.4.

When UDP is used, the payload consists of a new layer header (UDPTL), followed by the concatenated indicator or data packet. The UDPTL header is a packet sequence number to account for packets arriving via different paths and out of order. The UDPTL payload also contains an optional Forward Error

Correction (FEC) field to recover from bit errors. Also optionally, redundant messages can be included in a single UDPTL packet. It has been figure out in figure 3.5.

A simplified block diagram of the message flow under T.38 is shown in Figure 3.6. For exact details of the T.38 message types and exchanges, ITU-T Recommendation T.38 should be studied.

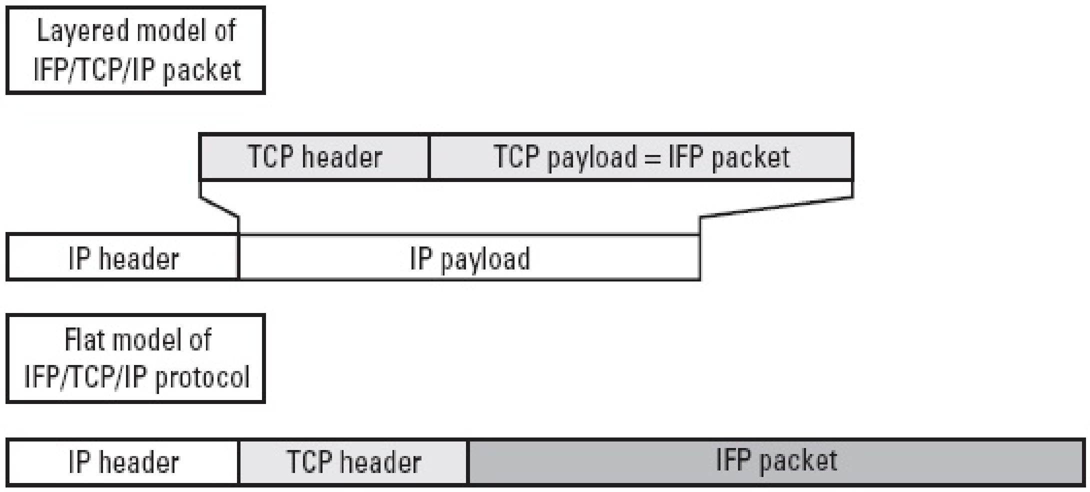


Figure 3.4 High-Level IFP / TCP Packet Structure [7].

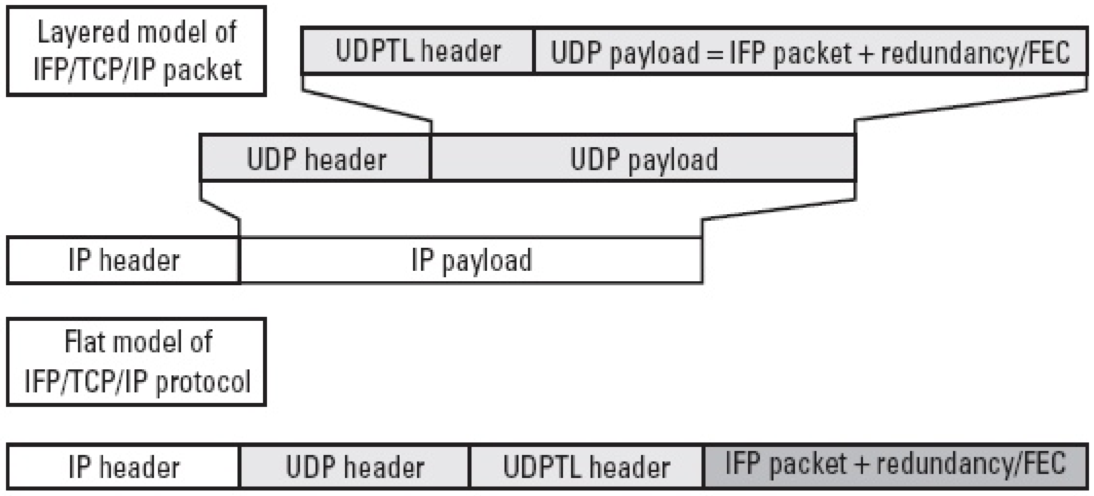

Layered model of IFP/TCP/IP packet
UDPTL header
UDP payload = IFP packet + redundancy/FEC
UDP header
UDP payload
IP header
IP payload
Flat model of IFP/TCP/IP protocol
IP header
UDP header
UDPTL header
IFP packet + redundancy/FEC

Figure 3.5 High-Level UDPTL / IP Packet Structure [7].

Flag sequences are required for every line turnaround and are transmitted as indicator (T30_INDICATOR) packets. Training is sent as an indicator packet, with the V-type modulation used by the sending terminal. This is used to adjust the speed of the terminal, for instance, to switch from sending image data with V.17 modulation to V.21 modulation for control sequences.

The same type of training is generated by the receiving gateway at the far end toward the sending fax terminal. The modulation training sequences have timing requirements which must be carefully adhered to in an end-to-end communication, in order for the presence of the IP network between the gateways to be completely transparent to the fax application.

Finally, the Training Check Field (TCF) can be used in one of two ways in T.38-compliant networks. For connection-oriented, TCP-based implementations, the TCF is generated by the far end gateway toward the receiving fax terminal. When UDP is used, the TCF needs to be sent across the packet network. The difference is in the decision logic of the speed selection.

The call flow of Figure 3.6 shows the T.30 protocol being executed between the calling terminal and its local T.38 gateway. All signal types and their timing restrictions must be supported at that interface, regardless of what timing constraints may be challenging the gateway on the packet network side.[5][9]

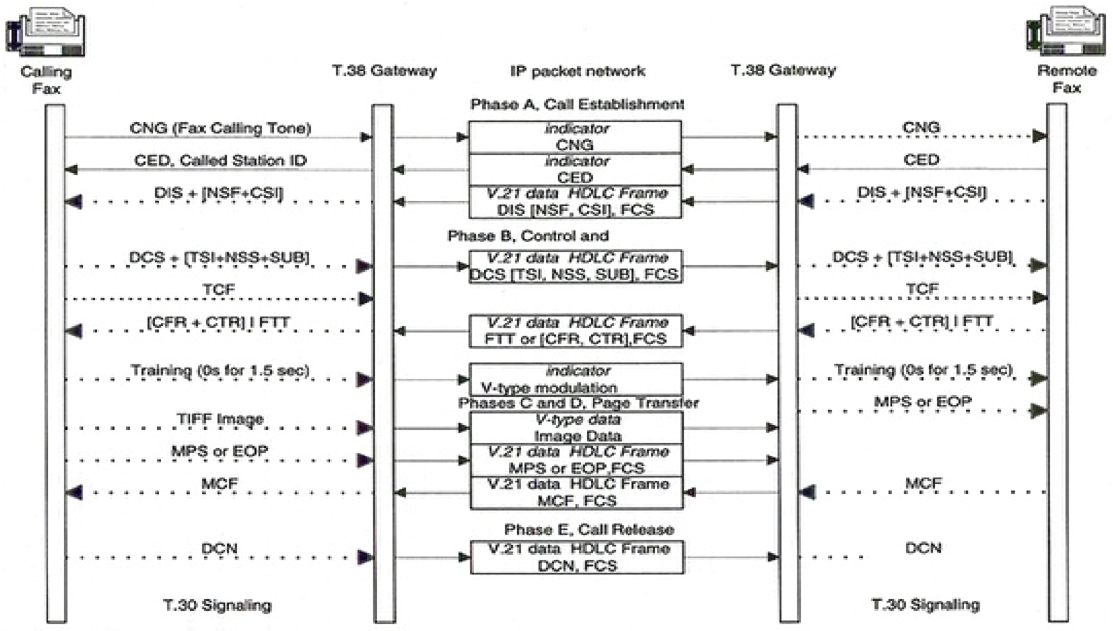


Figure 3.6 T.38 High Level Message Flows [5]

3.5 FAX OVER IP SURVEY

Most companies are unaware of just how much time and money is lost by traditional faxing. The average Fortune 500 company spends $40 million per year on phone service, 40 percent of which goes to faxing, according to a Gallop/Pitney Bowes survey [8]. By switching to emerging fax over IP, companies can save as much as 70 percent on their long distance phone bill, while gaining some important new features, says Maury Kauffman, managing partner of The Kauffman Group, a fax technology consulting firm in Vorhees, NJ. And that's not all. When calculating the full benefits of fax over IP, companies must factor in the cost of fax machines (which can run as high as $2,000 to $3,000 per machine); the cost of operating and maintaining those machines; and wasted labor each time an employee walks to the fax machine, waits for the fax to go through and returns to work. For larger companies using fax servers, the cost can be enormous. Companies can eliminate all of this by switching to fax over IP.

Fax over IP can also help companies cut down on the cost of other delivery methods, such as mail, overnight delivery and courier services. Fax over IP also solves the problem of mobile professionals who cannot receive faxes when they are out of the office. Users simply create a document in a program like Microsoft Word, click on file/print and then choose the installed fax-configured printer. After entering the appropriate fax number, users can send the document right from their desktops. The technologies allow users to send and receive fax documents via desktop computers.

The fax over IP market shows no signs of slowing as more and more companies turn to the Net to transmit documents. According to research by The Gartner Group, fax over IP reached 5.6 billion pages carried in 2001, up from 44 million pages in 2001. IDC estimates that fax transmissions represented an $83 billion dollar market in 2002 and grew to $90 billion in 2006. There is a wide range of numbers describing the current size of the IP telephony market and the growth of the market over the next three to five years. While the specific projections vary, even the most conservative analysts are predicting phenomenal growth. The numbers are summarized below.

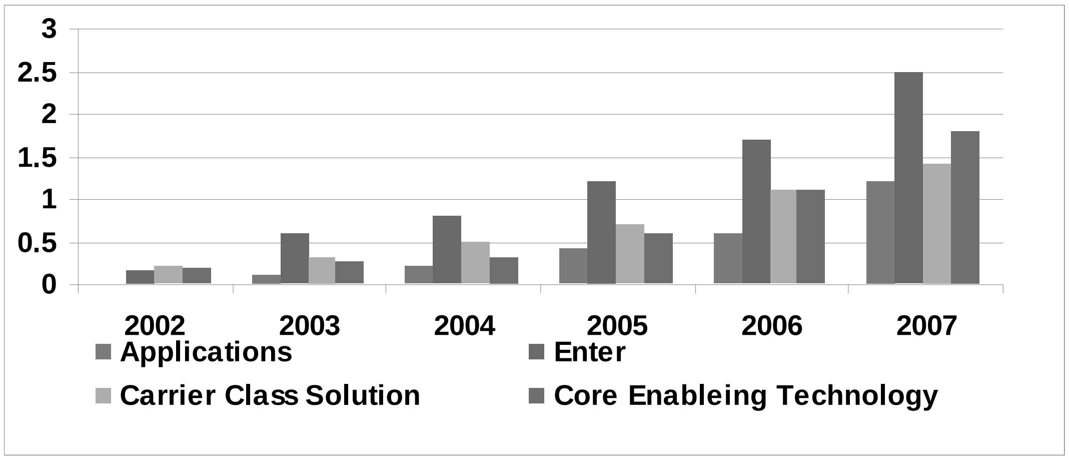


Figure 3.7 Piper-Jaffray, IP Telephony, Driving the Open Communications Revolution [8].

Chapter 4

SIP: Session Initiation Protocol

4.1 INTRODUCTION

In this chapter, the Session Initiation Protocol (SIP) is treated thoroughly. In the next section, an introduction of the protocol is given. In Section 3, SIP user agents, gateways and the 3 types of servers are discussed. A brief introduction of SIP request and response messages and headers is given in Section 4. Session Description Protocol (SDP) is treated next. SIP and T.38 interactions are explained briefly in Section 6. And the chapter has been concluded with introduction of the materials for further studying.

4.2 INTRODUCING SIP

4.2.1 A Brief History of SIP

SIP was originally developed by the IETF Multi-Party Multimedia Session Control Working Group, known as MMUSIC. Version 1.0 was submitted as an Internet-Draft in 1997. Significant changes were made to the protocol and resulted in a second version, version 2.0, which was submitted as an Internet-Draft in 1998. The protocol achieved Proposed Standard status in March 1999 and was published as RFC 2543 in April 1999. In September 1999, the SIP working group was established by the IETF to meet the growing interest in the protocol. An Internet-Draft containing bug fixes and clarifications to SIP was submitted in July 2000, referred to as RFC 2543 "bis". In June 2002, RFC 3261 was published which defined the latest SIP specification and made RFC 2543 obsolete. To advance from Proposed Standard to Draft Standard, a protocol must have multiple independent interworking implementations and limited operational experience. To this end, forums of interoperability tests have been organized by the SIP working group. The final level, Standard, is achieved after operational success has been demonstrated.

SIP incorporates elements of two widely used Internet protocols: HTTP (Hyper Text Transport Protocol) used for web browsing and SMTP (Simple Mail

Transport Protocol) used for e-mail. From HTTP, SIP borrowed a client-server design and the use of uniform resource locators (URLs). From SMTP, SIP borrowed a text-encoding scheme and header style. For example, SIP reuses SMTP headers such as To, From, Date and Subject.

