An Analytical Simulator for Deploying IP Telephony

By

Meshal Abdulaziz Almashari

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

KING FAHD UNIVERSITY

OF PETROLEUM & MINERALS

An Analytical Simulator for Deploying IP Telephony

BY

Meshal Abdulaziz Almashari

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

May 2006

Dhahran, Saudi Arabia

May, 2006

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SUADI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis written by **MESHAL ABDULAZIZ ALMASHARI** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

Dr. Khalid Salah (Thesis Advisor)

Dr. Nasir Al-Darwish (Member)

Dr. Muhammed Alsuwaiyel (Member)

Kanaan Faisal

Department Chairman
Dr. Kanaan Faisal

Dean of Graduate Studies
Dr. Mohammad A. Al-Ohali

June, 18 2006

Date

Dedication

This thesis is dedicated to my wife

Mona

Acknowledgment

First of all, all praise and thanks are due to Allah the Almighty for his blessings. I would also like to thank King Fahd University of Petroleum and Minerals for their support.

I wish to express my sincere gratitude to my adviser Dr. Khalid Salah, who guided this effort and with his valuable support and contribution drove it to success. Appreciation also goes to my thesis committee members, Dr. Nasir Darwish and Dr. Muhammed Alsuwaiyel, and to my colleague Mr. Abdulaziz Al-Khoraidly for their support. I would also like to thank the Collage of Computer Science and Engineering for their ongoing support.

Appreciation also goes to my colleagues and management at Saudi Aramco for their encouragement and support. Special thanks go to my division head Mr. Barjas Al-Barjas for his patients and continuous encouragement.

I would like to express my deepest gratitude for the constant support, understanding, prayers and love that I received from my wife and for offering words of encouragement at challenging moments.

Finally, my sincere appreciation goes to my mother who holds an MS degree in mathematics and is the person behind any success in my life and who is always there for me with her prayers and valuable support. I simply cannot thank her enough.

Table of Contents

List of Figures

List of Tables

Abstract

NAME: Meshal Abdulaziz Almashari

TITLE: An Analytical Simulator for Deploying IP Telephony.

MAJOR FIELD: Computer Science.

DATE OF DEGREE: May 2006.

Multimedia applications require real-time quality of service assurance. Such applications need a well understanding of the data network they ride on. The performance of the data network also needs to be closely monitored to know what impact the application has and whether or not the network is capable of carrying it. However, due to the fact that today's data network runs many critical applications, data network engineers seek for a way to measure the impact of these applications before deploying them. This thesis will focus on VoIP as the multimedia application. We will assess the readiness of existing networks to support VoIP using an analytical simulator. Although there are some simulation tools that attempt to measure network readiness but there are no tools available that measure VoIP network readiness using mathematical analysis. Therefore the major contribution of this master thesis is to design and implement such a tool. We will also highlight the advantages of mathematical analysis and the drawbacks of simulation tools. Our tool will ask the user for the network topology of the current network and using delay and throughput analysis it will generate a report identifying the number of calls the network can support and the bottleneck. We will also compare our results with results obtained using a simulation tool such as OPNET.

خلاصة الرسالة

الإســـــم : مشعل عبدالعزيز المشاري

عنوان الرسالة : محاكي تحليلي لاطلاق خدمة الهاتف عبر الأي بي(IP)

التخصص : علوم الحاسب الآلي

تاريخ التخرج مايو 2005

تحتاج برامج الوسائط المتعددة إلى شبكة بيانات ذات جودة عالية. تلك البرامج تستلزم معرفة تامة بشبكة البيانات التي تعبر عليها. ويتوجب أيضاً متابعة أداء شبكة البيانات عن قرب لمعرفة أثر البرنامج عليها وما إذا كانت الشبكة قـــادرة على حمل البرنامج. وبما أن شبكة البيانات أصبحت في عصرنا تحمل الكثير من البرامج المهمة والحساسة، أصبحت مهمة مهندسي شبكة البيانات هي إيجاد طرق لقياس تأثير هذه البرامج على الشبكة قبل إطلاقها. هذه الرسالة ترتكز علـــى اعتبار خدمة الهاتف عبر الأي بي (IP) كبرنامج للوسائط المتعددة. سوف نقيم جاهزية الشبكة لدعم خدمات الهاتف عبر الأي بي (IP) باستخدام محاكي تحليلي. بالرغم من وجود بعض المحاولات لقياس جاهزية الشبكة إلا أنها تستخدم برامج محاكاة تقليدية لا ترتكز على التحليل الرياضي. لذلك فقد خصصنا القسم الأكبر من هذه الرسالة لتصميم وتنفيذ هذه الأداة. علاوة على ذلك، سوف نشير الى مزايا التحليل الرياضي وعيوب أدوات المحاكاة التقليدية. المحاكي الرياضي الذي طورناه في هذه الرسالة يقوم بسؤال المستخدم عن المخطط الحالي للشبكة ثم يقوم باستعمال قـــدرة الإخـــراج (throughput) والتأخر (delay) كمتغيرين للخروج بتقرير يحدد عدد المكالمات الهاتفية التي تحتملها الشبكة والمواضع

التي تشكل عنق الزجاجة. أخيراً سنقوم بمقارنة النتائج التي تحصلنا عليها باستخدام المحاكي الرياضي بنتائج حصلنا عليها باستخدام برنامج المحاكاة التجاري OPNET.

CHAPTER 1

INTRODUCTION

IP networks were designed to support non-real time applications which are characterized by their burst traffic and high bandwidth demand at burst time but they are not highly sensitive to delay. On the other hand, real time application such as VoIP requires timely packet delivery with low delay, jitter and packet loss. Integration of voice and data onto single network is becoming a priority for many network operators. To achieve this goal, IP network must be enhanced with mechanism that ensure the quality of service required to carry real time traffic such as voice. VoIP is a real time service, unlike the classical Internet applications, such as browsing, file transfer or mail application. Most of these applications use TCP with its control mechanisms. VoIP is usually implemented using UDP which can provide better real time responsiveness and lower overheads.

When deploying a new network service such as VoIP over existing network, many network architects, managers, planners, designers, and engineers are faced with common strategic, and sometimes challenging, questions. What are the QoS requirements for VoIP? How will the new VoIP load impact the QoS for currently running network services and applications? Will my existing network support VoIP and satisfy the standardized QoS requirements? If so, how many VoIP calls can the network support before upgrading prematurely any part of the existing network hardware?

This thesis work will start with an overview of VoIP and its various protocols. We will explain VoIP end to end delay and traffic flow then we will highlight the advantages of VoIP. After that we will mention the various types of existing network design tools giving examples of every type and explaining them.

The rest of the paper will explain in details the tool we implemented and the way it is used. We will also explain the code of the program. Finally, we will run a simulation on an existing network and compare the result with results obtained using a simulation tool on the same network topology.

1.1 Voice over IP

VoIP (Voice over Internet Protocol) is a collection of technologies, protocols, and devices that provide telephony services over an IP network. This means that instead of using the traditional PSTN, an IP network is used to carry digitized voice in discrete packets based on the H.323 specification, the specification for transmitting multimedia (voice, video, and data) across a packet-based network, which does not provide quality of service (QoS). Ultimately, VoIP strives to provide the efficiency of a packet-switching network while rivaling the voice quality of a circuit-switched network.

1.2 VoIP Protocols

Although many aspects of IP Telephony remain unstandardized, some standards are now beginning to emerge. It is important when considering a VoIP telephony solution

that, wherever possible, the emergent standards are supported to facilitate interoperability with other existing products and future releases. There are many protocols in existence but the main ones are discussed below.

H.323 is a family of telephony-based protocols, developed and is being maintained by the International Telecommunications Union-Telecommunications (ITU-T), for the transmission of multimedia traffic across packet based networks. The H.323 suite primarily consists of H.225, the call signaling protocol, H.235, the security protocol (including authentication, integrity, privacy, and non-repudiation), and H.245, the capability exchange protocol. In essence, H.323 is an application-level protocol suite and thus can be used on top of any packet-based network transport, such as TCP/IP, to provide real-time multimedia communication.

Session Initiation Protocol (SIP) defined by the Internet Engineering Task Force (IETF), is a lightweight, application-layer control (signaling) protocol specifically designed for the Internet for creating, modifying, and terminating sessions with one or more participants. SIP also includes support for transport name mapping and redirection services. Normally, SIP is used in conjunction with H.323 to better support real-time communication over the Internet.

G.723.1 defines how an audio signal with a bandwidth of 3.4 KHz should be encoded for transmission at data rates of 5.3Kbps and 6.4Kbps. G.723.1 requires a very low transmission rate and delivers near carrier class quality. The VoIP Forum has chosen this encoding technique as the baseline Codec for low bit rate IP Telephony. The ITU standardized PCM (Pulse Code Modulation) as G.711. This allows carrier class quality audio signals to be encoded for transmission at data rates of 56Kbps or 64Kbps.

G.711 uses A-Law or Mu-Law for amplitude compression and is the baseline requirement for most ITU multimedia communications standards. Real-time Transport Protocol (RTP) is also used to provide end-to-end delivery services for data with real-time characteristics. These services include payload type identification, sequence numbering, time stamping and delay monitoring. RTP works on top of UDP to take advantage of its services (e.g. multiplexing and checksums). Another supporting protocol is RTCP, Real-Time Control Protocol. RTCP's main function is to provide feedback on the quality of the data distribution, giving an insight on the performance and behavior of the media stream. This is achieved by the periodic transmission of control packets to all participants in the session.

Resource Reservation Protocol (RSVP) is the protocol, which supports the reservation of resources across an IP network. RSVP can be used to indicate the nature of the packet streams that a node is prepared to receive.

1.3 VoIP Characteristics

The overall technology requirements of an IP telephony solution can be split into four categories: signaling, encoding, transport and gateway control.

The purpose of the signaling protocol is to create and manage connections between endpoints, as well as the calls themselves. Signaling is required to determine the status of the called party, available or busy, and to establish and terminate the call.

Next, when the conversation starts, the analog signal produced by the human voice needs to be encoded in a digital format suitable for transmission across an IP

network. An efficient voice encoding and decoding mechanism is vital for using the packet-switched technology. The purpose of a voice coder (vocoder) – also referred to as a codec (coder/decoder) – is to use the analog signal (human speech) and transform and compress it into digital data. A voice coder samples the analogue signal at regular intervals (125 microseconds is a typical value), and converts the measured analogue value into a numeric representation (known as quantizing). The resultant output comprises discrete blocks of information sent at regular intervals. The compressed data is put into IP packets (a process called packetization) and these packets are routed over the network to the intended destination. The IP network itself must ensure that the real-time conversation is transported across the available media in a manner that produces acceptable voice quality. Finally, it may be necessary for the IP telephony system to be converted by a gateway to another format, either for interoperation with a different IP-based multimedia scheme or because the call is being placed onto the PSTN.