In keeping with its philosophy of "one problem, one protocol", the IETF designed SIP to be a pure signaling protocol. SIP uses other IETF protocols for transport, media transport, and media description.

4.3 SIP CLIENTS AND SERVERS

4.3.1 SIP User Agents

An SIP-enabled end-device is called an SIP user agent (UA). The main purpose of SIP is to enable sessions to be established between user agents. As the name implies, a user agent takes direction or input from a user and acts as an agent on their behalf to set up and tear down media sessions with other user agents. In most cases, the user will be a human, but the user could be another protocol, as in the case of a gateway described in the next section. A user agent must be capable of establishing a media session with another user agent. Since SIP may be used with any transport protocol, there is no requirement that a UA must support either TCP or UDP for message transport. The standard states, however, that a UA should support both TCP and UDP [7].

4.3.2 SIP Servers

SIP servers are applications that accept SIP requests and respond to them. An SIP server should not be confused with a user agent server or the client-server nature of the protocol, which describe operation in terms of clients (originators of requests) and servers (originators of responses to requests). An SIP server is a different type of entity. The types of SIP servers discussed in this section are logical entities. Actual SIP server implementations may contain a number of server types, or may operate as a different type of server under different conditions. Because servers provide services and features to user agents, they must support both TCP and UDP for transport. Figure 5.3 shows the interaction of user agents, servers, and a location service. Note that the protocol used between a server and the location service or database is not in general SIP and is not discussed here.

An SIP proxy server that receives an SIP request from a user agent acts on behalf of the user agent in forwarding or responding to the request. A proxy server typically has access to a database or a location service to aid it in processing the request (determining the next hop). The interface between the proxy and the location service is not defined by the SIP protocol. A proxy can use any number of types of databases to aid in processing a request.

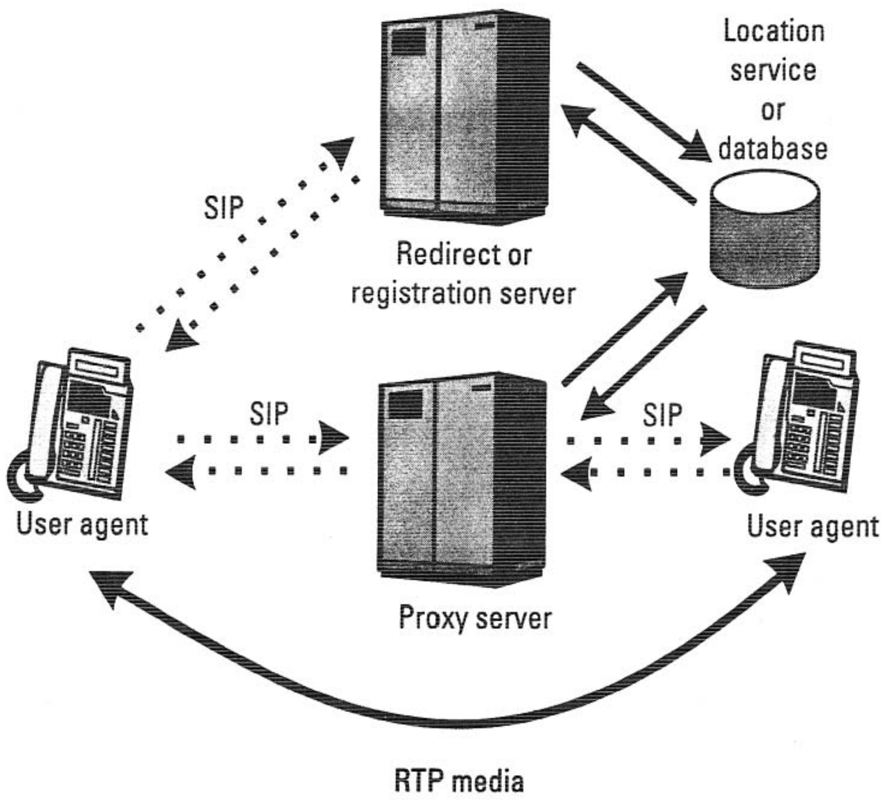


Figure 4.1 SIP user agent, server, and location service interaction [9].

Databases could contain SIP registrations, or any other type of information about where a user is located.

A proxy server is different from a user agent or gateway in two key ways:

1. A proxy server does not issue a request; it only responds to requests from a user agent. (A CANCEL request is the only exception to this rule.)

2. A proxy server has no media capabilities.

A proxy server can be either stateless or stateful. A stateless proxy server processes each SIP request or response based solely on the message contents. Once the message has been parsed, processed, and forwarded or responded to, no information about the message is stored—no call leg information is stored. A stateless proxy never retransmits a message, and does not use any SIP timers. A stateless proxy has no memory of any requests or responses it has sent or received. A stateless proxy is still capable of detecting message looping since SIP uses a stateless method to implement loop detection using via headers.

A stateful proxy server keeps track of requests and responses received in the past and use that information in processing future requests and responses. For example, a stateful proxy server starts a timer when a request is forwarded. If no response to the request is received within the timer period, the proxy will retransmit the request, relieving the user agent of this task. Also, a stateful proxy can require user agent authentication.

Redirect Servers

A redirect server is introduced as a type of SIP server that responds to, but does not forward requests. Like a proxy server, a redirect server uses a database or location service to look up a user. The location information, however, is sent back to the caller in a redirection class response, which concludes the transaction.

Registration Servers

A registration server accepts SIP REGISTER requests; all other requests receive a 501 Not Implemented response. The contact information from the request is then made available to other SIP servers within the same administrative domain, such as proxies and redirect servers. In a registration request, the To header contains the name of the resource being registered, and the Contact headers contain the alternative addresses or aliases.

Registration servers usually require the registering user agent to be authenticated so that incoming calls can not be hijacked by an unauthorized user. This could be accomplished by an unauthorized user registering someone else's SIP URL to point to their own phone. Incoming calls to that URL would then ring the wrong phone. Depending on the headers present, a REGISTER request can be used by a user agent to retrieve a list of current registrations, clear all registrations, or add a registration URL to the list.

4.4 SIP REQUEST AND RESPONSE MESSAGES AND HEADERS

4.4.1 SIP Request Messages

This section explains the types of SIP requests called methods. Six are described in the SIP specification document. Two more methods are work items of the SIP working group. Other proposed methods are still in the early stages of development, or have not yet achieved working group consensus.

SIP requests or methods are considered "verbs" in the protocol, since they request a specific action to be taken by another user agent or proxy server.

The INVITE, REGISTER, BYE, ACK, CANCEL, and OPTIONS methods are the original six methods in version 2.0 of SIP. The INFO and PRACK methods are the subsequent additions.

A proxy does not need to understand a request method in order to forward the request. A proxy treats an unknown method as if it were an OPTIONS; that is, it forwards the request to the destination if it can. This allows new features and methods useful for user agents to be introduced without requiring support from proxies that may be in the middle. A user agent receiving a method it does not support replies with a 501 Not Implemented response.

4.4.2 SIP Response Messages

This section covers the types of SIP response messages. An SIP response is a message generated by a UAS or an SIP server to reply to a request generated by a UAC. A response may contain additional headers containing information needed by the UAC. Or, it may be a simple acknowledgement to prevent retransmissions of the request by the UAC. Many responses direct the UAC to take specific additional steps. There are six classes of SIP responses. The first five classes were borrowed from HTTP; the sixth was created for SIP.

4.4.3 SIP Headers

This section describes the headers present in SIP messages. There are four types of SIP headers: general, request, response, and entity. SIP headers in most cases follow the same rules as HTTP headers. Headers are defined as header: field where header is the case-insensitive token used to represent the header, and field is the case-insensitive set of tokens that contain the information. Header fields can continue over multiple lines as long as the line begins with at least one space or horizontal tab character. Unrecognized headers are ignored by proxies. Many common SIP headers have a compact form, where the header name is denoted by a single lower-case character. Headers can be either end-to-end or hop-by-hop. Hop-by-hop headers are the only ones that a proxy may insert, or with a few exceptions, modify. A proxy should never change the header order. Because SIP typically involves end-to-end control, most headers are end-to-end.

General headers: The set of general headers includes all of the required headers in an SIP message. General headers can be present in both requests and responses. These headers are created by user agents and cannot be modified by proxies, with a few exceptions. The general headers are: Call-ID, Contact, CSeq, Date, Encryption, From, Organization, Retry-After, Subject, Supported, Timestamp, To, User Agent, Via.

Request headers: They are added to a request by a UAC to modify or give additional information about the request. The request headers are: Accept, Accept-Contact, Accept-Encoding, Accept-Language, Authorization, Hide, In-Reply-To, Max-Forwards, Priority, Proxy-Authorization, Proxy-Require, Record-Route, Reject-Contact, Request-Disposition, Require, Response-Key, Route, RAck, Session-Expires.

Response headers: They are added to a response by a UAS or SIP server to give more information than just the response code and reason phrase. They are generally not added to a request. The response headers are: Proxy-Authenticate, Server, Unsupported, Warning, WWW-Authenticate, RSeq.

Entity headers: They are used to provide additional information about the message body or resource requested. This term comes from HTTP where it has a more specific meaning. In SIP, “entity” and “message body” are used interchangeably. The entity headers are: Allow, Content-Encoding, Content-Disposition, Content-Length, Content-Type, Expires, MIME-Version.

4.5 SDP: A COMPANION PROTOCOL

The Session Description Protocol, defined by RFC 2327, was developed by the IETF MMUSIC working group. It is more like description syntax than a protocol in that it does not provide a full-range media negotiation capability. The original purpose of SDP was to describe multicast sessions set up over the Internet's multicast backbone, the MBONE. The first application of SDP was by the experimental Session Announcement Protocol (SAP) used to post and retrieve announcements of MBONE sessions.

SDP contains the following information about the media session:

I. IP Address (IPv4 address or host name);

II. Port number (used by UDP or TCP for transport);

III. Media type (audio, video, fax, interactive whiteboard, etc.);

IV. Media encoding scheme (PCM A-Law, MPEG II video, etc.).

In addition, SDP contains information about the following:

I. Subject of the session;

II. Start and stop times;

III. Contact information about the session.

Like SIP, SDP uses text coding. An SDP message is composed of a series of lines, called fields, whose names are abbreviated by a single lowercase letter, and are in a required order to simplify parsing.

SDP was not designed to be easily extensible, and parsing rules are strict. The only way to extend or add new capabilities to SDP is to define a new attribute type. However, unknown attribute types can be silently ignored. An SDP parser must not ignore an unknown field, a missing mandatory field, or an out-of-sequence line.

4.6 SIP AND T.38 UTILIZATION FOR FOIP

The best current practices for SIP T.38 fax and SIP fax pass-through sessions are documented in this IETF Internet-Draft: “SIP Support for Real-time Fax: Call Flow Example And Best Current Practices” [20]. Here a brief overview of this document has been provided.

The Session Initiation Protocol (SIP) allows the establishment of real-time Internet fax communications. Real-time facsimile communications over IP may follow 2 modes of operation: T.38 fax relay as defined by the ITU-T T.38 recommendation or fax pass through.

This document clarifies the options available to Internet telephony gateway vendors to handle real-time fax calls using SIP. While the primary focus is to address the more reliable real-time T.38 Group 3 fax mode, fax pass-through mode to enable fallback operations and super G3 fax communications using SIP are also briefly covered. Examples of SIP call flows for real-time Internet fax gateways or SIP proxy redirect servers are given as well. Elements in these call flows include SIP User Agents, SIP Proxy Servers, and Gateways to the PSTN (Public Switch Telephone Network).

A session starts with audio capabilities, and, upon fax tone detection, T.38 fax capabilities are negotiated; upon successful negotiation, the session continues with fax capabilities and the media termination hosts exchange T.38 Internet fax packets. The T.38 fax call scenarios include various aspects of the call sequence: the detection of fax transmission, the usage of the T.38 session description attributes the optional fallback into fax pass-through mode and the session termination. The fax pass-through call scenarios involve some specific SDP media attributes to enable proper fax transmission. Fax transmission can be detected by the receiving side, the emitting side or both.

For T.38, this document deals primarily with one transport protocol for the media: T.38 over UDP/UDPTL; T.38 fax packet transport over TCP using SIP session establishment can easily be considered as well. These T.38 call flows were developed in the design of carrier-class SIP Telephony products supporting voice and real-time fax traffic.

The Internet telephony gateway only supports T.38 real-time fax communications. In this case, the Internet fax gateway should initiate the SIP session with T.38 SDP capabilities.

Chapter 5

Simulator Implementation Details

5.1 INTRODUCTION

In this chapter, J-Sim, an open-source simulator, is introduced. This simulator is utilized throughout the implementation part of this thesis to explore the behavior of the developed protocol and components. Many details have been left out to simplify the introduction of the simulator. A brief introduction is given and the salient features of the simulator are discussed in the Section 3. A quick overview of the inner workings of the simulator and how one can develop new modules are given in Section 4. Simulation scenario creation, configuration and running are briefly introduced in the last section.

5.2 INTRODUCING J-SIM

The implication of the implementation is three fold:

1. With all the Internet protocol classes available, one can compose the protocol stack and conduct the simulation under different network scenarios in a plug-and-play fashion.
2. With the abstract classes that capture the fundamental features of network entities and yet are flexible enough to accommodate new technology advances, one can extend J-Sim to a new network architecture, e.g., wireless LANs, optical networks with WDM technology, networks with satellite communication links, or ad hoc networks consisting primarily of mobile sensors. This is done by subclassing appropriate network modules and redefining their network attributes and methods that manipulate the attributes. For example, one needs only to modify the network interface card (NIC) component and the link component to incorporate the error characteristics and the mobility Characteristics in order to model wireless mobile networks. Similarly, one can readily implement a new algorithm/protocol for experimentation and validation, simply by sub classing one or more appropriate protocol modules.
3. By virtue of the component hierarchy (i.e., a component can be a

composite component that contains child components), one can vary the level of details to which simulation is conducted.

Table 5.1 Algorithms and protocols supported in J-Sim.

Network Architecture	Application	Socket Layer	Transport	Routing	Traffic Model	Tagger Marker	Buffer Management	NI Scheduling
Best Effect Services	FTP, FSP, WWW	BSD 4.3	TCP Reno TCP Tahoe TCP Vegas TCP Sack UDP	RIP (DV) OSPFv2 Multicast shortest path tree, Multicast minimum load tree , Multicast Steiner tree, DVMRP MOSPF CBT			Drop-Tail	FIFO
Differentiated Services						Token Bucket TSW ETSW	RED FRED SRED BRED	FIFO
Integrated Services			RSVP	Unicast QoS routing QoS-enhanced OSPFv2 QoS-enhanced CBT	Periodic message (CBR) Peak rate model Leaky bucket model Token bucket model IETF/Intserv Flowspec (r,t)-smooth model (C,D)-smooth model			RM EDF Stop-and-go DCTS VirtualClock LFVC SCFQ PGPS STFQ WF2Q Leave-in-time

BSD: Berkeley socket distribution	FSP: file service protocol
RIP: routing information protocol	TCP: transmission control protocol
DVMRP: distance vector multicast routing protocol	OSPF: open shortest path first
CBT: core based tree protocol	MOSPF: multicast extension to OSPF
TSW: time sliding window	FIFO: first in first out
RED: random early drop	ETSW: enhanced time sliding window
SRED: stable random early drop	FRED: fair random early drop
RSVP: Resource reservation protocol	BRED: balanced random early drop
RM: rate monotonic	CBR: constant bit rate
DCTS: distance constrained task system	EDF: earliest deadline first
SCFQ: self-clocked fair queuing	LFVC: leap forward virtual clock
STFQ: start time fair queuing	PGPS: packet-by-packet generalized processing shari
	WF2Q: worst-case fair weighted fair queuing

5.3 NETWORK SIMULATION FRAMEWORK AND SIMULATION SCENARIO CREATION

INET is a network simulation framework built upon the autonomous component architecture and specific to network simulation. Essentially features common to each network component (such as an IP layer, a network interface card, a link, etc.) have been factored out and all the network components (and their contracts) are defined and implemented in INET. Internal structure of a node (either an end host or a router) is also defined. Users may then compose a network scenario in a plug-and-play fashion, by connecting components in their desired manner. Users may also subclass an appropriate component and redefine new attributes and methods to incorporate their own protocols/algorithms. To create a network simulation scenario, following items should be taken into consideration:

-Topology creation
-Building the internal structure of nodes
-Configuring the network scenario and miscellaneous issues

A network is a composite component which consists of nodes, links and smaller networks. A node is also a composite component which consists of applications, protocol modules, and a core service layer (CSL).

The core service layer is an abstract component which encapsulates the functions of the network layer and the layers beneath the network layer. It provides network services and events to protocols, in the form of inter-component contracts.

In particular, since configuring the internal structure of nodes usually follows similar patterns and repetitively cycles through several tedious steps, some utility classes have been provided to automate the process. The basic way to compose a scenario is to build it "by hand". That is, every necessary component, from networks, nodes, links to protocols and modules inside a node is created and they are connected together afterwards. The idea is very simple but the tasks are repetitive and can get a bit tedious even for a small-sized network. Fortunately, both tasks of creating a network topology and building network nodes can be made follow certain patterns and then automate the processes. The utility class drcl.inet.InetUtil and the builder classes drcl.inet.NodeBuilder and drcl.inet.CSLBuilder are developed for this purpose. Building a network simulation scenario in J-Sim is outlined in the following TCL script:

```
# Create a container to hold the scenario cd
[mkdir drcl.comp.Component scene]
    # Step 1: Create topology
```

```
...
# Step 2: Build nodes
...
# Step 3: Configure nodes
...
# Attach simulator runtime to "scene" attach_simulator.
# Start all "active" components under "scene" if there is anyrun.
```

In what follows, the process of creating a scenario with the utility functions in drcl.inet.InetUtil are introduced. Followed by that, builder classes and their usages are introduced. Finally, few other utility functions in drcl.inet.InetUtil are introduced (The process of building/configuring scenario by hand is not discussed here).

5.3.1 Create Topologies

There is a set of createTopology(...) methods in the drcl.inet.InetUtil class for automating the process of creating a topology . The simplest form of all is as follows:

```
public static void createTopology(Component network_,
                                   int[][] adjMatrix_, Link link_);
```

The network_ argument is where the nodes are to be created in. The most important argument in all the createTopology() methods is the adjacency matrix, adjMatrix_. It is a two-dimensional array. The length of the first dimension, i.e., adjMatrix_.length, is the number of nodes. Each element in the first dimension is a one-dimensional array, which represents the neighbors of the node. The position of a neighbor in this one-dimensional array is the ID of the port that the node uses to connect to the neighbor. Nodes are indexed as 0, 1,... (adjMatrix_.length-1). The neighbors are represented by their indices. Each node may have a different number of neighbors. link_ is the physical link component used to connect two nodes. The most complete form of all the createTopolgy() methods is the following:

```
public static void createTopology(  Component network_,
                                      String routerIDPrefix_,
                                      String hostIDPrefix_,
                                      Object[] existing_,
                                      int[][] adjMatrix_,
                                      long[] ids_,
                                      Link link_,
                                      boolean assignAddress_);
```

5.3.2 Builders

After a network topology is created, the next task is to build the nodes. Note that the nodes created during the process of creating a network topology are just empty composite components. Appropriate protocols and modules need to be put in to make them functional. One way to do this, is of course building them by hand. In this section, using builder classes to automate the process is explained. The rationale behind this is very simple. Instead of building network nodes one by one by hand, nodes are first categorized. Supposedly there should be far less types of nodes in the network than the number of nodes themselves. Then a node template for each type of node is built by hand, and then the nodes of the same type are built by duplicating the structure of the template node.

5.3.3 Configuring the Network Scenario and Miscellaneous Issues

I) Static Routes Setup

Instead of using the node properties to manually set up static route entries along a path, drcl.inet.InetUtil includes a set of setupRoutes(...) methods to automate the task. Given source and destination nodes, the methods compute the unicast or multicast routes with minimum hop count, and then install appropriate route entries in the nodes along the routes. The following is one of the forms of the method:

public static void **setupRoutes**(Node src_, Node dest_, String bidirect_);

II) Online Interactions

In addition to creating scenarios, running simulations often involve a lot of online activities such as debugging, tuning parameters and collecting results. A set of RUV system commands and utility components are developed to facilitate these tasks:

III) Save Results Directly to a File - drcl.comp.io.FileComponent

The drcl.comp.io.FileComponent component saves incoming data to a file. To use it, connect a FileComponent to the port at which the target component originates interested results.

IV) xy Plot - drcl.comp.tool.Plotter

The drcl.comp.tool.Plotter component displays incoming data on an xy plot. The Plotter component is able to display multiple datasets on a plot as well as display multiple plots at the same time. Multiple plots are ordered by IDs starting from 0, so the datasets on a plot. The port ID and the port group ID of the port at which data arrives are used as the dataset ID and the plot ID respectively to draw the data on its corresponding plot. In addition to be integrated as part of the component system, Plotter can be used as a standalone Java Program with the following usage:

```
java drcl.comp.tool.Plotter ?-1? <file1> ?<file2>...?
```

V) NAM Trace - drcl.inet.tool.NamTrace

The NamTrace component is an instrument class that probes appropriate components to collect interested events and produce trace outputs in the NAM (VINT/UCB Network Animator) trace format. Currently, NamTrace supports the following NAM events/configurations: node, link, queue, color and packet. The first four events are usually used in the initial/configuration part of a trace that defines the network topology and the color index. Packet events are collected from probing appropriate components in the system. In all cases, the traces are produced at the output port of the NamTrace component. To save the output in a file, one must connect a file component, drcl.comp.io.FileComponent, to the output port of the NamTrace component. With the following utility method, all the necessary configurations can be done in one line no matter how many nodes and links exist:

```
set nam [java::call drcl.inet.InetUtil setNamTraceOn [! .] \
"SimNAMTrace.nam" [_to_string_array "red blue yellow green black orange"]]
```

Here I wrap up the introduction of the relevant simulator capabilities and move on to study the developed modules and the simulation scenario in the next chapter.

Chapter 6

Developed Modules, Simulation Scenario & Corresponding Results

6.1 INTRODUCTION

In this chapter, as the title of the chapter suggests, the thesis has been wrapped up with presenting the final elements, i.e. presenting the developed modules, studying the simulation scenario and presenting the accomplished results. In Section 3, the developed SIP protocol and components are presented. Specifically, some extracts of the outputs of the Javadoc software pro duced from sifting through the source codes are presented. These APIs can visualize the outline of how modules really operate. In Section 4, a typical simulation scenario is analyzed and different stages of scenario construction and running are explained. In the last section, results of the aforementioned scenario are presented and discussed.

6.2 THINGS THAT ARE IMPLEMENTED

First, I review what is supposed to come out of this simulation and then move on to the modules details in the next section. It has been intended to explore whether fax parameters details can be negotiated using SIP/SDP. Specific SDP attributes and the interaction of SIP and T.38 protocols are not of high importance in this thesis. The implemented parts are the session establishment, starting a typical file transfer, which can serve as a demonstration of T.38 fax transfer, and the subsequent session tear-down after file transfer is complete. Due to the fact that specific T.38 protocol SDP headers are not studied in this thesis, in the message content prints in the simulation results section, only few constant symbolic SDP header fields are present. As pointed out in the next chapter, studying of these fields is considered as possible future work. One other important analysis carried out in this simulation, is the utilization of SIP contact header for reducing the load on proxy servers which is a highly desirable feature.

6.3 DEVELOPED MODULES

In total, six modules have been developed and they are: SDPMessage which is a class implementing SDP headers; SIPMessage which is a class implementing SIP headers and also embeds an instance of SDPMessage in itself if the body type indicates so; SipPS which is a class implementing an SIP proxy server; SipUA which is a class implementing an SIP user agent and finally T38Receiver and T38Sender which subclass ftpd and ftp respectively and act on behalf of real T.38 modules.

The classes implementing SDP headers and T.38 receiver and sender are some simple classes, source codes of which are provided in the appendix. Some extracts of the APIs of classes implementing SIP message, user agent and proxy server are provided here and briefly explained. The full source codes of these modules can be found in the appendix as well.

6.3.1 SIP Message Class

This class provides a mechanism for storing the SIP headers. It utilizes the java.util.Properties class of Java for easily setting and retrieving the SIP headers and their corresponding values. It also embeds an instance of SDPMessage in itself if in its constructor the type of content is set to “application/sdp”. It also provides methods for retrieving both SIP and SDP headers as Properties objects to further manipulate them. Part of its API appears in Table 6.1.

Table 6.1 SIP Message Class API.

Field Summary	
java.util.Properties	**headers** SIP headers are held in this Properties object.
SDPMessage	**sdpMessage** An SDP message which is in the SIP message.
java.lang.String	**SIPContentType** If set to "application/sdp", an SDP message is created as an embedded object.

Constructor Summary

SIPMessage(java.lang.String contentType) Constructor.

Method Summary	
java.util.Properties	**returnSDPHeaders**() This is for someone who wants to set other SDP headers as well.
java.util.Properties	**returnSIPHeaders**() This is for someone who wants to set other SIP headers as well.

6.3.2 SIP Proxy Server

This class implements the SIP proxy server. It provides some initializing methods such as: setAddress(), setNodeViaField(), setRegisteredNode() and setOtherNetworkProxyServerAddress(). These method are called with appropriate arguments during the scenario building in the TCL script. It also defines methods for sending and processing these SIP requests: ACK, INVITE and BYE. It defines a response processor which prints informational messages based on the response class and it especially handles the OK response. The method dataArriveAtDownPort() handles the incoming data and directs it to the appropriate processor. Two utility methods, constructMessage() and printMessageContent(), are also provided and they carry out tasks described by their names. User agent registers itself with the proxy server during the initialization process through the TCL script. The transaction ID of the first received SIP message, its Call-ID header, is stored in the proxy server so the server can discard messages not belonging to this transaction. The rest of the details about the proxy server can be found in the following API or its full source code in the appendix.

Table 6.2 SIP Proxy Server API.