The high-level processing flow of a two-party, VoIP telephone call proceeds as follows:

1. The user picks up the handset; this signals an off-hook condition to the signaling application part of VoIP in the router.

2. The session application part of VoIP issues a dial tone and waits for the user to dial a telephone number.

3. The user dials the telephone number; those numbers are accumulated and stored by the session application.

4. After enough digits are accumulated to match a configured destination pattern, the call initiation request is forwarded to the gatekeeper. The gatekeeper examines its

local registration information and bandwidth availability and requirements. If all conditions are met, the telephone number is mapped to an IP address if the destination is internal to the network or the request is forwarded to a PBX (Private Branch Exchange) through a gateway that is responsible for completing the call to the configured destination pattern.

5. The session application then establishes a transmission and a reception channel for each direction over the IP network. If the call is being handled by a PBX, the PBX forwards the call to the destination telephone.

6. The coder-decoder compression schemes (Codecs) are enabled for both ends of the connection and the conversation proceeds using Real-Time Transport Protocol/User Datagram Protocol/Internet Protocol (RTP/UDP/ IP) as the protocol stack.

7. Any call-progress indications (or other signals that can be carried inband) are cut through the voice path as soon as end-to-end audio channel is established. Signaling that can be detected by the voice ports is also trapped by the session application at either end of the connection and carried over the IP network encapsulated in Real-Time Transport Control Protocol (RTCP).

8. When either end of the call hangs up, it sends a call termination notification to the other end (or the PBX), which in turn acknowledges the notification and resources are then released. In some implementations where the gatekeeper should be notified about call teardown (such as for billing purposes), each end further sends a termination notification message to the gatekeeper informing it that bandwidth is being released.

9. Each end becomes idle, waiting for the next off-hook condition to trigger another call setup.

Figure 1.1 shows a typical VoIP configuration, where the following components can be identified:

1. VoIP Gateway: Gateways are responsible for converting packet-based audio formats into protocols understandable by PSTN systems (mainly Systems Signaling 7, SS7, the set of standards for signaling used for call setup, teardown, and maintenance in the PSTN), and vice versa. They may also provide conversion between different codecs (transcoding) if a codec other than G.711 is used by the IP network.

2. Gatekeeper: A gatekeeper runs the H.323/SIP combination to coordinate call setup and call termination and perform address mapping. It can also be configured to be responsible for bandwidth control, authentication, authorization, and accounting.

3. VoIP Terminals (Clients): These are either multimedia PCs with VoIP software installed (IP SoftPhones) or VoIP-enabled phones (IP Phones).

4. MCU: Multi-point Control Units are required only if sessions with more than two endpoints are to be established.

5. PBX (or PBAX): A Private (Automatic) Branch Exchange is a private telephone network controller owned by an enterprise, users of which share a certain number of outside lines for making telephone calls external to the PBX. Additionally, a PBX provides some services internal to the enterprise, such as call transfer and forwarding.

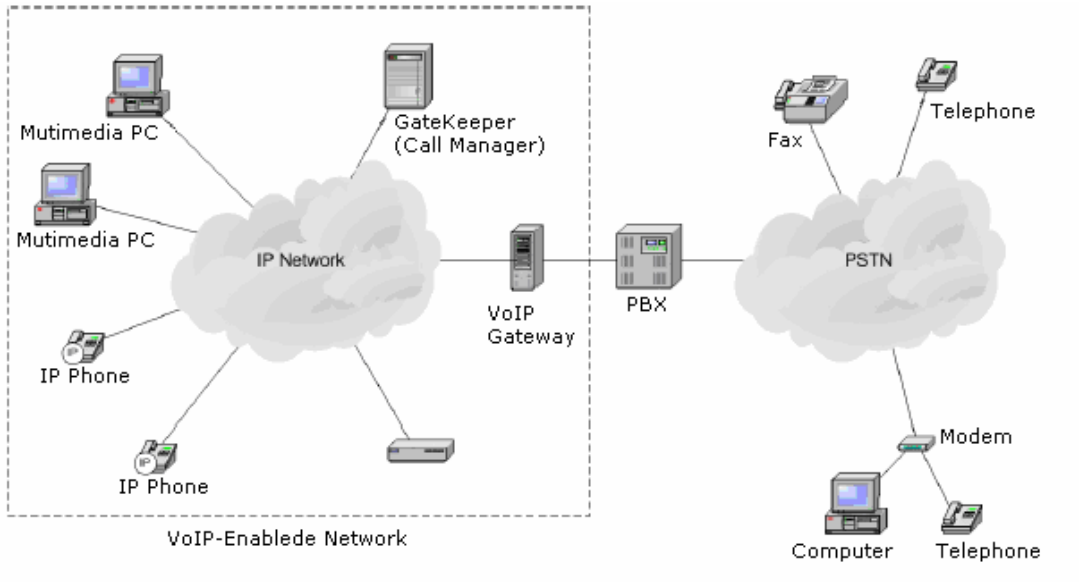

Figure 1.1 Typical VoIP network

1.4 VoIP End-To-End Delay

For introducing a new network service such as VoIP, one has to characterize first the nature of its traffic, QoS requirements, and any additional components or devices. For simplicity, we assume a point-to-point conversation for all VoIP calls with no call conferencing. For deploying VoIP, a gatekeeper or CallManager server has to be added to the network. The gatekeeper handles signaling for establishing, terminating, and authorizing connections of all VoIP calls. Also a VoIP gateway server is required to handle external calls. A VoIP gateway is responsible for converting VoIP calls to and from the Public Switched Telephone Network (PSTN). As an engineering and design issue, the placement of these servers in the network becomes crucial. Other hardware requirements include a VoIP client terminal, which can be a separate VoIP device.

Figure 1.2 [SAL06] identifies the end-to-end VoIP components from sender to receiver. The first component is the encoder which periodically samples the original voice signal and assigns a fixed number of bits to each sample, creating a constant bit rate stream. The traditional sample-based encoder G.711 uses Pulse Code Modulation (PCM) to generate 8-bit samples every 0.125 ms, leading to a data rate of 64 kbps. The packetizer follows the encoder and encapsulates a certain number of speech samples into packets and adds the RTP, UDP, IP, and Ethernet headers. The voice packets travel through the data network. An important component at the receiving end is the playback buffer whose purpose is to absorb variations or jitter in delay and provide a smooth playout. Then packets are delivered to the depacketizer and eventually to the decoder which reconstructs the original voice signal.

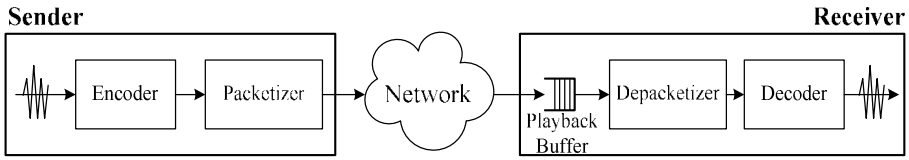

Figure 1.2 VoIP end-to-end components

Figure 1.2 [SAL06] illustrates the sources of delay for a typical voice packet. G.711 imposes a maximum total one-way packet delay of 150ms end-to-end for VoIP applications. We can break this delay down into at least three different contributing components, which are as follows (i) encoding, compression, and packetization delay at the sender (ii) propagation, transmission and queuing delay in the network and (iii) buffering, decompression, depacketization, decoding, and playback delay at the receiver.

1.5 VoIP Traffic Flow and Call Distribution

Knowing the current telephone call usage or volume of the enterprise is an important step for a successful VoIP deployment. Before embarking on further analysis or planning phases for a VoIP deployment, collecting statistics about the present call volume and profiles is essential. Sources of such information are organization's PBX, telephone records and bills. Key characteristics of existing calls can include the number of calls, number of concurrent calls, time, duration, etc. Also call location and flow is important. This will include the percentage of calls made internally or externally. Call distribution must include percentage of calls within and outside of a floor, building, department, or organization. As a good capacity planning measure, it is recommended to base the VoIP call distribution on the busy hour traffic of phone calls for the busiest day of a week or a month. This will ensure support of the calls at all times with high QoS for all VoIP calls.

1.6 Advantages of VoIP

It is evident from the discussion above that the characteristics of voice do not match those of IP networks. Voice speaks to circuit-switching, not to packet switching and routing. Voice speaks to channelized TDM (Time Division Multiplexing), not to unchannelized statistically multiplexed circuits. Voice speaks to committed bandwidth from call setup to call teardown, not to bandwidth that's available whenever it happens to be available. The nature of voice gives rise to some serious issues when transported

across an IP network. It is inevitable to try to resolve these issues if this technology is to be practical. There are a number of strong incentives to devise solutions to these issues, some of which are given below:

1. Reduced initial installation costs. This is achieved by reusing the devices and wiring already setup for the data network.

2. More efficient use of network capacity. The available bandwidth is used only as needed. Also, the available bandwidth is maximized because there is no reserved connections across the network.

3. Integration. A unified architecture that provides multiple functionalities to end-users helps streamline operation and management. Universal use of the IP protocols for all applications holds out the promise of both reduced complexity and more flexibility. Related facilities such as directory services and security services may be more easily shared.

4. Simplification. An integrated infrastructure that supports all forms of communication allows more standardization and reduces the total equipment complement. This combined infrastructure can support dynamic bandwidth optimization and a fault tolerant design.

5. Call cost reduction. Reducing long distance telephone costs is always a popular topic and provides a good reason for introducing VoIP. Users can bypass long-distance carriers and their per-minute usage rates and run their voice traffic over the Internet for a flat monthly Internet-access fee.

6. Advanced applications. Even though basic telephony and facsimile are the initial applications for VoIP, the longer term benefits are expected to be derived from

multimedia and multi-service applications, e.g. voice-enabled e-commerce applications. Needless to say, voice is an integral part of conferencing systems that may also include shared screens, white boarding, etc. Combining voice and data features into new applications will provide the greatest returns over the longer term.

VoIP deployments are best done in stages, starting with a few sites or locations where success is likely and where the ROI will be high. The following are some good candidates for early deployment:

1. **New branch offices** - A new branch office or a new wing of a building still in the planning stage is a good place to consider an early VoIP implementation.
2. **Sites with a planned data network upgrade** - A network upgrade means changing the network's architecture and installing devices, such as IP routers and switches, with a much higher capacity. Include VoIP requirements in the planning, and make sure the new devices support the VoIP characteristics that will be used.
3. **Sites with an expiring PBX lease or service contract** - When renegotiating current PSTN contracts, it's a good time to consult a secondary set of potential providers and to consider converting a portion of the network to VoIP.
4. **Remote users** - VoIP can be an excellent way to extend telephone service to remote workers, such as those providing help desk support from their home offices.

5. **Converging technologies after a company merger or acquisition** - Mergers or acquisitions often bring together different network technologies and phone systems. In these situations, it often makes sense to begin the process of convergence.