Field Summary	
java.lang.String	**contentType** A variable used for setting the content type of the SIP message, typically set to "application/sdp".
java.lang.String	**messageType** An intermediate variable which is used for checking whether a message is a request or it is a response and directing

	it to the related processor.
int	**nodeAddress** Node address is set during initialization through the TCL interface.
java.lang.String	**nodeViaField** It is set during initialization through the TCL interface.
int	**otherNetworkProxyServerAddress** It is set during initialization through the TCL interface.
SIPMessage	**receivedMessage** An intermediate variable which is used for processing.
int	**registeredNode** All requests of the set node first goes to this proxy server and it is set during initialization through the TCL interface.
int	**responseMessageClass** A vaiable to store the response class from one of six possible classes.
SIPMessage	**toBeSentMessage** This is a message which is created by different methods of the class.
java.lang.String	**transactionID** Used for storing the transaction ID so that junk messages can be discarded.

Constructor Summary
SipPS() Constructor.

Method Summary	
SIPMessage	**constructMessage** (java.lang.String startLine,java.lang.String via, java.lang.String to,java.lang.String from, java.lang.String callID,java.lang.String contentType)
protected void	**dataArriveAtDownPort** (java.lang.Object data, drcl.comp.Port downPort) Arrived data first gets processed by this method.
void	**duplicate**(java.lang.Object source)
java.lang.String	**info**()
void	**printMessageContent**(SIPMessage message)
void	**processACK**(SIPMessage receivedMessage)
void	**processBYE**(SIPMessage receivedMessage)
void	**processINVITE**(SIPMessage receivedMessage)
void	**processResponse**(SIPMessage receivedMessage) This method first checks to see whether the message is a valid one then checks to see if it's a response or an

	unsupported request, after that if the message is a response it goes on to handle each type of response classes.
void	**reset**()
void	**sendACK**(SIPMessage receivedMessage, int nextHop)
void	**sendBYE**(SIPMessage toBeSentMessage, int nextHop)
void	**sendINVITE**(SIPMessage toBeSentMessage, int nextHop)
void	**sendOK**(SIPMessage receivedMessage, int address)
void	**setAddress**(int address)
void	**setNodeViaField**(java.lang.String s)
void	**setOtherNetworkProxyServerAddress**(int address)
void	**setRegisteredNode**(int address)

6.3.3 SIP User Agent

This class implements the SIP user agent. Like proxy server, it provides some initializing methods: setAddress(), setConfiguredProxyServerAddress(), setNodeViaField() and setAlwaysUseProxyServer(). The last method sets the user agents to always use the proxy servers and never bypass them and contact each other directly. The default is false which means user agents, whenever they can, contact each other directly using the address found in the received SIP message Contact header. The same set of sending and processing requests and responses methods, found in proxy servers, are present here as well with some modifications. Some other methods which were in the proxy server class, as can be seen in the API, are also present here. There is also a field call faxPort which is used to alert the T.38 module to start sending the fax after the session establishment is complete. Again, the rest of the details about the user agent can be found in the following API or its full source code in the appendix.

Table 6.3 SIP User Agent API

Field Summary	
boolean	**alwaysUseProxyServer** It is set during initialization through the TCL interface.
int	**configuredProxyServerAddress** All requests of the node first goes to this address which is set during initialization through the TCL interface.
java.lang.String	**contentType** A variable used for setting the content type of the

	SIP message, typically set to "application/sdp".
int	**destination** Used for storing the other party's address.
drcl.comp.Port	**faxPort** Used for alerting the T.38 fax module to start sending the fax.
java.lang.String	**messageType** An intermediate variable which is used for checking whether a message is a request or it is a response and directing it to the related processor.
int	**nodeAddress** Node address is set during initialization through the TCL interface.
java.lang.String	**nodeViaField** It is set during initialization through the TCL interface.
SIPMessage	**receivedMessage** An intermediate variable which is used for processing.
int	**responseMessageClass** A vaiable to store the response class from one of six possible classes.
SIPMessage	**toBeSentMessage** This is a message which is created by different methods of the class.
java.lang.String	**transactionID** Used for storing the transaction ID so that junk messages can be discarded.
boolean	**transactionInitiator** Used for determining if the node should respond like acknowledging an OK with ACK only if the node is indeed the initiator of the request-response.

Constructor Summary
SipUA() Constructor.

Method Summary	
SIPMessage	**constructMessage** (java.lang.String startLine,java.lang.String via, java.lang.String to,java.lang.String from, java.lang.String callID,java.lang.String contentType)
protected void	**dataArriveAtDownPort** (java.lang.Object data,drcl.comp.Port downPort) Arrived data first gets processed by this method.
void	**Duplicate** (java.lang.Object source)
java.lang.String	**info**()
void	**printMessageContent**(SIPMessage message)

void	**processACK**(SIPMessage receivedMessage)
void	**processBYE**(SIPMessage receivedMessage)
void	**processINVITE**(SIPMessage receivedMessage)
void	**processResponse**(SIPMessage receivedMessage) This method first checks to see whether the message is a valid one then checks to see if it's a response or an unsupported request, after that if the message is a response it goes on to handle each type of response classes.
void	**reset**()
void	**sendACK**(SIPMessage receivedMessage)
void	**sendBYE**() **sendINVITE**(SIPMessage toBeSentMessage)
void	**sendOK**(SIPMessage receivedMessage)
void	**setAddress**(int address)
void	**setAlwaysUseProxyServer**(boolean x)
void	**setConfiguredProxyServerAddress**(int address)
void	**setNodeViaField**(java.lang.String s)

6.4 SIMULATION SCENARIO

As pointed out in the previous chapter, J-Sim simulator uses TCL scripts to carry out scenario building and configuration. In this section the TCL scripts used to do such tasks are analyzed line by line. In the first part, the script has been analyzed used for building the simulation scenario and in the next part, the script used for running the simulation is presented.

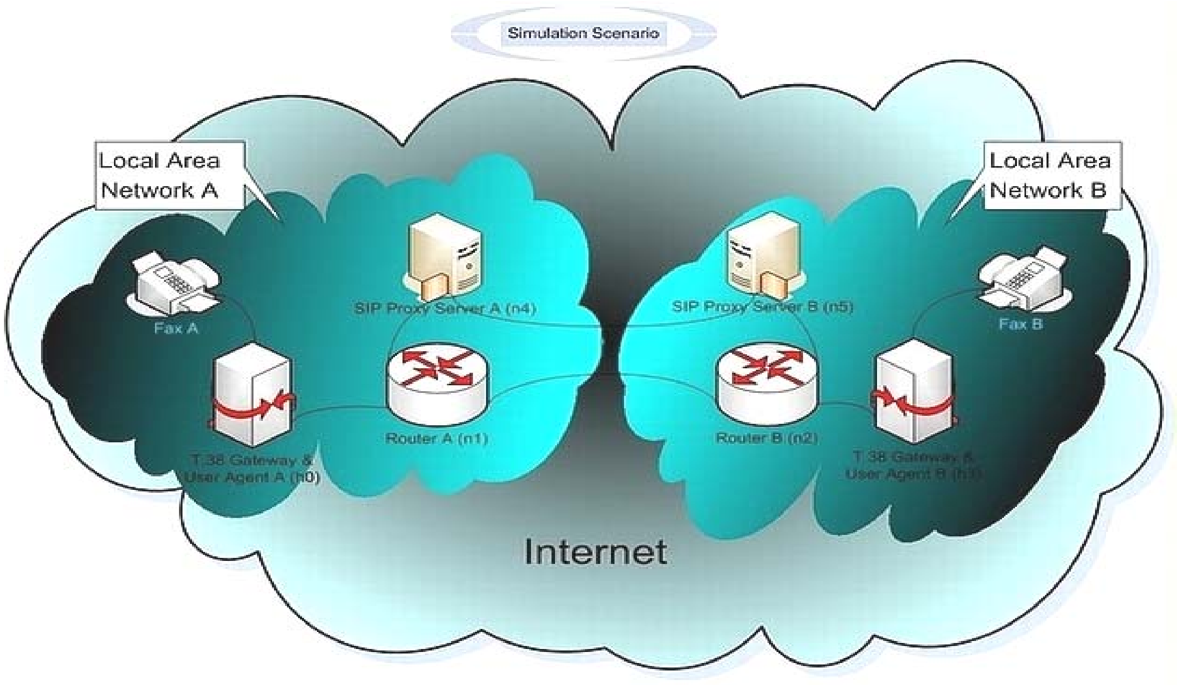


Figure 6.1 The Simulation Scenario.

6.4.1 The Network Topology

The simulated network scenario is depicted in Figure 6.1.

6.4.2 Scenario Building

```
#####################################################
# Topology:
#
# ------------------------
# / \
# h0-------n1-------n4-------n5-------n2-------h3
#
#####################################################
cd [mkdir drcl.comp.Component /SimScenario]

puts "Creating topology..."
set link [java::new drcl.inet.Link]
$link setPropDelay 0.3; # 300ms
set adjMatrix [java::new {int[][]} 6 {{1} {0 2 4} {5 1 3} {2} {1 5} {4 2}}]
java::call drcl.inet.InetUtil createTopology [! .] $adjMatrix $link

puts "Creating builders..."
# router builder:
set rb [mkdir drcl.inet.NodeBuilder .routerBuilder]
$rb setBandwidth 1.0e6; #1Mbps
# Host builder:
set hb [cp $rb .hostBuilder]
#Setting up transport protocols
set TCPModule [mkdir drcl.inet.transport.TCPb $hb/tcp]
set UDPModule [mkdir drcl.inet.transport.UDP $hb/udp]

# Adding a Data Counter to each host:
mkdir drcl.comp.tool.DataCounter $hb/counter

#Setting up T.38 Fax Senders/Receivers:

$TCPModule addPort "up" "t38fax"
set T38FaxReceiver [mkdir ketan.sip.T38Receiver $hb/t38FaxReceiver]
connect -c $hb/t38FaxReceiver/down@ -and $hb/tcp/up@

set T38FaxSender [mkdir ketan.sip.T38Sender $hb/t38FaxSender]
connect -c $hb/t38FaxSender/down@ -and $hb/tcp/t38fax@up

puts "Building Nodes ..."
$rb build [! n?]
$hb build [! h?]

#Setting up UAs:
set SipUA1 [mkdir ketan.sip.SipUA h0/ua]
set SipUA2 [mkdir ketan.sip.SipUA h3/ua]
$SipUA1 setName "SIP User Agent 1"
$SipUA2 setName "SIP User Agent 2"
$SipUA1 setAddress 0
$SipUA2 setAddress 3
$SipUA1 setNodeViaField "SIP/2.0/UDP here.com:5060"
$SipUA2 setNodeViaField "SIP/2.0/UDP there.com:5060"
# $SipUA1 setAlwaysUseProxyServer true
# $SipUA2 setAlwaysUseProxyServer true
```

```

connect -c h0/ua/down@ -and h0/udp/5060@up
connect -c h3/ua/down@ -and h3/udp/5060@up

# Fax port configuration:

connect -c h3/ua/faxPort@down -to h0/t38FaxSender/faxPort@down
connect -c h3/ua/faxPort@down -to h3/t38FaxReceiver/faxPort@down

# For sending a BYE request after fax transfer completes:
connect -c h3/t38FaxReceiver/notify@ -to h3/ua/down@

#Setting up proxy servers:
set SipPS1 [mkdir ketan.sip.SipPS n4/ps]
set SipPS2 [mkdir ketan.sip.SipPS n5/ps]
$SipPS1 setName "SIP Proxy Server 1"
$SipPS2 setName "SIP Proxy Server 2"
$SipPS1 setAddress 4
$SipPS2 setAddress 5
mkdir drcl.inet.transport.UDP n4/udp
mkdir drcl.inet.transport.UDP n5/udp
connect -c n4/ps/down@ -and n4/udp/5060@up
connect -c n5/ps/down@ -and n5/udp/5060@up

$SipUA1 setConfiguredProxyServerAddress 4
$SipUA2 setConfiguredProxyServerAddress 5
$SipPS1 setOtherNetworkProxyServerAddress 5
$SipPS2 setOtherNetworkProxyServerAddress 4
$SipPS1 setRegisteredNode 0
$SipPS2 setRegisteredNode 3
$SipPS1 setNodeViaField "SIP/2.0/UDP SIP Proxy Server 1:5060"
$SipPS2 setNodeViaField "SIP/2.0/UDP SIP Proxy Server 2:5060"

# Configure the bottleneck bandwidth and buffer size
! n1 setBandwidth 1 1.0e5; # 100Kbps at interface 1
! n1 setBufferSize 1 6000; # ~10 packets at interface 1

puts "Setting up static routes..."
java::call drcl.inet.InetUtil setupRoutes [! h0] [! h3]
"bidirection"
java::call drcl.inet.InetUtil setupRoutes [! h0] [! n4]
"bidirection"
java::call drcl.inet.InetUtil setupRoutes [! n4] [! n5]
"bidirection"
java::call drcl.inet.InetUtil setupRoutes [! n5] [! h3]
"bidirection"
! h0/tcp setPeer 3
! h3/tcp setPeer 0
# Realistically, these should be set from "destination" fields of UAs
puts "Creating The Plotters..."
set plot1 [mkdir drcl.comp.tool.Plotter .plot1]
set file1 [mkdir drcl.comp.io.FileComponent .file1]
$file1 open "SimPlot.plot"
connect -c $plot1/.output@ -to $file1/in@

attach -c $plot1/0@0 -to h0/tcp/cwnd@
attach -c $plot1/3@0 -to h3/tcp/cwnd@

set tm1 [mkdir drcl.net.tool.TrafficMonitor .tm1]
connect -c h3/csl/6@up -to $tm1/in@
connect -c $tm1/bytecount@ -to $plot1/3@1
```

```
112
113 attach -c $plot1/3@2 -to h3/tcp/rcv/seqno@
114 attach -c $plot1/0@2 -to h0/tcp/rcv/seqno@
115
116 attach -c h0/counter/in@ -to h0/csl/6@up
117 attach -c h3/counter/in@ -to h3/csl/6@up
118
119 puts "Setting up the NAM trace..."
120 set nam [java::call drcl.inet.InetUtil setNamTraceOn [! .] \
"SimNAMTrace.nam" [_to_string_array "red blue black orange"]]
121
122 puts "Building the scenario is finished now, you can proceed with
running the simulation ..."
```

Analysis of Scenario:

-**Line 9:** Creating the network element itself to hold the network components.

-Lines 12-13: Creating and configuring the link element.

-Line 14: Creating the adjacency matrix; the matrix which defines the topology by determining the neighboring nodes of each node.

-Line 15: Creating the network by calling createTopology() and passing the network component, adjacency matrix and link component as arguments.

-Lines 19-20: Creating a typical node builder and naming it rb.

-Line 22: Duplicating the rb object and naming it hb for building the hosts after further customizing the host builder (hb).

-Lines 24-25: Creating the transport protocol modules (The UDP module is used for signaling and the TCP module is used for the fax data transfer).

-Line 28: Putting a data packet counter in each of the hosts.

-Lines 32-37: Creating T.38 sender and receiver modules and putting them in the host builder and connecting the modules to the TCP module.

-Lines 40-41: Building the hosts and routers using the previously configured host builder and node builder.

-Lines 44-51: Creating the SIP user agent modules in the hosts and configuring their names, addresses and nodeViaField parameters.

-Lines 53-54: Setting whether the user agents always have to use the proxy servers for signaling or they are allowed to bypass them if they can.

-Lines 56-57: Configuring the SIP modules to use UDP as transport.

-Lines 61-65: Setting a mechanism for auto-starting the fax transfer immediately after signaling is complete and also auto-starting the signaling (the last BYE-OK) after the fax transfer is complete.

-Lines 68-77: Creating SIP proxy server modules in the nodes and configuring their names and addresses. The UDP modules are also setup in the nodes and proxy server modules are set to use them for transport.

-Lines 79-86: Further customizing the SIP modules: Setting user agents' configured proxy server addresses, making each proxy server know the other network's proxy server, setting each proxy server's registered node and setting the nodeViaField of each proxy server.

-Lines 89-90: A sample bottleneck is incorporated in the node 1 to examine its effect on fax data transfer parameters.

-Lines 93-98: Setting up the nodes routing table entries and TCP peers.

-Lines 101-117: Creating the plotters and data counter instruments. Plotters monitor fax data throughput, congestion window and received packets sequence numbers. The parameters are shown in real-time manner during the simulation and since the plots are saved into file, they can be seen and examined later as well.

-Line 120: The nodes are configured to output packets traces in the format understandable by Network Animator.

6.4.3 Scenario Running

```
puts ""
puts "Starting the Simulation..."
set sim [attach_simulator .]

puts "Fax initiation..."
puts "Negotiating the fax parameters with the other party..."

##########################################
# Fax Signaling and sending
##########################################

set message [$SipUA1 constructMessage "INVITE" "SIP/2.0/UDP here.com:5060" "sip: user@there.com" "0" 1234@here.com "application/sdp"]

$SipUA1 sendINVITE $message




##########################################
## SIP message constructing method syntax:
##
## SIPMessage constructMessage(String startLine, String via, String to, String from, String ##callID, String contentType)
##########################################

##########################################
## Further SIP message headers customization can be done using this syntax:
```

```
27 ##
28 ## set hd [$message returnSIPHeaders]
29 ## set SDPhd [$message returnSDPHeaders]
30 ## puts "SIP header (Call-ID):"
31 ## $hd getProperty "Call-ID"
32 ## puts "SDP header (v):"
33 ## $SDPhd getProperty "v"
34 ## $hd setProperty "CSeq" "1"
35 ##########################################
36
37 ##########################################
38 ## SIP message sending method syntax:
39 ##
40 ## sendINVITE(SIPMessage toBeSentMessage)
41 ##########################################
42
43
44 ### The command for viewing the plots:
45 ### java drcl.comp.tool.Plotter SimPlot.plot
```

Analysis:

Line 3: Attaching simulation run-time to the network.

Line 12: Creating an SIP message in user agent 1 using its constructMessage() method according to the syntax given in line 22.

Line 14: The constructed message is then sent using the sendINVITE() method of user agent 1 (The syntax is given in line 40).

Lines 19-45: These are provided to serve as reference and because each line is prefixed with a #, it doesn’t get executed.

6.5 SIMULATION RESULTS

6.5.1 The Simulated SIP Call Flow

Please note that the generation and processing of informational class responses have not been simulated. The simulated call flow is depicted in Figure 6.2.

6.5.2 Terminal Output

As shown in the terminal output following Figure 6.2, after building and configuring the network, the simulation starts by user agent 1 sending an INVITE request. From there, one can track the call flow from the terminal output and also from Figure 6.2. As can be seen, user agents contact each other directly after knowing each other’s address from the SIP contact header.

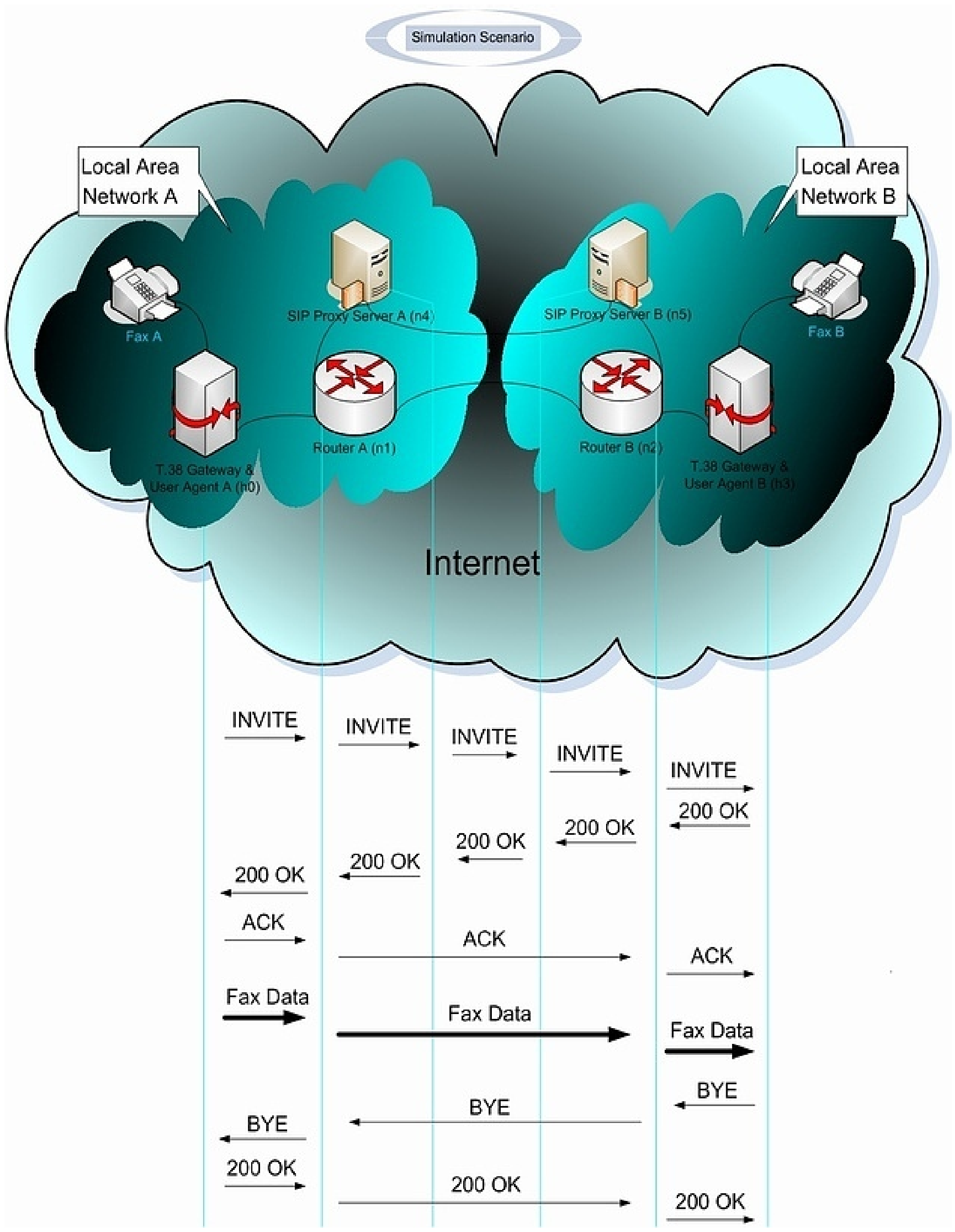
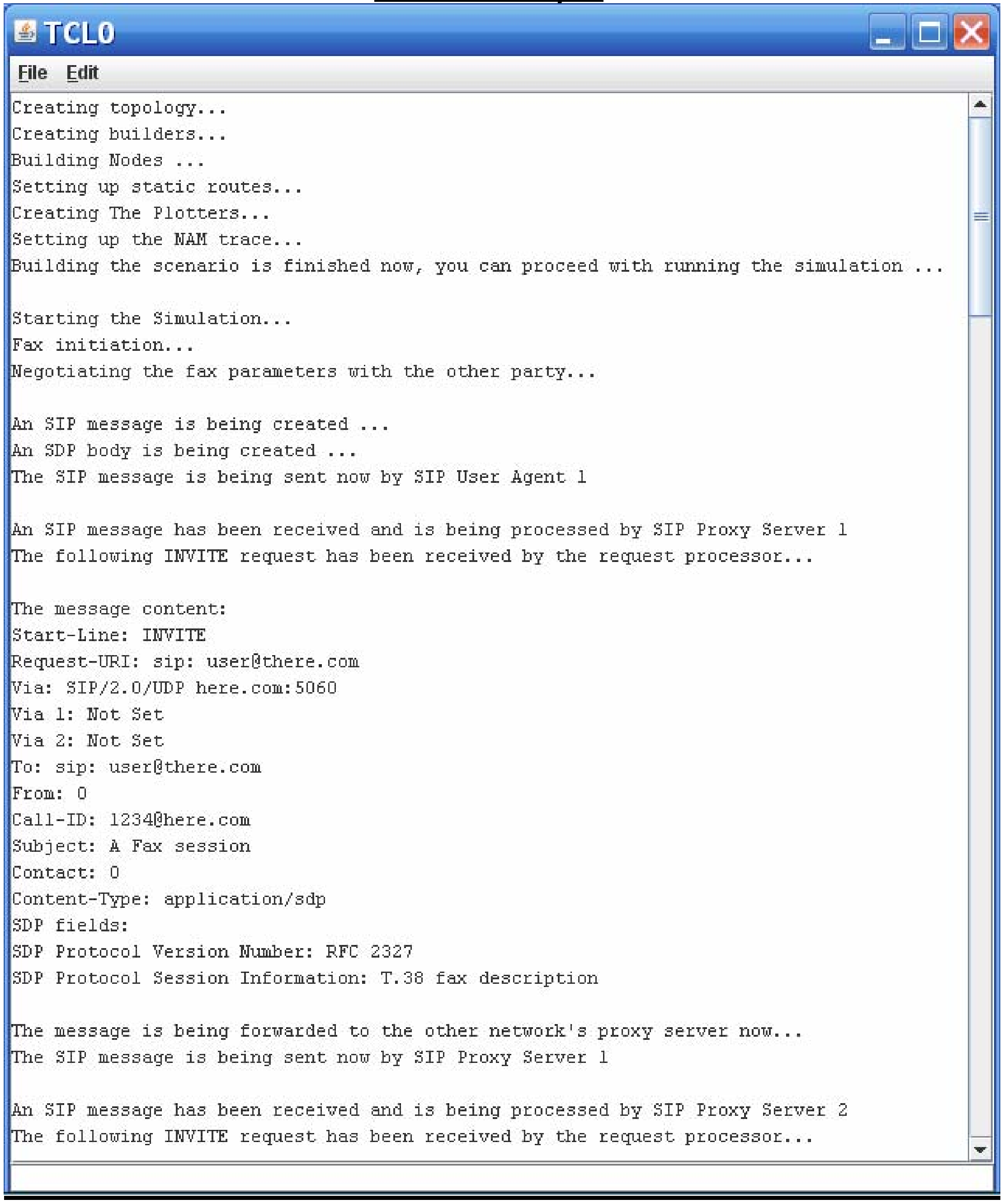


Figure 6.2 The Simulated SIP Call Flow.