CHAPTER 2

LITERATURE REVIEW

This section reviews some of the recent work on estimating the network readiness for VoIP. Data network engineers usually over engineer their design to guarantee that the network is more than ready for any type of application they might need. However, nowadays most organizations are trying to cut cost as much as possible. Therefore, tools to predict if an existing network was ready for VoIP were desperately needed.

2.1 General Purpose Network Design Tools

Many new network applications are introduced every year. These applications require high bandwidth and availability. At the same time networks are becoming more and more complex. Due to that, it becomes very difficult for network engineers to predict if the network is ready for such applications. Many network tools have been developed to assist network engineers in making the right choices.

Tools analyze (visually, mathematically, or by simulation) what is happening in the network and predict future behavior. Nearly every general-purpose network design tool works the same way. The designer either uses a drag-and-drop graphical editor to create or modify a network topology or imports the topology directly from a network management tool like HP OpenView. Most tools have extensive libraries of device and

link models. Therefore, building an Ethernet local area network (LAN) or IP-routed backbone network with the help of these tools is straightforward.

2.1.1 Users of Network Design Tools

The ideal network design tool can mean different things to different people. A sales manager sells network hardware or provides network services and wants to show customers reasonably accurate representations of how a product, service, or technology will improve their network and support their business case. Such a person needs an intuitive tool that runs on a laptop computer and that can be master in just a few days with fast execution speed and extensive presentation features.

A network manager or engineer operates the network, troubleshoot and solve performance problems, and ensures compliance with service-level agreements. And tries to gauge how proposed changes will affect cost, performance, capacity, and availability before the changes go through. Changes typically include introducing new routing protocols, adding new devices and links, supporting new applications such as those for enterprise resource planning (ERP) or e-mail, upgrading servers, and changing service-level agreements. Topologies and traffic data might need to be imported from other tools. The set of alternatives is usually enormous, so scenarios need to be evaluated in tens of minutes rather than hours.

A network designer specifies and builds new networks or overhaul existing ones to meet performance requirements without overbuilding. He also identifies potential bottlenecks and overloads. Designers need an extensive library of link technologies,

devices, architectures, and protocols to build or upgrade the network, and need tools to accurately predict its performance.

Researchers test the effects of new or modified protocols, devices, architectures, component designs, and traffic models in the lab or on the workbench to reduce development costs and risks. Therefore they need complete control of simulated behavior at the programming language level, and want the language to provide a rich set of special-purpose modeling functions. Researchers typically simulate discrete events, and mimicking actual behavior which can take billions of events. Researchers prefer accuracy over speed [ARN00].

We can categorize the general purpose network design tools into 3 categories as shown in Figure 2.1.

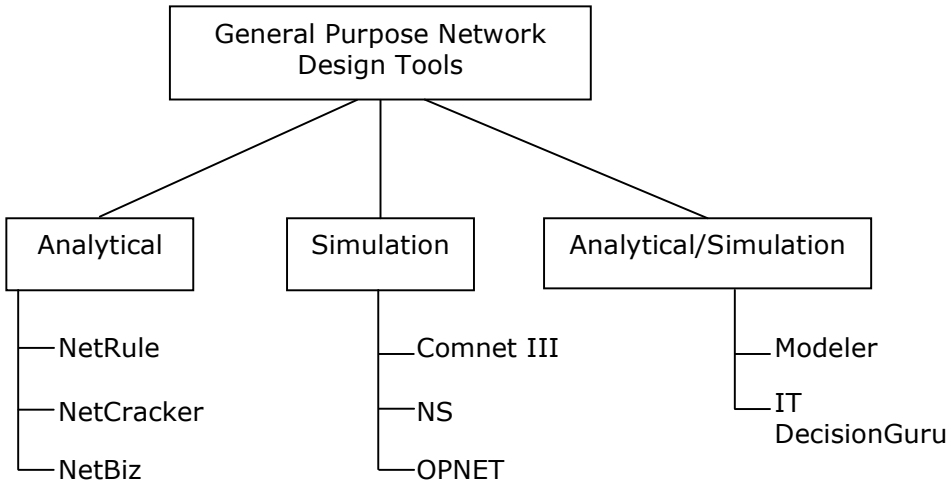

Figure 2.1 General purpose network design tools

2.1.2 Analytical Network Tools

NetCracker Technology has two network design products: NetCracker Designer and NetCracker Professional. Designer is a low-cost, low-end tool for designing, verifying, documenting, and visually analyzing networks via animation. It can import topologies from Visio and OpenView, and automatically confirm device connectivity and compatibility. NetCracker Designer supports the most common network and routing protocols and media. Its most interesting feature is a database of more than 25,000 network devices. Each device has dozens of attributes (supported media, protocols, port configurations, latency, bandwidth, price, etc) [NETC].

Analytical Engine's NetRule 2.0 is a basic, entry-level product. Analytical Engine considers NetRule to be a modeling package and notes that simulation is just one of many techniques for modeling a network's performance. The company's intent is to provide an easy-to-use product with a user friendly interface. NetRule provides various computer models, such as a file server, generic server, Internet server and 200-MHz Pentium PC. Company representatives say it's the user's responsibility to properly tune each device to reflect its true performance characteristics. Because NetRule is implemented in Java, it runs on a variety of platforms without a problem.

NetRule’s developers believe that simulation is too complex, too slow, and too expensive for practical net-work planning. The analytical engine uses steady-state queuing theory formulas, mathematical modeling, search algorithms, component level simulation, and rule-based inference tools. Advertisements claim evaluation times rarely exceed 10 seconds, even for large networks. NetRule can model and stress test very large

networks (1,000 to 10,000 nodes) because runtimes don't depend on the number of packets transiting the network [ANAL05].

NetBiz is for sales, professional services, and field services staff. It provides a rules-based environment for network design that one can customize for his organization. It has excellent presentation and configuration features and requires a minimal learning time. It targets companies selling networking services or equipment [ARN00].

2.1.3 Simulation Network Tools

Simulation tools (discrete-event) create an extremely detailed packet by packet model of predicted network activity but require extensive calculations to simulate a very brief period. Typical simulations running on a powerful server can take several hours or even days depending on the simulation period and the complexity of the network.

OPNET and NS are general purpose simulation tools. They have well defined VoIP profiles. Therefore, they will be discussed in more details in the coming section regarding VoIP Design tools.

Compuware (http://www.compuware.com) acquired several excellent tools from CACI International in 1999. Two of them are mature products with large installed bases. Comnet III is a high-end design and discrete-event simulation tool. Designers can create hierarchical network models using a drag-and-drop tool palette, or they can import topology and traffic data from several network management tools. Tightly integrated with other Compuware tools, Comnet III has an extensive library of devices and protocols. It

also has optional add-on modules for circuit switched traffic (voice, video, and satellite), distributed applications, and mobile or wireless networks [ARN00].

2.1.4 Analytical/Simulation Network Tools

Modeler is a more expensive, high-end product used mostly by network R&D engineers. It can precisely model protocols, devices, and behaviors using a finite-state-machine paradigm, C/C++ language features, and about 400 special-purpose modeling functions. Modeler has optional add-on modules for radio and satellite modeling, multivendor import, and service-level prediction.

Originally a discrete-event simulator, Modeler now supports hybrid simulations, which combine discrete-event simulation and analytical modeling. It can also run a simulation in parallel over several CPUs. Both hybrid and parallel simulations can significantly reduce simulation runtimes.

IT DecisionGuru uses a hybrid simulation technology combining simulation and analytical techniques that lets network managers control the amount of detail provided. Turn up the level of detail, and run-time increases proportionally. Scale it back, and run-time is shortened. A feature unique to IT DecisionGuru - and one that makes it particularly appropriate for organizations with enterprise resource planning programs such as SAP R/3, PeopleSoft and Baan - is MIL3's Simulation Methodology for Application Response Time Engineering (SMARTE).

Instead of depending on canned application models, SMARTE allows profiling the exact behavior of an application by capturing packet traces, adding background traffic levels and then investigating what-if scenarios. SMARTE give the user the opportunity to tailor a simulation to reflect his real network, rather than a network that the creators of the program envisioned [ARN00].

Figure 2.2 shows the intended audience for some general purpose network design tools and the level of skills required.

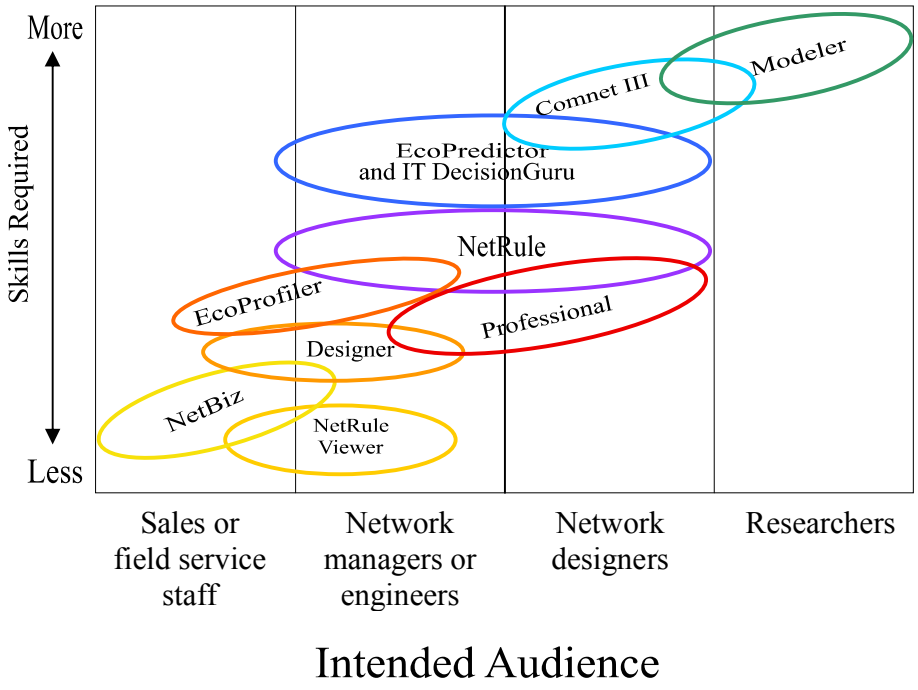

Figure 2.2 Intended Audience for some general purpose network design tools

2.2 VoIP Design Tools

Many tools have been developed to measure and predict if a network is ready to carry VoIP. These tools can be categorized into two categories as shown in Figure 2.3.

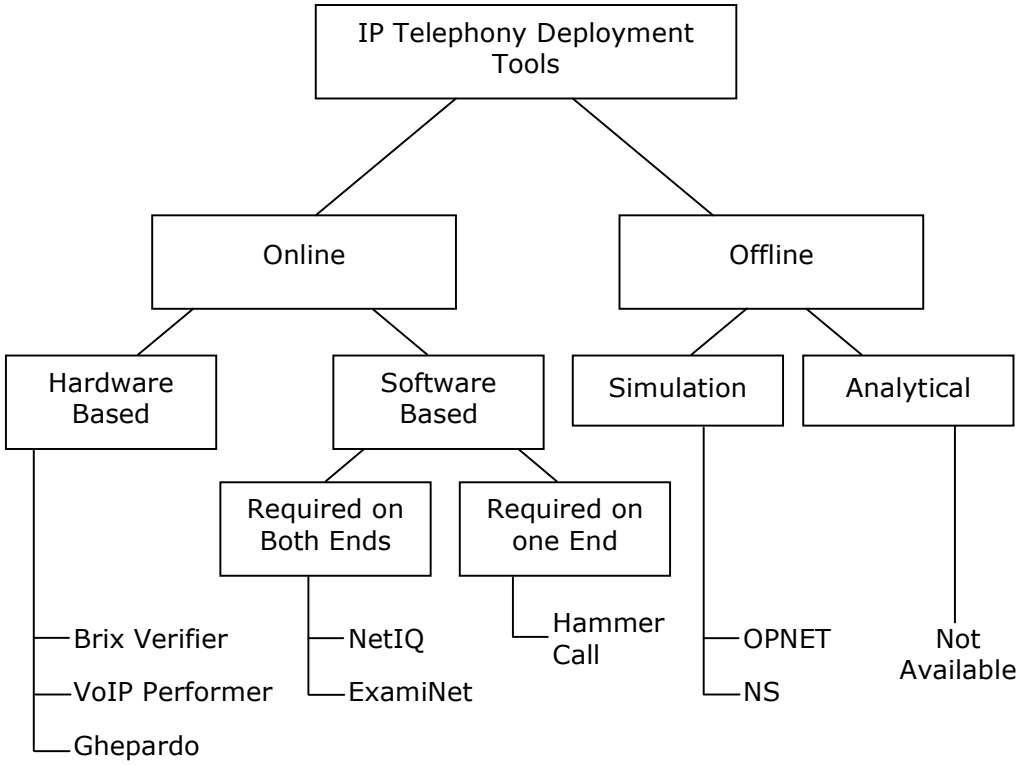

Figure 2.3 IP Telephone Deployment Tools

2.2.1 Tools Performed Online

Some of the tools available need to do their measurements on an existing network. The drawback of such tools is that they do not suggest the type of equipment needed when trying to build a network from scratch. They will only show weather or not the existing network is capable of carrying VoIP and some might identify existing bottlenecks. These tools can be categorized into hardware based and software based tools.

2.2.1.1 Hardware-Based Tools

Hardware based tools are those that require network appliances to be installed somewhere on the network. These tools usually give accurate readings but are expensive to buy and are difficult to use and install. They usually come with software used to configure and monitor the hardware. In this section, examples of hardware-based tools shown in Figure 2.3 will be introduced, along with the functionality of each one.

The Brix Verifier family consists of three hardware appliances which are Brix 2500 verifier, Brix 1000 verifier and Brix 100 verifier. It also has a family of three software tools which are Brix Verifier agents, BrixMon and BrixWorx.

Brix 2500 differs from the other models by having more processing powers and memory. It is usually connected to the core of the network and at key locations. On the other hand Brix 1000 is usually installed at smaller datacenters and has less power than the 2500. Brix 100 is installed at the end user location and alternatively the Brix Verifier Agent can be used as a software solution for the end user. BrixMon and BrixWorx are used for provisioning and managing the verifiers (Brix 2500, 1000 and 100). The difference between them is that the prior is the service provider edition and the latter is the enterprise edition.

Another hardware-based tool is the VoIP Performer. The VoIP Performer family consists of five hardware based appliances which are 323SIM VoIP Caller Generator, SIPSim, MediaPro, MegaSIP and QPro. It also consists of two software components which are Capture, and MasterScript.

The 323Sim VoIP Call Generator is capable of emulating the functionality of an H.323 terminal. Several H.323 simulators can be deployed to increase call volume to thousands of simultaneous calls, with each 323Sim generating 2,000 calls at a rate of over 50,000 calls per hour. The SIPSim is a session initiation protocol simulator that generates high-volume, SIP-based traffic for VoIP systems testing. The major additional feature in MegaSIP that differs from the SIPSim is its ability to emulate the stress levels of real-world network conditions. The MediaPro on the other hand, is a real-time VoIP monitor. It analyzes media and signaling data generated from H.323/Megaco/MGCP/SIP protocols and provides voice quality measurements. The QPro is a speech quality performance measurement tool that allows users to objectively measure, analyze and predict end-to-end speech quality over any network using a PSTN interface. Capture and MasterScript are tools provided by VoIP performer. The prior provides protocol decoding up to the application layer and supporting filters and analysis of the captured data. The latter is a tool for preparing scripts that control different performer entities [RAD05].

Ghepardo is also a hardware-based VoIP tool. It does Call Simulation, Traffic Generation, Voice patterns generation and QoS measurement [SUN05].

2.2.1.2 Software-Based Tools

Software based tools are those that do not require hardware appliances to be installed. These tools can be categorized into two groups which are tools that need to be installed on both ends and tools that only need installation on one end. The tools that need to be installed at both ends have the drawback of being less flexible. For example if the user doesn't have physical access or the physical access is not easily reached, then the

tools that require installation on both ends are not usable. However, the tools that only need to be installed on one end usually do not measure round trip end to end delay and also have other limitations. NetIQ and ExamiNet are examples of tools that need installation on both ends. Empirix is an example of a tool that needs installation only on one end. In this section these three tools will be discussed in more details.

NetIQ (Vivinet Assessor) is a software that supports Windows platform servers and Windows/UNIX endpoint clients. It predicts call quality by calculating a Mean Opinion Score (MOS) based on the industry standard E-model specified in the ITU recommendation G.107. Vivinet Assessor also take into account additional network factors that can impact call quality, such as jitter and consecutive lost datagrams. Vivinet Assessor uses a patent-pending method, based on freely distributed software agents, for calculating one-way network delay. The one-way network delay is then combined with the delay introduced by packetization and defined jitter buffers to create a complete end-to-end delay measurement for the call. Vivinet Assessor also has the ability to simulate VoIP traffic. The start times of simulated calls can vary to realistically emulate call traffic patterns. Vivinet Assessor can support up to 2,500 calls per assessment. It also enables definition and selection of advanced QoS parameters, including support for DiffServ and 802.1p [VIVI05].

Another software-based tools is ExamiNet. ExamiNet's approach does not rely on network and call simulation models. ExamiNet use synthesized traffic injection and observe actual end-to-end QoS measurements, rather than relying on traffic simulation to predict the end-to-end QoS. In particular, ExamiNet seek to observe call behavior at actual peak network load, rather than attempting to simulate and predict network load.

ExamiNet consists of five main components, namely: topology discovery, network device monitoring, call generation and call quality monitoring, database, and visualization and analysis.

Network Topology Discovery has the ability to discover devices in a customer's network and their topology at both layer 3 and layer 2 using SNMP queries. Network Topology Discovery serves three important purposes. First, it identifies devices in the network to monitor. Second, it allows the endpoint selection process to intelligently choose endpoint pairs whose paths cover the entire network. Third, it identifies which network elements are in the path between endpoint pairs so that the analysis can associate call data between the endpoints and data collected from network devices.

The Network Device Monitoring component of ExamiNet collects network utilization and load statistics on discovered devices via SNMP MIBs. The database provides the list of devices (from the discovery phase) to be monitored. This component collects values for two types of MIB variables from the SNMP agents on discovered devices at regular intervals. The first type is device-specific MIB variables such as the total number of input datagrams received on all interfaces. The second type is interface-specific MIB variables such as the total number of octets received on an interface.

The call generation and monitoring component of ExamiNet injects voice traffic to the network while collecting call quality metrics and layer-3 path information. Call quality metrics are measured at the endpoints of each call, and measurements are preserved for both directions of RTP traffic flow. The layer-3 path information, collected using traceroutes initiated by the endpoints during the synthesized call, is used to verify that the call is following the path predicted by the discovery phase based on router tables.

Voice traffic injection is carried out “around the clock” for several days, typically at least five business days. The objective is to ensure that the data collection occurs during time sensitive congestion that may occur in the network, and in particular to observe the network at daily and weekly peak, or “busy hour”, loads.

The ExamiNet Database architecture stores all the information collected from the network in a persistent store. The other components interact with the database. The Visualization and Analysis component provides a tabular and an interactive interface to the information stored in the database.

Hammer Call Analyzer (HCA) is another software-based tools. It has the ability to save capture packets to a capture file. While capturing filters can be applied specifying the type of packets and the source and destination. HCA also has the ability to view G.711 RTP streams as a wave form and playback recorded RTP streams. It also displays statistics of jitter and mean opinion score. Unlike the previous tools this tool only requires installation on one end. But a drawback of this is that it is not able to detect end to end delay of deploying VoIP [EMPR05].

2.2.2 Tools Performed Offline

The most popular simulation tool in the market is OPNET and NS. Although OPNET is not specific for VoIP, but it has VoIP profiles where VoIP attributes can be specified. It also gives useful VoIP output.

OPNET is an object oriented simulation environment that is modular, hierarchical and takes advantage of graphical potential of today's workstations. It is ideal for interpreting and synthesizing output data. It has a built-in Proto-C language support that helps it recognize almost any function and protocol. Opnet supports hierarchical modeling. It provides Network, Node, Process and Parameter editors that support the model level reuse. Models are reusable by another model at a higher layer. Network Editor defines the physical topology of the network by specifying the positions and interconnections of the network devices. Node models are specified as interconnected modules grouped into two sets of modules. First has predefined characteristics and built-in parameters like packet generators and the second one consist of highly programmable modules like processors and queues. Process models define the logic flow and behavior of queue modules and processor [OPNET05].

NS is an open source discrete-event network modeling software developed at Lawrence Berkeley Labs in 1989. It is considered as the de facto standard for research-oriented network simulation tool. It is interesting to know that NS development is supported by DARPA and the National Science Foundation (NSF). NS is hosted at the Information Sciences Institute at the University of Southern California's School of Engineering. NS is widely used in academic institutions and research and development (R& D) labs. Although it lacks the user friendliness of commercial products, it is good enough to do the job. The use of NS is not only restricted to simulation. One main feature of it is its ability to be used as an educational tool for network visualization. The network animator (NAM) aids networking students to visualize protocols and animate network traffic.

The current version is NS-II that is significantly different from NS-I. For backward compatibility, an API library is used for that [NET05].

Mathematical analysis is another offline approach to measure network readiness. As mentioned earlier there are many general purpose network tools that use mathematical analysis but there does not exist a tool that is specific for measuring VoIP network readiness. In this study, a tool will be developed that uses mathematical queuing analysis to measure a network readiness for VoIP deployment. Not only is this approach new but it also has many advantages against all the other approaches as mentioned in the coming section.