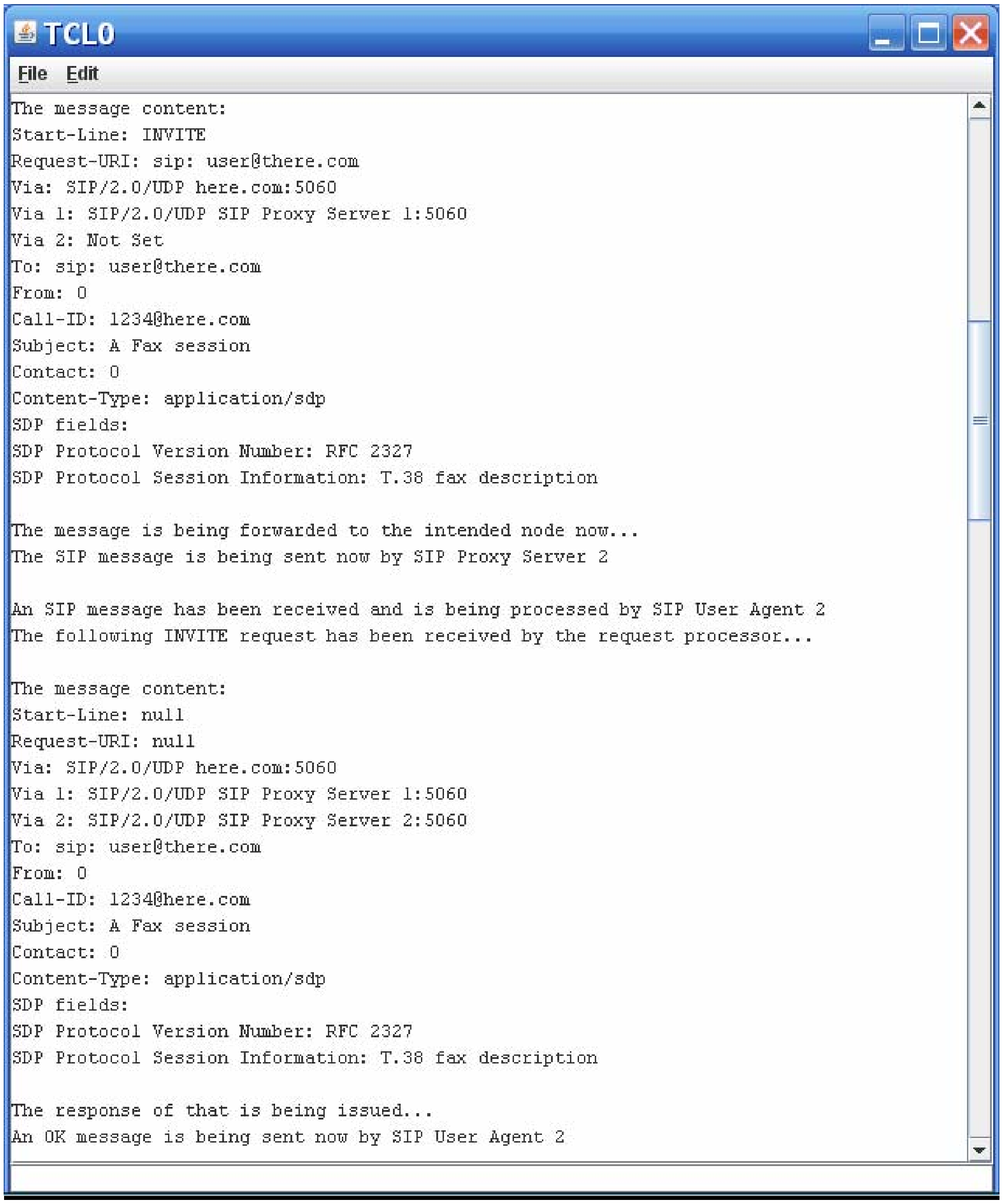
***Terminal Output* 1**



```
Creating topology...
Creating builders...
Building Nodes ...
Setting up static routes...
Creating The Plotters...
Setting up the NAM trace...
Building the scenario is finished now, you can proceed with running the simulation ...

Starting the Simulation...
Fax initiation...
Negotiating the fax parameters with the other party...

An SIP message is being created ...
An SDP body is being created ...
The SIP message is being sent now by SIP User Agent 1

An SIP message has been received and is being processed by SIP Proxy Server 1
The following INVITE request has been received by the request processor...

The message content:
Start-Line: INVITE
Request-URI: sip: user@there.com
Via: SIP/2.0/UDP here.com:5060
Via 1: Not Set
Via 2: Not Set
To: sip: user@there.com
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 0
Content-Type: application/sdp
SDP fields:
SDP Protocol Version Number: RFC 2327
SDP Protocol Session Information: T.38 fax description

The message is being forwarded to the other network's proxy server now...
The SIP message is being sent now by SIP Proxy Server 1

An SIP message has been received and is being processed by SIP Proxy Server 2
The following INVITE request has been received by the request processor...
```

It has been shown in terminal output-1 that Fax initiation is started. The fax parameters have been negotiated with the other party and an SIP message and an SDP body has been created. The SIP message has been sent by SIP User Agent 1. An SIP message has been received and is being processed by SIP Proxy Server 1. INVITE request has been received by the requested processor. Then message is being forwarded to the other network’s proxy server. An SIP message has been received and is being processed by SIP Proxy Server 2.

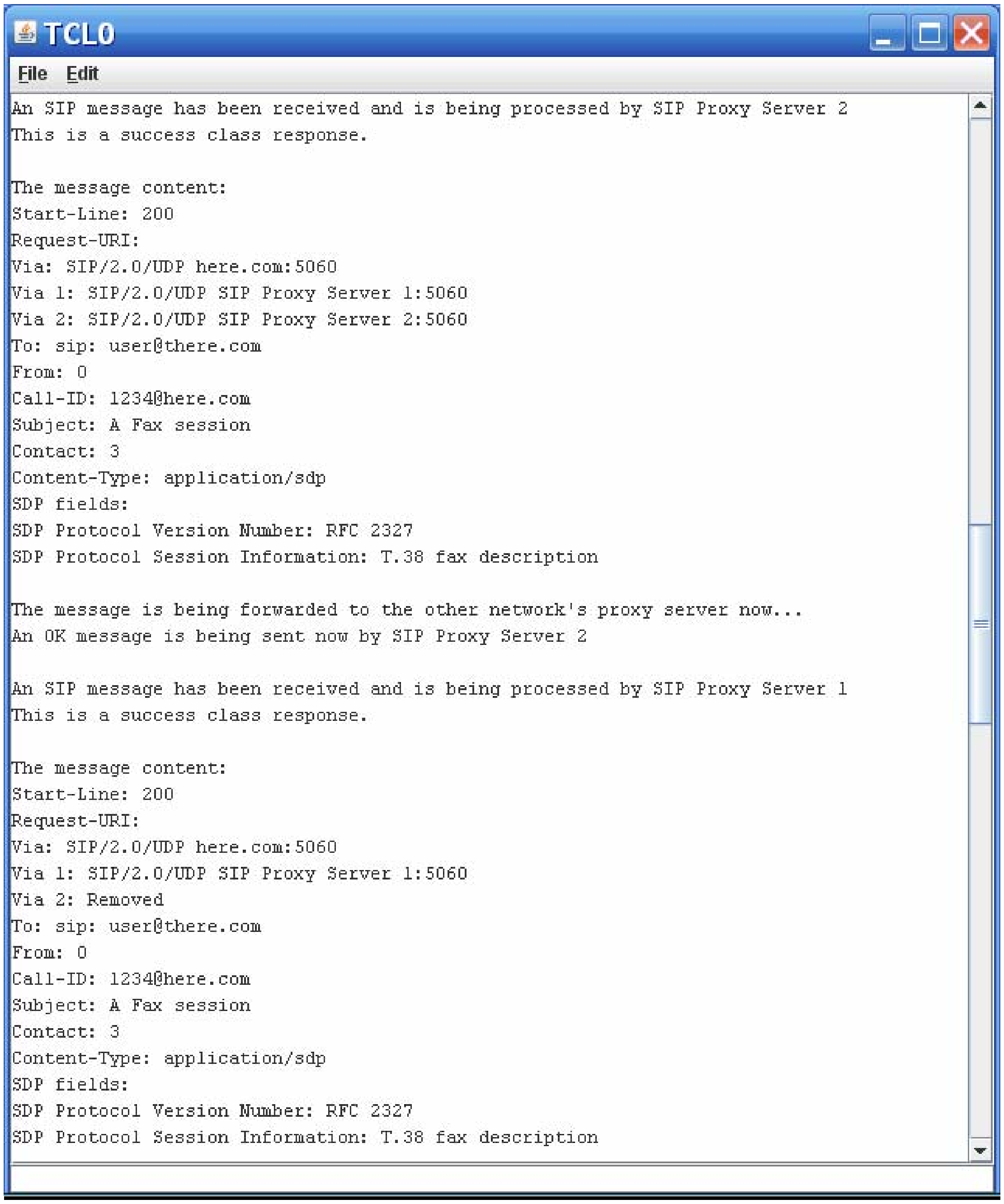
***Terminal Output* 2**



```
The message content:
Start-Line: INVITE
Request-URI: sip: user@there.com
Via: SIP/2.0/UDP here.com:5060
Via 1: SIP/2.0/UDP SIP Proxy Server 1:5060
Via 2: Not Set
To: sip: user@there.com
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 0
Content-Type: application/sdp
SDP fields:
SDP Protocol Version Number: RFC 2327
SDP Protocol Session Information: T.38 fax description

The message is being forwarded to the intended node now...
The SIP message is being sent now by SIP Proxy Server 2

An SIP message has been received and is being processed by SIP User Agent 2
The following INVITE request has been received by the request processor...

The message content:
Start-Line: null
Request-URI: null
Via: SIP/2.0/UDP here.com:5060
Via 1: SIP/2.0/UDP SIP Proxy Server 1:5060
Via 2: SIP/2.0/UDP SIP Proxy Server 2:5060
To: sip: user@there.com
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 0
Content-Type: application/sdp
SDP fields:
SDP Protocol Version Number: RFC 2327
SDP Protocol Session Information: T.38 fax description

The response of that is being issued...
An OK message is being sent now by SIP User Agent 2
```

It has been shown in terminal output-2 that the SIP message has been sent by SIP Proxy Server 2. An SIP message has been received and is being processed by SIP User Agent 2. INVITE request has been received by the requested processor. Then response of that is being issued. Then OK message has been sent by SIP User Agent 2.

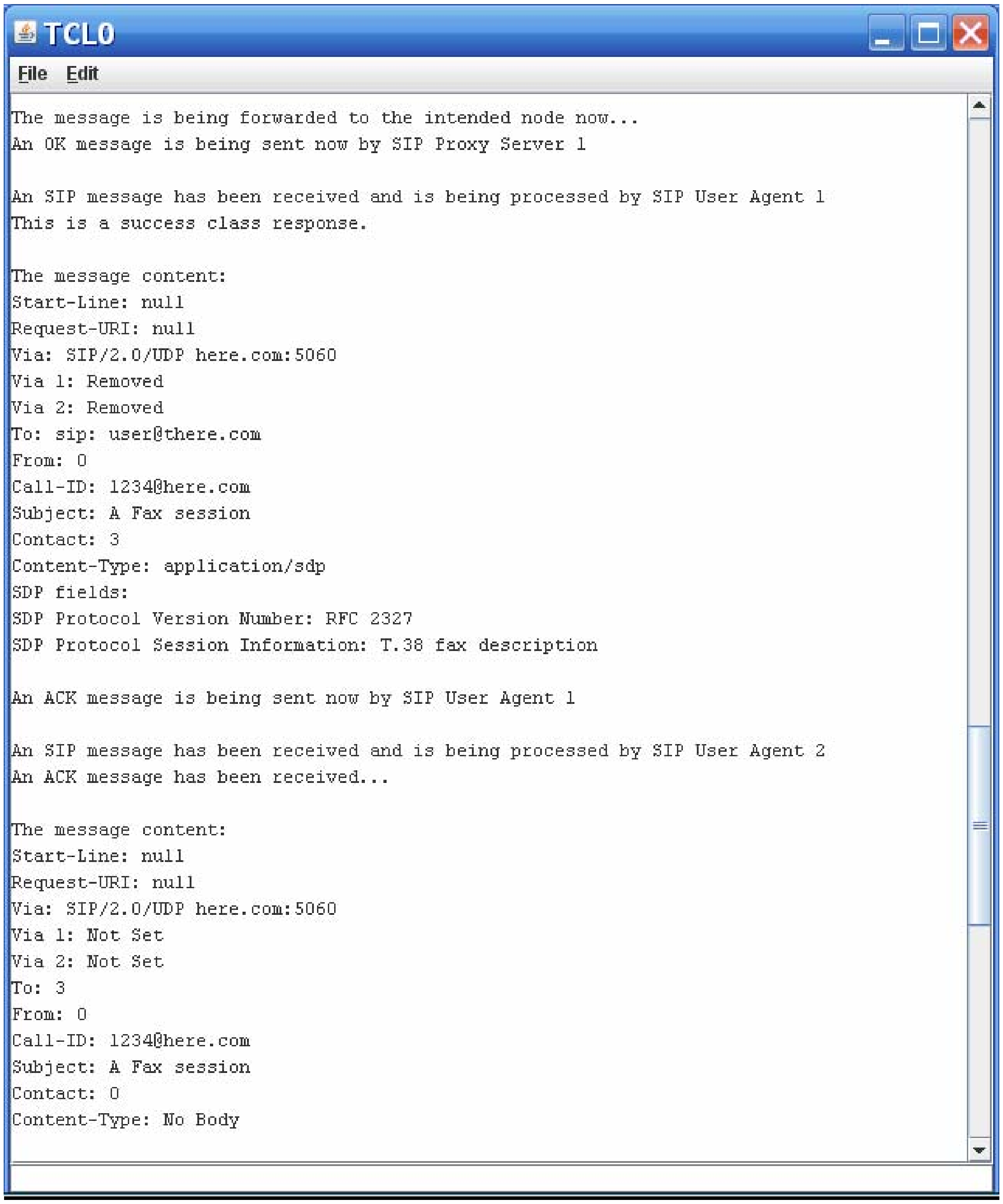
<u>Terminal Output</u> 3



```
An SIP message has been received and is being processed by SIP Proxy Server 2
This is a success class response.

The message content:
Start-Line: 200
Request-URI:
Via: SIP/2.0/UDP here.com:5060
Via 1: SIP/2.0/UDP SIP Proxy Server 1:5060
Via 2: SIP/2.0/UDP SIP Proxy Server 2:5060
To: sip: user@there.com
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 3
Content-Type: application/sdp
SDP fields:
SDP Protocol Version Number: RFC 2327
SDP Protocol Session Information: T.38 fax description

The message is being forwarded to the other network's proxy server now...
An OK message is being sent now by SIP Proxy Server 2

An SIP message has been received and is being processed by SIP Proxy Server 1
This is a success class response.

The message content:
Start-Line: 200
Request-URI:
Via: SIP/2.0/UDP here.com:5060
Via 1: SIP/2.0/UDP SIP Proxy Server 1:5060
Via 2: Removed
To: sip: user@there.com
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 3
Content-Type: application/sdp
SDP fields:
SDP Protocol Version Number: RFC 2327
SDP Protocol Session Information: T.38 fax description
```