CHAPTER 3

PROBLEM DEFINITION

In the previous sections the most common available network tools that measure VoIP readiness were discussed. Some non VoIP specific network tools that use mathematical analysis were also discussed. But there are no tools available that measure VoIP network readiness using mathematical analysis. Therefore the goal of this master thesis is to design and implement such a tool. This work will be presented as a program written in C#.

The tool will take the call distribution and the background traffic as input. And it will identify bottle necks in the existing network. It will also determine the maximum number of calls and the average and maximum delay in along with the utilization of the link.

The main advantage of mathematical analysis when compared to simulation is the noticeable speed difference. For example running a 5 minute simulation could take hours or even days especially in a complex network but running it using mathematical analysis takes seconds no mater how complex the network is. A second important advantage is that a tool using mathematical analysis is easier to use and requires less training and the results are very similar to those obtained using simulation. A third advantage is that the tool will require much less machine resources (memory, CPU…) to run therefore it can

run on a normal workstation whereas the simulation tools need a powerful server and some tools use multiple servers.

Not all companies can afford simulation tools whereas analytical tools are much cheaper. Not only do analytical tools cut cost by not needing expensive servers to be installed on but they tend to be cheaper than simulation tools due to there simplicity. Analytical tools also require less training and less support which also reduces expenses they can also be used to assess the accuracy of other commercial tools.

CHAPTER 4

IP TELEPHONY SIMULATOR

The following section will be divided into two parts, the first part (6.1) is the users manual of the IP Telephony Simulator. This part will explain in general the program and what it does and how to use it. The second part (6.2) is the program code. This part will explain in more details how the program was written and the algorithms used along with the source code of the program.

4.1 The Users Interface

The IP Telephony Simulator program is written using Microsoft Visual Studio. The programming language used was C#. Therefore the program can run on any Windows XP SP2 workstation. If SP2 is not installed then .Net runtime needs to be installed which can be downloaded from Microsoft website. The program is light and can run on any machine with minimal recourses.

To run the program, Lithium.dll is required to be in the same directory as the IP Telephony Simulator. This DLL file is needed to draw that network diagram. We will explain the network diagram along with more details about the program in the following sections.

4.1.1 The Input

The program provides five main steps that the user should follow in order to provide the network topology. These steps must be followed in the correct order and the program will alert the user if a step is skipped.

4.1.1.1 Step 1 (Nodes & Floors)

When the user runs the program he will get a form as shown in Figure 4.1. In this form he is required to provide all the nodes of the network. These nodes include servers routers and switches. The servers are needed because they contribute to the background traffic, so when links are added to the servers in the following step the user can specify the traffic they are generating.

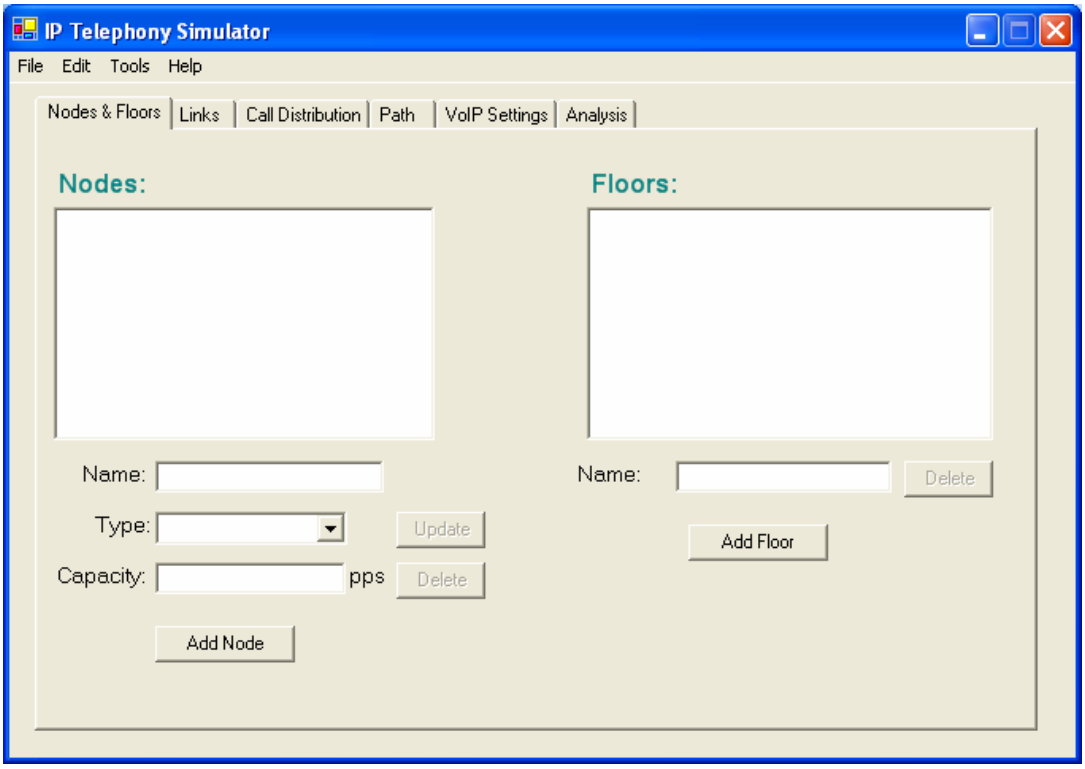

Figure 4.1 IP Telephony Simulator (Nodes & Floors)

In the case of switches and routers, the user has to enter their capacity in Packets per Second (pps). This is usually found in the data sheet specifications of the device. When the user provides the name type and capacity he has to click Add Node button for the node to be added and shown in the list.

The user can update the capacity of any node at any time, but he can only update the name in this step. If the user associates links to the node (as we will discuss in step 2) then he has to delete these links before modifying the name.

The program will not allow deletion of a node if there exists links associated with it. Therefore all the links associated with that node must be deleted first. Since we are still at step 1 we can update and delete nodes flexibly.

The program also takes floors as input. These floors are groups of users connected to the network. They could be departments, VLANS, buildings or simply different floors in a building. The program will refer to these groups as floors. The floors are needed to specify the call distribution and the path between the floors. This will be discussed in more details in step 3 and 4.

When a floor is deleted all of the call distribution assigned to it will be deleted and all of the paths that the floor uses to reach other floors will also be deleted. Therefore at any point in time the user can remove and add floors and see what effect that would have in the analysis.

4.1.1.2 Step 2 (Links)

After adding nodes and floors, the user can now add links between the nodes. By click the select box of Node 1 (as shown in Figure 4.2) a list of all the nodes will be shown, the user should choose one node. Similarly, by click Node 2 a list of all the nodes will also be shown and the user can choose which node he desires to connect to the first node. Then the user enters the capacity of the node in Mbps.

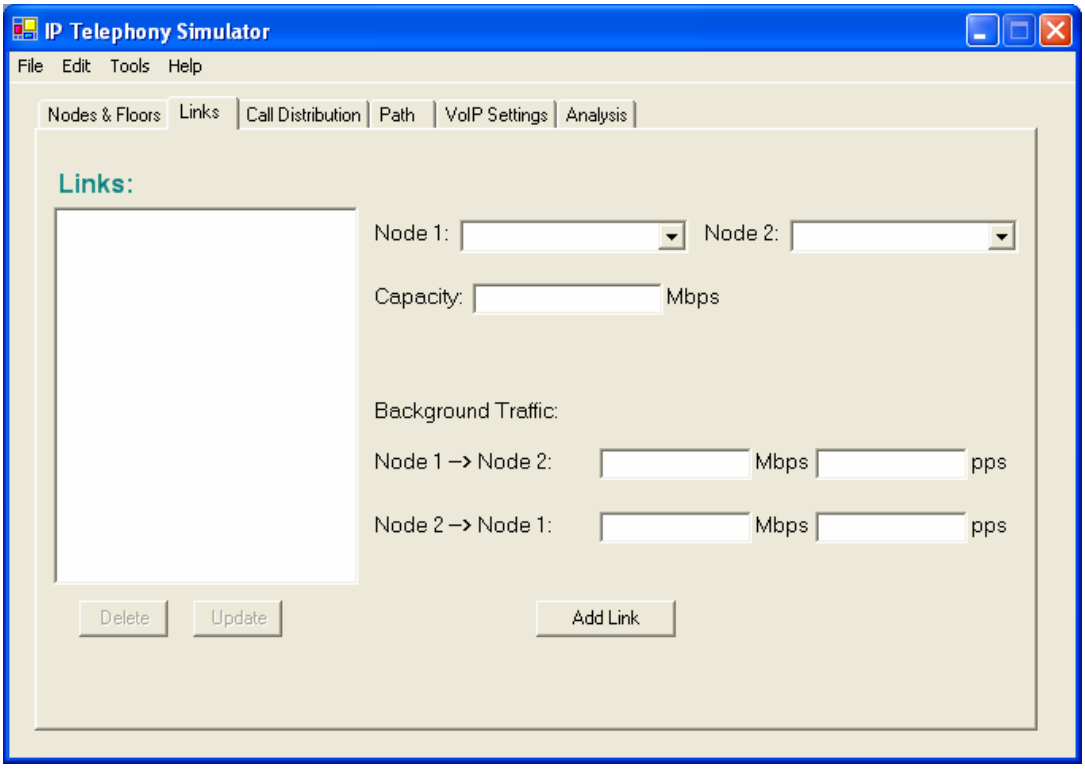

Figure 4.2 IP Telephony Simulator (Links)

After that, the user enters the background traffic of the link in both directions. This has to be done in Mbps and in pps. The reason for having both Mbps and pps is because the capacity of links is measured in Mbps and the capacity of the nodes is measured in pps.

Now the user can click the add link button and the link will be added to the list of links. At any point in time the user can update the links capacity or background traffic. He can also delete the link if it is not used in any path, otherwise it is required to move it from the path first.

4.1.1.3 Step 3 (Call Distribution)

In this step the user can put the call distribution for each flow. This step can only be done after adding the floors. As shown in Figure 4.3 there is a list of flows. This list is automatically generated as soon as the call distribution tab is clicked. The list consists of all the possible call flows among the floors.