It has been shown in terminal output-3 that an SIP message has been received and is being processed by SIP Proxy Server 2. This is success class response. Then message is being forwarded to the other network's proxy server. Then OK message has been sent by SIP Proxy Server 2.An SIP message has been received and is being processed by SIP Proxy Server 1. This is success class response.

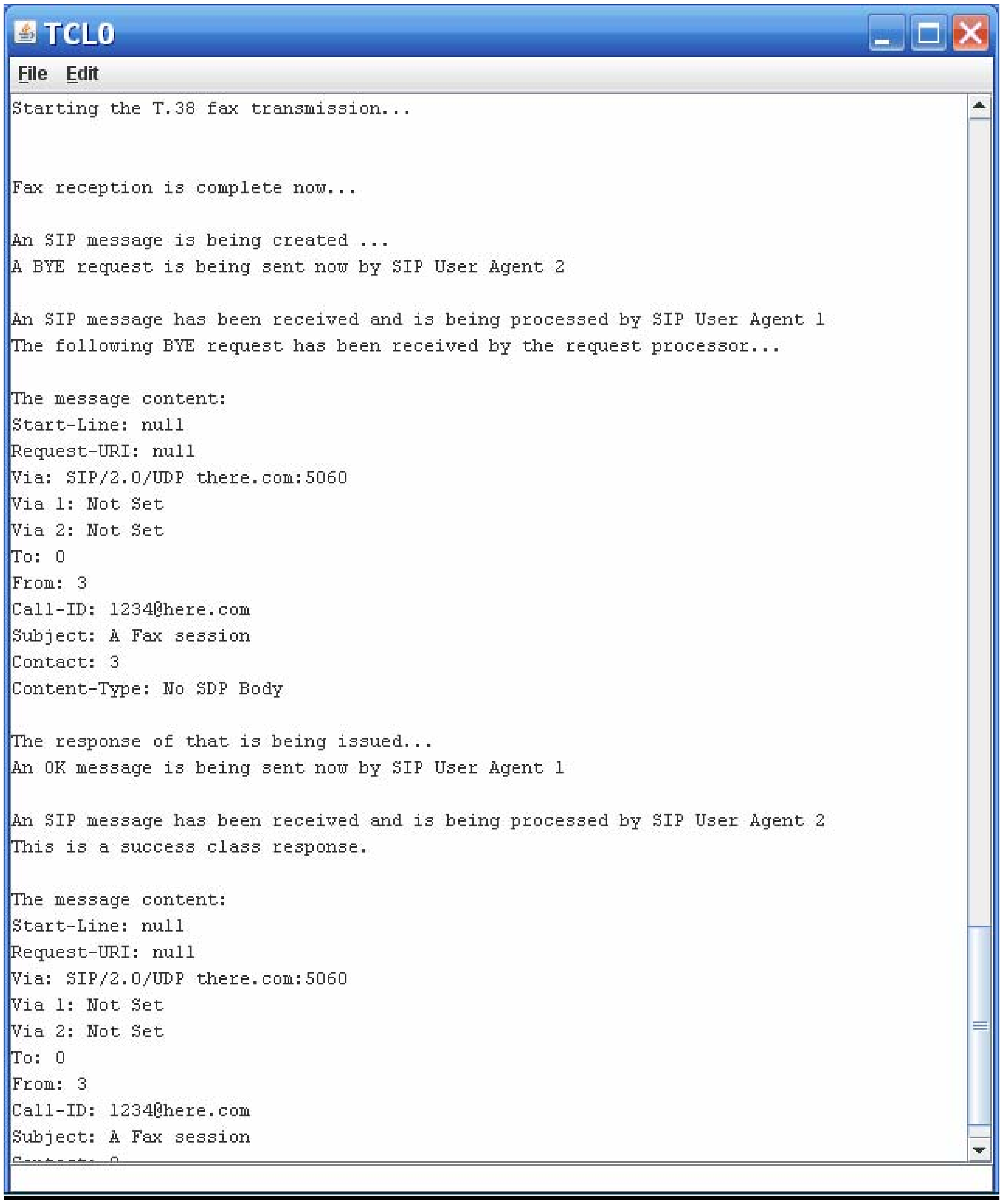
***Terminal Output* 4**



```
The message is being forwarded to the intended node now...
An OK message is being sent now by SIP Proxy Server 1

An SIP message has been received and is being processed by SIP User Agent 1
This is a success class response.

The message content:
Start-Line: null
Request-URI: null
Via: SIP/2.0/UDP here.com:5060
Via 1: Removed
Via 2: Removed
To: sip: user@there.com
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 3
Content-Type: application/sdp
SDP fields:
SDP Protocol Version Number: RFC 2327
SDP Protocol Session Information: T.38 fax description

An ACK message is being sent now by SIP User Agent 1

An SIP message has been received and is being processed by SIP User Agent 2
An ACK message has been received...

The message content:
Start-Line: null
Request-URI: null
Via: SIP/2.0/UDP here.com:5060
Via 1: Not Set
Via 2: Not Set
To: 3
From: 0
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 0
Content-Type: No Body
```

It has been shown in terminal output-4 that the message has been forwarded to the intended node. The OK message has been sent by SIP Proxy Server 1. An SIP message has been received and is being processed by SIP User Agent 1. This is success class response. Then ACK message has been sent by SIP User Agent 1. An SIP message has been received and is being processed by SIP User Agent 2. And then ACK has been received.

***Terminal Output* 5**



```
Starting the T.38 fax transmission...


Fax reception is complete now...

An SIP message is being created ...
A BYE request is being sent now by SIP User Agent 2

An SIP message has been received and is being processed by SIP User Agent 1
The following BYE request has been received by the request processor...

The message content:
Start-Line: null
Request-URI: null
Via: SIP/2.0/UDP there.com:5060
Via 1: Not Set
Via 2: Not Set
To: 0
From: 3
Call-ID: 1234@here.com
Subject: A Fax session
Contact: 3
Content-Type: No SDP Body

The response of that is being issued...
An OK message is being sent now by SIP User Agent 1

An SIP message has been received and is being processed by SIP User Agent 2
This is a success class response.

The message content:
Start-Line: null
Request-URI: null
Via: SIP/2.0/UDP there.com:5060
Via 1: Not Set
Via 2: Not Set
To: 0
From: 3
Call-ID: 1234@here.com
Subject: A Fax session
```

And finally it has been shown in terminal output-5 that T.38 fax transmission has been started. After some time it has been completed. Then BYE request has been received by the requested processor. Then OK message has been sent by SIP User Agent 1. An SIP message has been received and is being processed by SIP User Agent 2.

6.5.3 Packets Traces Analyzed With Network Animator

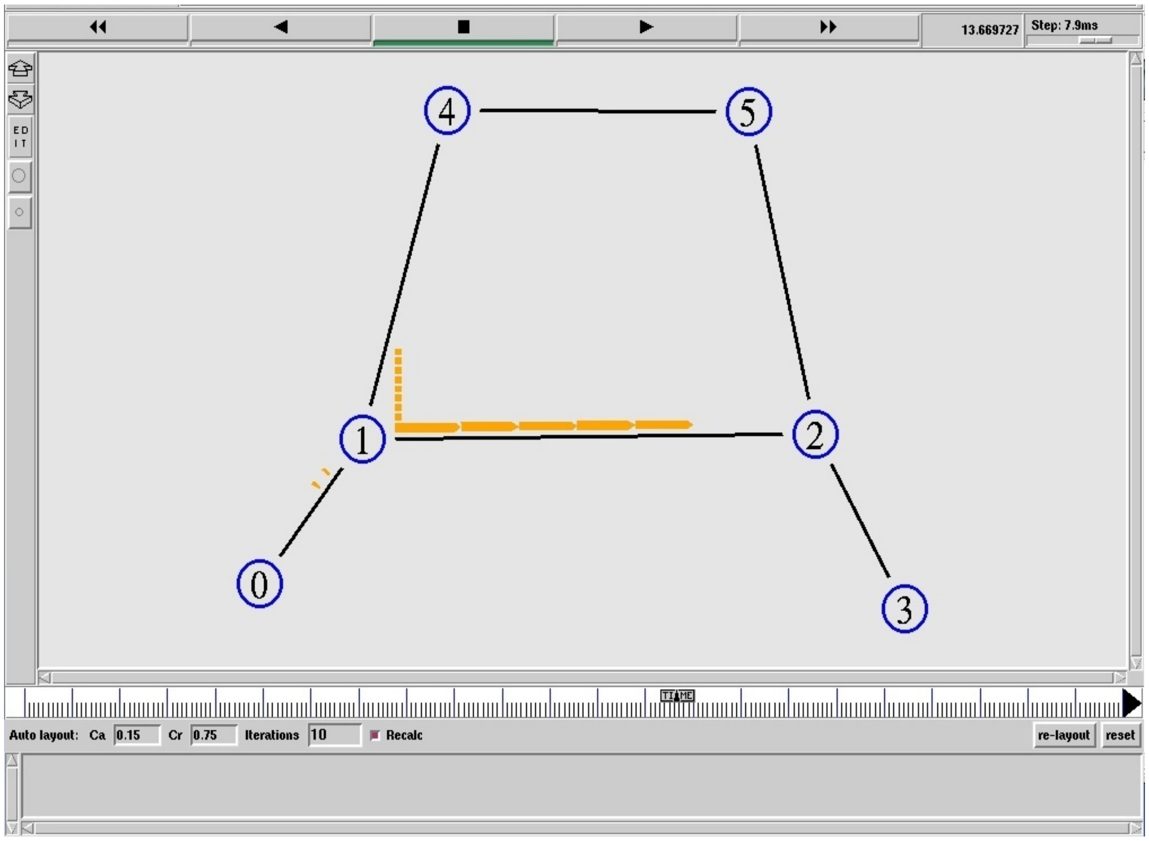


Figure 6.3 Packets Traces Analyzed With Network Animator.

With the help of Network Animator (NAM), the simulation's packets traces can be gathered and analyzed after the simulation is finished and output file is opened using NAM. A snapshot of my simulation scenario analyzed using this program is provided in Fig. 6.3. A time instance when fax data is being sent directly from user agent 1 to user agent 2, bypassing proxy servers, is shown in Figure 6.3.