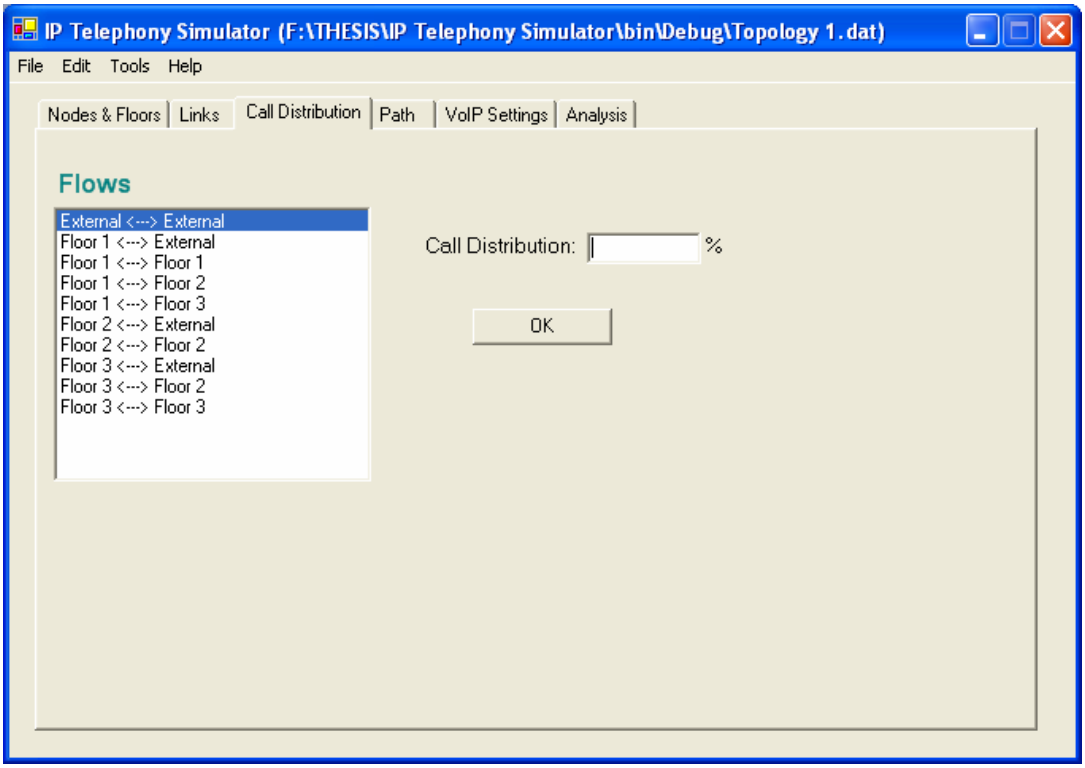

Figure 4.3 IP Telephony Simulator (Call Distribution)

By default the call distribution is divided equally between all floors. The user can select a flow and specify the percentage of calls he wants for that flow and then clicks the ok button.

4.1.1.4 Step 4 (Path)

Now that the user has entered the nodes, floors, links and call distribution he has to specify the path the call will take between the floors (for each flow). As shown in Figure 4.4 there is an automatically generated list of flows similar to the list in the Call distribution tab.

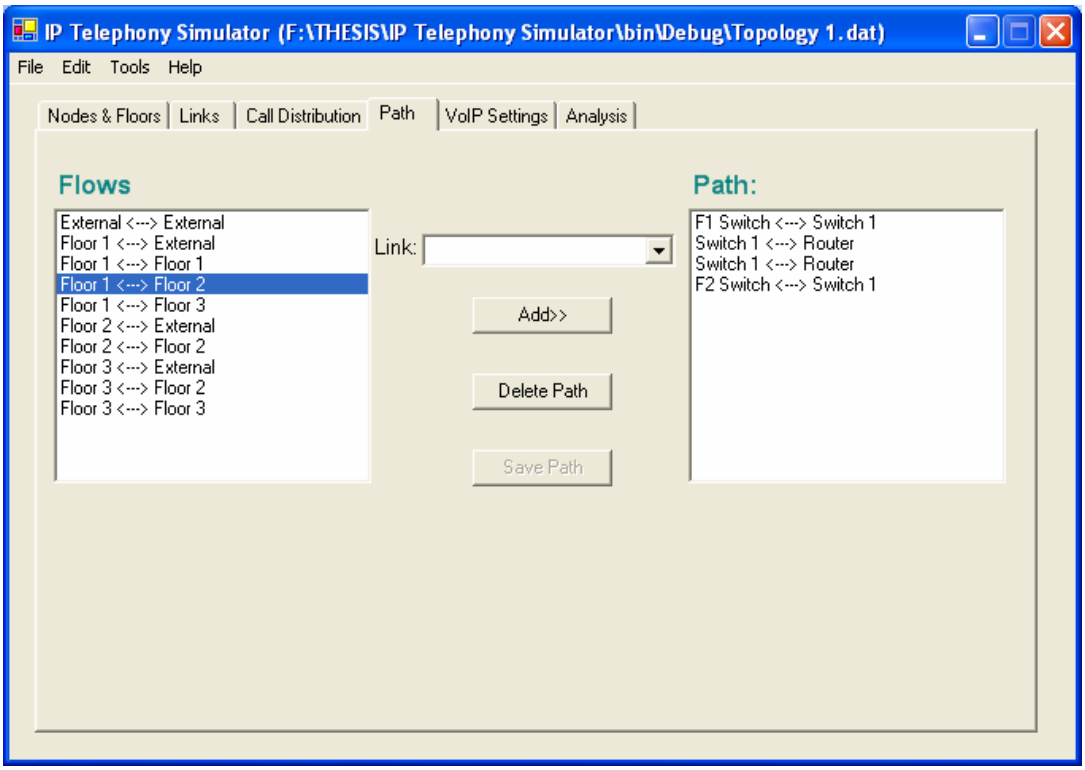

Figure 4.4 IP Telephony Simulator (Path)

The user first has to highlight one of the flows from the flow list then choose the links that the call will follow. The add button has to be clicked after selecting each link and once the path is complete the user has to click the save path button.

When the user starts adding links he will see a list of all the links in the links select box. After adding the first link, only valid links will be shown in the links select

box. This will minimize the possibility of selecting invalid links. For example, when the links Floor 1 Switch ⟷ Switch 1 was added only links coming out of or going into Floor 1 Switch or Switch 1 are shown in the links list.

The user can choose to delete a saved path at any time and insert a new path. The use can also leave a path empty for a certain flow if there are no calls on that path. In our example we are not concerned about calls between external ⟷ external therefore we did not specify a path for that flow.

Incase of internal flows for example Floor1⟷Floor 1 the path is internal to Floor 1 switch. In order for the user to specify such links the link list will show all the switches with the word [internal] next to them. When the selected flow is internal the user should choose the switch name with the word [internal] next to it. After doing that the user cannot add anymore links to that path and he can only save it or delete it.

4.1.1.5 Step 5 (VoIP Settings)

In this step the user puts the VoIP settings. He can specify the percentage of growth. This is the percentage of growth expected for the links and nodes. The links and nodes capacity will be reduced by this amount in the calculations. The default value of percentage of growth is set to 25%.

The other settings are set by default to the ITU G.711u standard. The user has the ability to change these settings if needed. The user can also restore to ITU G.711u at anytime by clicking on the Restore ITU G.711u Defaults button as shown in Figure 4.5.

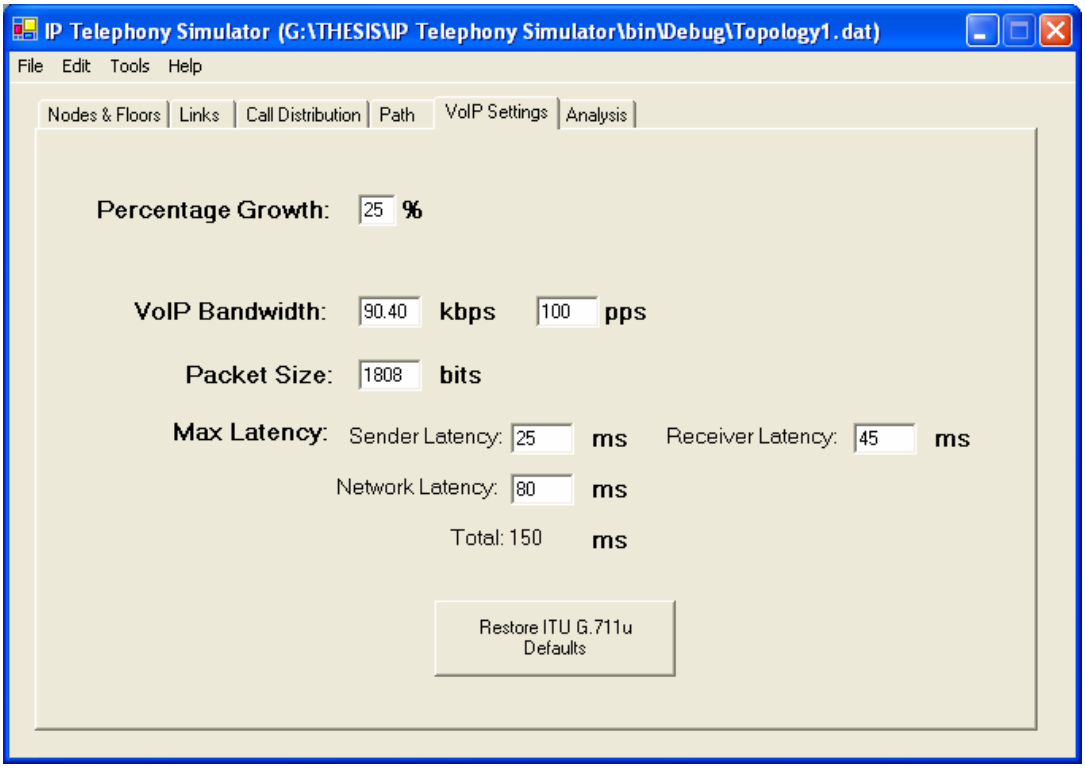

Figure 4.5 IP Telephony Simulator (VoIP Settings)

4.1.2 The Output

Finally, the user is now a click away from viewing the delay and throughput analysis reports. As soon as the user clicks on the analysis tab the following will be calculated:

1. Background traffic per node and per link
2. The flow per node and per link
3. Utilization (u_i) per node and per link
4. Maximum number of calls per node and per link
5. The reduced capacity of the nodes and links (μ)

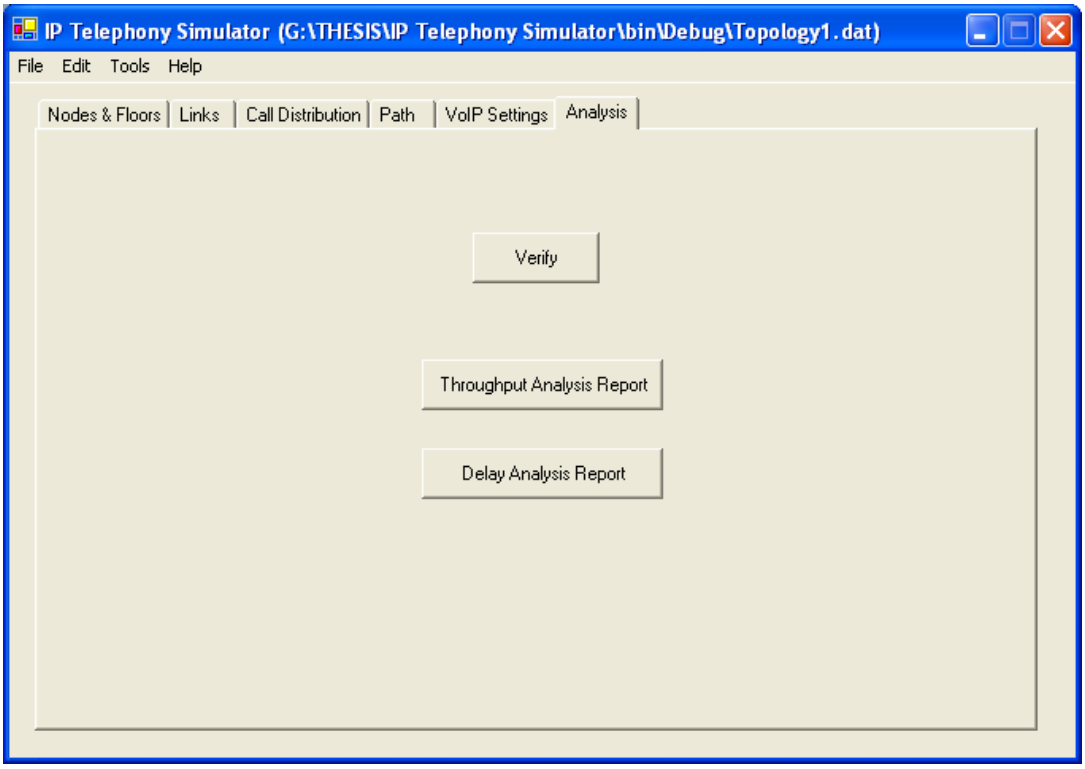

Figure 4.6 IP Telephony Simulator (Analysis)

When the user clicks on the Throughput Analysis Report button, a report as shown in Figure 4.7 will be generated showing the maximum number of VoIP calls supported per node followed by the maximum number of VoIP calls supported per link. The report also identifies the bottleneck of the network which is the node or link that supports the minimum number of calls.

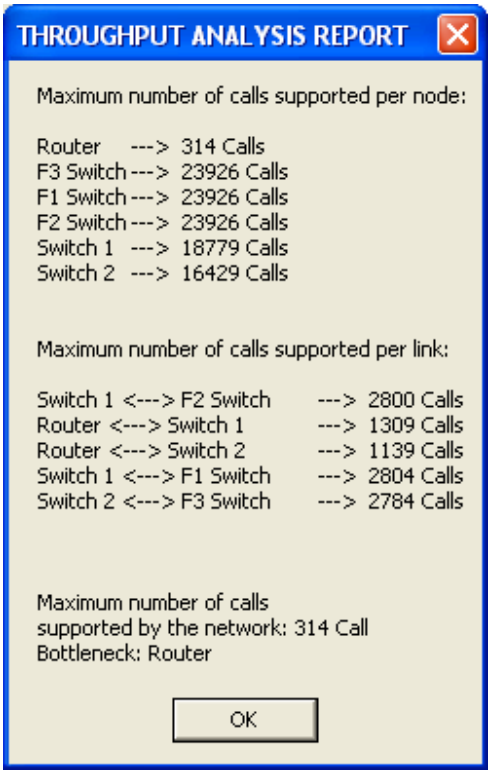

Figure 4.7 Throughput Analysis Report

When the user clicks the Delay Analysis Report button, a report as shown in Figure 4.8 will be generated. This report shows the maximum number of calls the network can support based on delay analysis. The delay for each node is calculated using MM1 and the delay for each link is calculated using MD1. After that the delays of all the nodes and links involved in each path are added up. Initially this is done for one call and if the total delay for all the paths doesn't exceed the maximum latency (150 ms based on ITU G7.11u) then the number of calls is incremented by one and so on. Once one of the paths has a total delay exceeding the maximum latency then that path is identified as the bottleneck and the report will show details of the delay of every node and link along that

path. The report will also show the total delay of that path and the number of calls it can support.

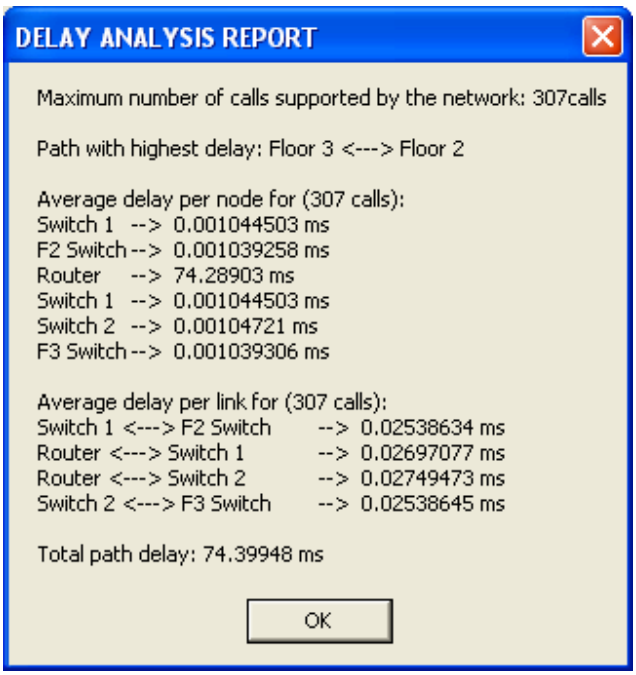

Figure 4.8 Delay Analysis Report

4.1.3 Other Features

To further assist the user the program can do some input validation. By selecting verify from the Tools menu, the program will verify that there is no isolated node. An isolated node is a switch or router that is not part of any path.

The tool also checks that all the paths go through valid links. This is done by making sure that when the user enters the links involved in the call path, he enters them sequentially. The following example is valid:

Floor 1 Switch ⟷ Switch 1

Switch 1 ⟷ Router

Router ⟷ Switch 2

Whereas an example such as the following is invalid and the program will not allow it because there is no link between Switch 1 and Router:

Floor 1 Switch ⟷ Switch 1

Router ⟷ Switch 2

The tool also checks that the total call distribution and the percentage of growth do not exceed 100% and that the nodes and links capacity are greater than zero. Furthermore the tool will not allow the user to input background traffic exceeding the link capacity.

IP Telephony Simulator will not allow the user to delete a node, link, floor or path unless it is done correctly. To delete a node the user must delete all the links connected to it first. To delete a link the user must delete all the paths involving that link first. If the user chooses to delete a floor the program will remove all call distributions involving that floor, therefore call distribution should be redone after deleting a floor to distribute the percentages to the other floors.

The program is also capable of automatically drawing the network diagram that the user inputted as shown in Figure 4.9. The user can view this diagram as soon as he starts adding links by selecting network diagram from the tools menu.

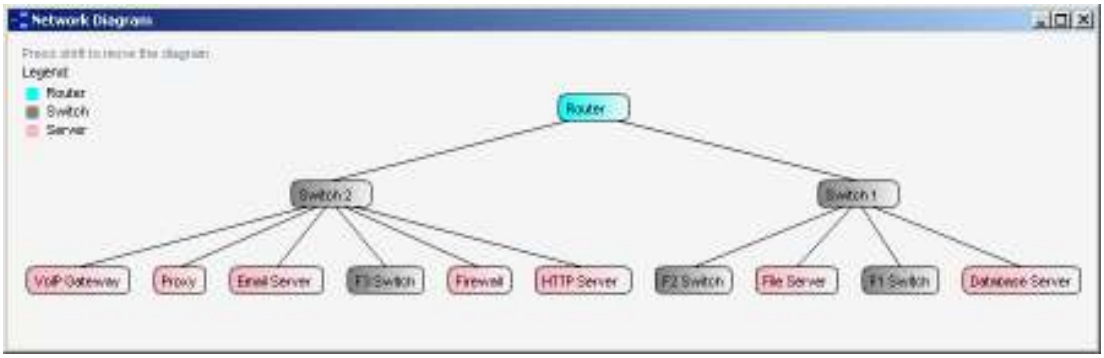

Figure 4.9 Network Diagram

4.2 The Program Code

In this section will explain how the program was written. This will be done by first explaining the different components of the program (6.2.1) and the overall structure of the code. Then a full walkthrough of the code will be explained (6.2.2) in details in the same order that a user would follow to use the program. All the algorithms used in the program will be explained in this section.

4.2.1 Program Main Components

The IPTelephonySimulator namespace is composed of five main components. This section will describe these components and the functionality each component

provides more details of the algorithms used in these components will be discussed in section 5.2.2.

4.2.1.1 ITSForm.cs

ITSForm.cs (IP Telephony Simulator form) is the main graphical user interface where the user can input his data and view the analysis reports. From this form the user can go anywhere in the program. The form is composed of multiple tab controls that the user follows in order from left to right to input the information about the network.

4.2.1.2 Nodes.cs

Another main file in IP Telephone Simulator is Nodes.cs. This file is composed of 5 classes that are the main data structures of the program as shown in the following table:

Class	Items	Description
Nodes	type	The node type (router, switch, server, floor)
	name	The name of the node
	capacity	The capacity of the node in pps
	BGTraffic	The background traffic imposed on the node in pps
	Mu	The reduced capacity of the node(after removing the percentage of growth from the capacity)
	lambda	(Lambda background + lambda voice) for the node in pps

	flow	The percentage of calls that will flow threw this node (based on call distribution)
	delay	The average delay of the node (MM1) in ms/p
	ui	The node average utilization from background traffic
	TPATotalCallsSupported	The total calls supported by the node based on throughput analysis
	DATotalCallsSupported	The total calls supported by the node based on delay analysis
Links	name	The name of the link. This name is automatically generated by the program after the user enters the two nodes that the link connects. (e.g. node 1 ⟷ node2)
	node 1	The node on one end of the link
	node 2	The node on the other end
	capacity	The capacity of the link in Mbps
	BGTraffic12	The background traffic on Mbps on the link from node 1 to node 2
		The background traffic in Mbps on the link from node 2 to node 1
	BGTraffic12pps	The background traffic on pps on the link from node 1 to node 2
	BGTraffic21pps	The background traffic in pps on the link from node 2 to node 1
	Mu	The reduced capacity of the link(after removing the percentage of growth from the capacity)

	lambda	(Lambda background + lambda voice) for the link in bps
	flow	The percentage of calls that will flow threw this links (based on call distribution)
	delay	The average delay of the link (MD1) in ms/p
	ui	The link average utilization from background traffic
	TPATotalCallsSupported	The total calls supported by the link based on throughput analysis
	DATotalCallsSupported	The total calls supported by the link based on delay analysis
CallDist	name	The name of the flow for which the call distribution is assigned to. These names are automatically generated by the program. (e.g. Floor 1⟷ Floor 2)
	percentage	the percentage of calls going through the flow
Paths	PathName	The names of the paths. These names are similar to the CallDist names
	delay	The total delay of the path
	PathArray	An array of links that build the path

Table 4-1 Program classes

These data structures are all the variables that the program uses. Whenever a new instance of a class is declared it is stored in a HashTable and when the user chooses to save the network topology all of the HashTables will be saved to a data file on the disk. A

HashTable object is collection of key-and-value pairs that are organized based on the hash code of the key. HashTables are a great way to track and use objects.