6.5.4 Possible Fax Data Transfer Analyses

Fax data analysis is not a concern of this thesis and the carried out simulation. But for depicting the types of possible analyses which can be done using J-Sim, as pointed out in the scenario building TCL script lines 89-90, a bottleneck is incorporated in node 1 and its effect in fax data transfer is shown in the following figures:

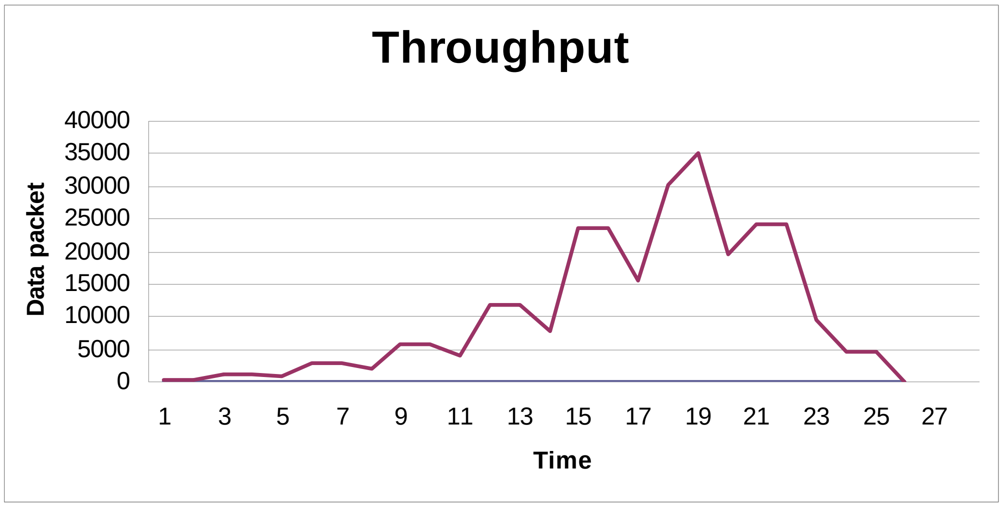


Figure 6.4 Throughput

Throughput, defining it "as the average number of packets successfully delivered per unit time". Here this data may be delivered over a physical link that is passing through a network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per timeslot.

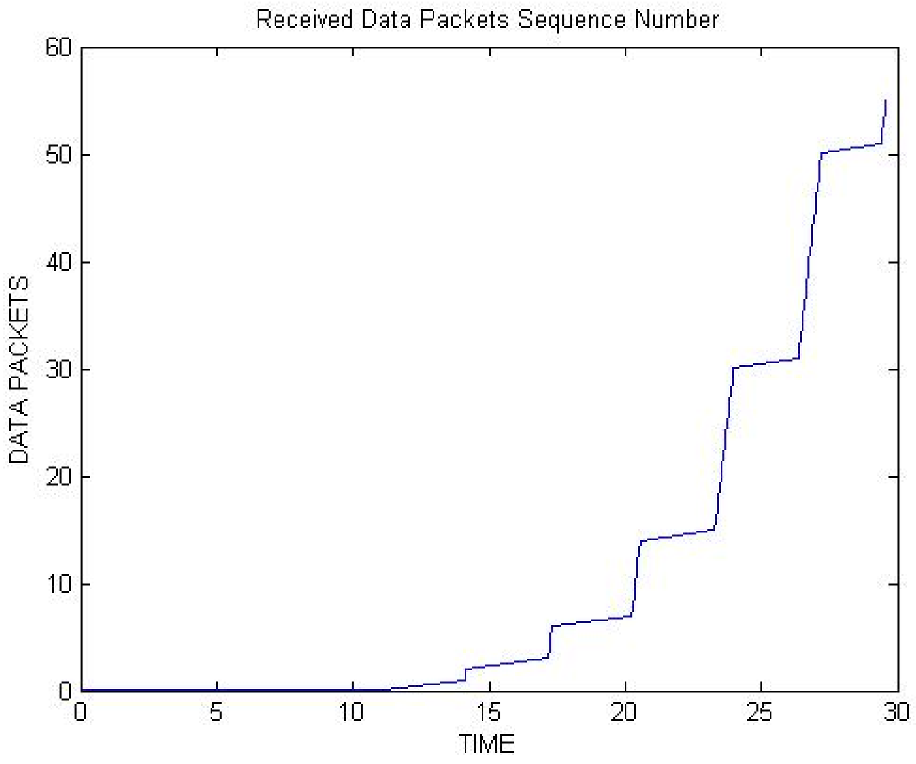


Figure 6.5 Received Data Packets Sequence Number.

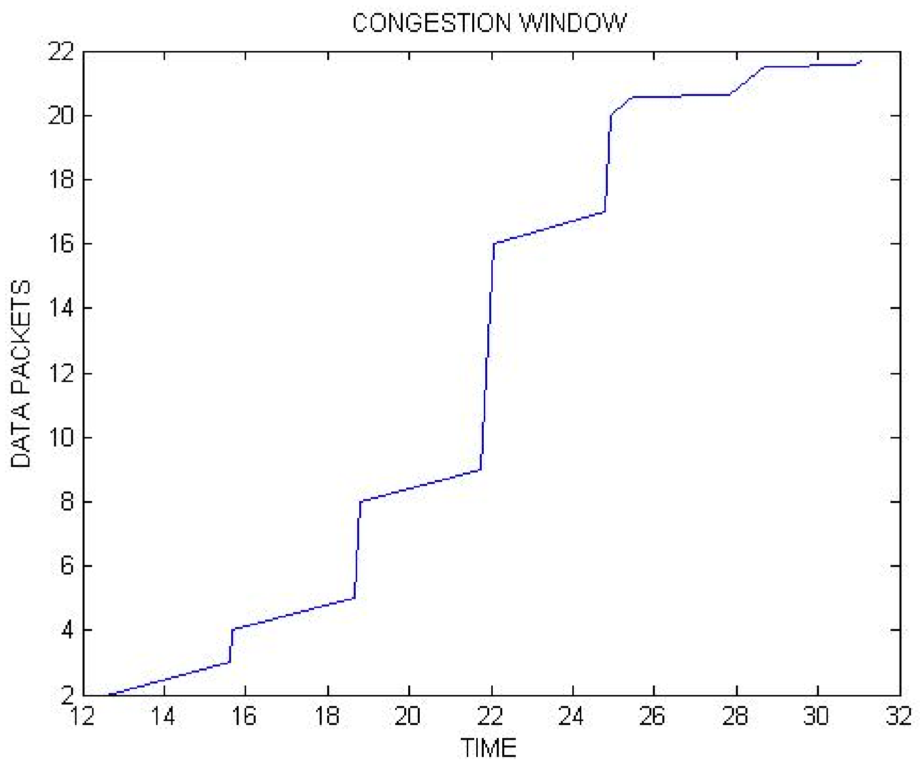


Figure 6.6 Congestion Window

Table 6.4 Results

Throughput		**Receive Data Packet Seq. No.**		**Congestion Window Size**	
Time	Data	Time	Data	Time	Data
10	0	10	0	10	1
15	$0.4*10^4$	15	2	15	4
20	$0.5*10^4$	20	9	20	7
25	$2.4*10^4$	25	30	25	21
28	$3.5*10^4$	27	51	27	21
30	$2.5*10^4$	28	51	30	22
35	$0.0*10^4$	29	53	31	22

The congestion window determines the number of bytes that can be outstanding at any time. This is a means of stopping the link between two places from getting overloaded with too much traffic. The size of this window is calculated by estimating how much congestion there is between the two places. Once this size is calculated that is the maximum number of bytes that can be transmitted without acknowledgment that they have been received (done through ACK packets). Basically the size of the window, to a large degree, controls the speed of transmission as transmission pauses until there is acknowledgment.

It has been point out that in my simulation; fax packets do not visit the proxy servers. This is a natural behavior, since proxy servers are highly specialized resources and I do not want to unnecessarily overload them. Hence, proxy servers will be able to handle more calls in the network.

Here the discussion regarding simulation scenario building and configuration and corresponding analyses comes to an end. In the next chapter a summary of all the important points of this thesis including this chapter is given.

Chapter 7

Concluding Remarks

7.1 CONCLUSIONS

What it has been intended to do was investigating whether fax parameters details could be negotiated using SIP/SDP. In this direction, session establishment, starting a typical file transfer, which served as a demonstration of T.38 fax transfer, and the subsequent session teardown, after file transfer was complete, were demonstrated in this thesis, through computer simulation. Call flow could be tracked and analyzed using the terminal output and also the simulation schematic diagram; the former was in complete agreement with the claimed results regarding the SIP capabilities.

Based on the investigated simulation scenario, it has been showed that SIP nicely lends itself to the task. Although only few constant symbolic SDP fields were used and hence present in the terminal output, but as mentioned before, the simulation results provided a good starting point for the full definition and usage of SDP attribute fields to thoroughly specify the T.38 fax parameters. This simulation scenario and its results exhibited the potential success of the proposed SIP/SDP combination for real-time fax session establishment, management and tear-down.

Another important analysis carried out in this simulation was utilization of SIP contact header for reducing the load on proxy servers for FoIP application which is a highly desirable feature. As shown in the terminal output of the simulator, after building and configuring the network, the simulation started by user agent 1 sending an INVITE request. From there, the call flow could be tracked from the terminal output and as could be seen, user agents contacted each other directly after knowing each other's address from the SIP contact header.

7.2 POSSIBLE FUTURE WORK

Specific T.38 protocol SDP attributes have not been studied in this thesis. As a natural next step, if the thorough investigation of real-time fax is contemplated, one can refer to this IETF Internet-Draft: "SIP Support for Real-time Fax: Call Flow Example and Best Current Practices” [20], which can serve as a very good starting point. Addition of few more capabilities to the proxy server can be a good proposition as well. Additional capabilities can be: handling multiple transactions at the same time, stateful operation of the proxy server, having location server and multiple registered nodes, implemented either in the proxy server code itself or in separate entities, to name a few.

References

[1] Bur Goode ”Voice over Internet Protocol (VoIP)” , Proceedings of the IEEE, Vol.90, No.9, September 2002

[2] “Voice and fax over IP”, The International Engineering Consortium, http://www.iec.org

[3] James Irvine and David Harle, “Data Communications and Networks: An Engineering Approach”, John Wiley 2002

[4] Lillian Goleniewski, “Telecommunications Essentials”, Addison-Wesley 2002

[5] Bill Douskalis, “IP Telephony: The Integration of Robust VoIP Services”, Prentice Hall 2000

[6] Kimmo Ahonen, Juha Koskelainen, “Transport Control Protocol”, University of Helsinki, October 1998

[7] Henning Schulzrinne (Columbia University), Jonathan Rosenberg (Bell Laboratories Lucent Technologies), “Internet Telephony: Architecture and Protocols an IETF Perspective”, July 1998

[8] “An Introduction to IP Telephony”, Mockingbird Networks

[9] Alan B. Johnston, “SIP: Understanding the Session Initiation Protocol”, Artech House 2001

[10] “Voice Performance over packet-based networks”, Alcatel, October 2002

[11] “IP Telephony Design Guide”, Alcatel, April 2003

[12] Mike Gray, “FAX Technology Tutorial and Testing Issues”, Agilent Technologies, February 2002

[13] ITU-T Recommendation T.4, “Standardization of Group 3 facsimile terminals for document transmission”, Terminals For Telematic Services, July 2007

[14] “T.38 and the Future of Fax”, Intel 2003

[15] K. Toyoda, H. Ohno, J. Murai and D. Wing, " A Simple Mode of Facsimile Using Internet Mail",RFC 2305, IETF, March 1998.

[16] K.Mimura, K.Yokoyama, T.Satoh and C.Kanaide, " Internet FAX Gateway Functions", Internet Draft, IETF, draft-ietf-fax-gateway-protocol-09.txt, April 14 2003.

[17] Y.Rafiq, O.Bashir,S.I.Shah and S,A.Khan, “FoIP gateways-architectures, implementation and QoS issues”, IEEE International Multi Topic Conference, 2001.

IEEE INMIC 2001. Technology for the 21st Century. Proceedings., Page(s): 87 - 92, 28-30 Dec. 2001

[18] “eBusiness Companies Can Reap Rewards by Faxing over IP”, DCI 1999, http://www.dci.com

[19] Phelim O’Doherty, “SIP Specifications and the Java Platforms”, 2003 Sun Microsystems

[20] Jean-Francois Mule and Jieying Li, "SIP Support for Real-time Fax: Call Flow Examples And Best Current Practices", draft-ietf-realtimefax-01.txt, August 2003.

[21] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg, "SIP: Session Initiation Protocol",RFC 2543, IETF, March 1999.

[22] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson. R. Sparks, M. Handley "SIP: Session Initiation Protocol",RFC 3261, IETF, June 2007.

[23] A. Johnston, S. Donovan, R. Sparks, C. Cunningham and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples", RFC 3665, IETF, December 2007.

[24] M. Handley " SDP: Session Description Protocol",RFC 2327, IETF, April 1998.

[25] J. Rosenberg and H. Schulzrinne, " An Offer/Answer Model with the Session Description Protocol (SDP)",RFC 3264, IETF, June 2002.

[26] Xiaotao Wu and Henning Schulzrinne, ”Programmable End System Services Using SIP”, IEEE International Conference on Communications ICC '03, Volume: 2 , Page(s): 789 -793, 11-15 May 2003

[27] G.Stojsic, R.Radovic and S.Srbljic, “Formal Definition of SIP Proxy Behavior”, EUROCON'2001, International Conference on Trends in Communications, Volume: 2, Page(s): 289 -292, 4-7 July 2001

[28] Phelim O’Doherty and Mudumbai Ranganathan, “JAIN SIP Tutorial", Sun Microsystems 2008

[29] JAIN team, “JAIN Technology”, Sun Microsystems 2007

[30] http://www.j-sim.org