4.2.1.3 Engine.cs

The Engine.cs file contains a class called Engine. Most of the formulas used in the program are defined here. Lots of other calculations are done directly from the ITSForm class. The formula for calculating MM1 and MD1 are defined in this class along with the formula for calculating the maximum number of calls.

4.2.1.4 NetworkDiagram.cs & Lithium.dll

The NetworkDiagram.cs file along with the Lithium.dll file are responsible for drawing the network diagram. This network diagram is automatically generated based on the users input; the nodes and links HashTables are passed to the NetworkDiagram class where it will be drawn as a tree.

One of the nodes is elected as the root of the tree, then based on the links that are connected to that node children are added recursively. This is done until all the nodes are visited. The algorithm used will be explained in more details in the next section.

4.2.2 Code Walkthrough

This section will walkthrough the code of the program explaining the main algorithms used to build the IP Telephony Simulator. The code of the program is well

documented therefore only complex algorithms will be discussed in this section whereas the simple algorithms can be understood from the code documentation.

The walkthrough will be in the same order as the user would input his data. The network diagram algorithm and the methods for verifying the correctness of the input will also be explained in this section.

4.2.2.1 The Program structure

The IP Telephony Simulator uses four HashTables to store all the instances. Each HashTable has a reference key and a value.

The first HashTable is called NodesHT this HashTable stores all the nodes along with their attributes. Nodes include routers, switches, servers and floors. The attributes of the nodes were shown in Table 1. Not all the attributes are applicable for all the nodes for example floor does not have a capacity or background traffic it only has a name and a type in this case the attribute value is null.

Another main HashTable for the program is LinksHT this HashTable stores all the links and their attributes. The reference key for this HashTable is the name of link which is composed of the two end nodes with a bidirectional arrow in between. The attributes of the links are also shown in Table 1.

The third HashTable in the IP Telephony Simulator is the PathHT this HashTable stores the paths used for every IP Telephony call as inserted by the user in step 4. The reference key of this HashTable is the name of the flow which consists of the two floor end points along with a bidirectional arrow in between. These flows are automatically

generated by the program as soon as the user clicks on the call distribution tab in step 3. The program does that using the following algorithm:

```
INPUT: floors
OUTPUT: all possible flows

x=0
y=0

foreach (node in the nodes HashTable)

      if (the node type is floor) then
            Add it to the FloorsHT HashTable
      end if

end foreach

do
x=y
      do
            Add to Flow List:(FloorsHT[x] " <---> "  FloorsHT[y])
            x = x + 1
      while(x < Number Of Floors)

y = y + 1
while(y < NumberOfFloors)
```

Figure 4.10 Pseudo-code for computing flows

The first for loop simply selects all the floors from the nodes HashTable and copies them to a temporary HashTable called FloorsHT. After that, the nested do loop goes through the floors HashTable to make a full mesh between all the floors. The inner do loop keep incrementing x in every iteration while y is fixed. Therefore, after adding a new flow one end of the flow is replaced with the proceeding floor in the HashTable. This is done until x is greater than or equal to the number of floors. The outer loop increments y by one and sets x = y then the inner loop is done again. Now the inner loop is executed again for the second item of y but starting from y to avoid repetitions. The outer loop will loop until y is greater than or equal to the number of floors. As a result, a

list showing a full mesh of the floors will be generated automatically representing all the possible flows as shown in Figure 4.3.

The fourth HashTable used in IP Telephony Simulator is called CDHT (call distribution HashTable). Similarly to the path HashTable the key here is also the flow. And the attributes for each flow are the name of the flow and the percentage of calls expected to go through that flow.

4.2.2.2 Validation and Verification

In this section, we will explain all the algorithms in the program to validate and verify user input. As we will see, some of the validation is done before the user input; meaning that the program will prevent the user from inputting invalid information. Whereas other validation and verification is done when the user selects verify from the tools menu. We will start by explaining the first type followed by the second type. We will explain them in the same order they appear when the user inputs the data.

When the user adds a node the program will check that the user inputted a name and a capacity for the node before it adds the node. The program will also prevent the user from typing any none numeric value in the capacity field. When the user chooses the type to be a server the capacity field will be hidden to avoid the user from inputting unneeded data.

When the user chooses to delete a node the program also does some verifications to make sure that this node is not an end point of any link. This is achieved using the following algorithm.

```
INPUT: node to be deleted
OUTPUT: removing the node after validation


OKtoDelete = true

foreach(link in LinksHT)

      if(node is an endpoint of the link)
            OKtoDelete=false
            halt
      else
            OKtoDelete=true
      end if

end foreach

if(OKtoDelete is true) then

foreach(link in LinksHT)

      if(LinksHT contains NodeName +" [Internal]")) then
            Remove link from LinkHT
      end if

end foreach

Remove node from NodeHT

end if
```

Figure 4.11 Pseudo-code for removing a node

The first for loop in the previous algorithm goes through all the links in the links HashTable searching for the node that the user wants to delete. If the node name is found as an end point of any link then the node is flagged as OKtoDelete.

Recall that when a switch is added a link called SwitchName + [Internal] is added to the nodes HashTable to deal with internal calls; this link must be deleted when deleting the switch. The second part of the algorithm takes care of this where it checks if it is ok to delete the node; if so a For Loop will loop through all the links again looking for the

selected node to be deleted, concatenated with the string [Internal] in front of it. Once this string is found it is removed from the links HashTable, afterwards the node is removed from the nodes HashTable.

Updating a node is a combination of deleting a node then adding the updated node. Therefore the same algorithms used for adding and deleting a node are used for updating it.

When the user selects to delete a floor the program will remove that floor from the nodes HashTable, the call distribution HashTable and the path HashTable. This is done using the following algorithm.

```
INPUT: floor to be deleted
OUTPUT: removing the floor from the nodes HashTable, the call
distribution HashTable and the path HashTable

j=0

foreach(Flow in CDHT)

      if(floor is an endpoint of the Flow) then
            DeleteArray[j]= Flow
            j = j + 1
      end if

end foreach

for i ← 0 to j do
      remove DeleteArray[i] from CDHT
      remove DeleteArray[i] from PathHT
end for

remove node from NodeHT
```

Figure 4.12 Pseudo-code for removing a floor

The first For Loop in the previous algorithm goes through the call distribution HashTable (CDHT) looking for the floor the user wants to delete as an endpoint of a flow. If the floor is found then the flow will be added to an array called DeleteArray. Due

to the fact that we cannot alter a HashTable while it is in a Loop; we cannot directly remove the flow from the HashTable unless we exit the Loop. For that reason, the second For loop will loop sequentially on the CDHT and the PathHT removing all the items stored in the DeleteArray; and finally the floor will also be removed from the NodeHT. As a result of removing the floor from the three HashTables the analysis will be redone taking in consideration that the floor no longer exists.

When the user adds a link the program verifies that the link has not been previously added. The program also verifies that the two ends of the link are not the same meaning that the program will not allow a link that connects a node to itself. The program will also prevent the user from entering background traffic exceeding the capacity of the link.

Validation and verification is also done when the user wants to delete a link, in a similar way it was done when deleting a node. The program will first check if it is ok to delete the link by making sure it does not contribute to any path. This is done using the following algorithm.

```
INPUT: LinkName to be deleted
OUTPUT: removing the link

OKtoDelete=true

foreach (thisPath in PathHT)

      for i ← 0 to length of PathArray of thisPath do

            if (name of PathArray[i] of thisPath contains LinkName)
            then
                  OKtoDelete=false
                  halt
            end if

      end for
end foreach


if(OKtoDelete is true) then
      remove Link from LinksHT
end if
```

Figure 4.13 Pseudo-code for removing a link

As shown in the previous algorithm, the first For loop goes through the path HashTable and in every iteration it will go through the PathArray of the flow. As mentioned in Table 1 the PathArray is a sequence of links that form the path. If the link that the user wants to delete is found in the array, then the OKtoDelete will be false. If the link is not found in any PathArray of any flow then the OKtoDelete will remain true and the link will be removed from the LinksHT.

The algorithm for updating a link is similar to that of updating a node. Thus, it is a combination of deleting a link then adding the updated link.

An additional verification that is done before the user input, is the path verification. In order to prevent the user from inputting an invalid path the program will only show valid links in the drop down menu. It does that using the following algorithm.

```
INPUT: left and right end nodes of the link that was previously added
to the path
OUTPUT: a list of valid links to choose from to construct the path

N1 = left end of the link previously added to the path
N2 = right end of the link previously added to the path

foreach (L in LinksHT)

      if (L is not an internal link) then

      item1 = left end of L
      item2 = right end of L

      if((L contains N1 OR N2) AND (item1 AND item2 are not servers))
      then
            add L to the list of links to choose from
      end if

      end if

end foreach
```

Figure 4.14 Pseudo-code for constructing a valid path

The previous algorithm will only allow the user to input valid path. In order to insure that the path is continuous, each link should be followed by a link that has at least one endpoint similar to one of its own endpoints. We say at least, because a path could have a link followed by a link that has both endpoints similar to its own endpoints; meaning that the call goes through a link and back through the same link. This scenario occurs between different segments for example when a call from floor one to floor 2 occurs it will go to floor 1 switch then to switch 1 then to router the back again to switch 1 along the same link then floor 2 switch.

To insure that the path is continuous the previous algorithm is triggered as soon as the user adds a link to the path in step 4. At first the list box will show a list of all paths when the user adds the first link the link is sliced into N1 and N2, representing the left endpoint and right endpoint respectively. Then the program will loop through the link

HashTable slicing every link into item 1 and item 2 representing the left endpoint and the right endpoint respectively. At every iteration, the program will check if the link L contains N1 or N2; it will also check that both endpoint of L (item 1 and 2) are not servers. If these conditions are meet then the link is added to the list. The reason why the program check that item 1 and 2 are not servers is because links that are connected to servers do not play a role in the call path. The only reason these links were added was because they contribute to the background traffic.

4.2.2.3 Drawing

An important part of the IP Telephony Simulator is the network diagram. It will assist the user in getting a visual view of what the network looks like. The program is composed two files that draw the diagram they are NetworkDiagram.cs and Lithium.dll. In this section we will only explain NetworkDiagram.cs as Lithium.dll is what takes care of the graphical aspects and is irrelevant to the contexts of this research.

The NetworkDiagram.cs file contains a class called NetworkDiagram. This class takes the Links HashTable and Nodes HashTable as input and draws the network diagram using the following algorithms.

```
INPUT: links HashTable and Nodes HashTable
OUTPUT: elects a node as a network root for the diagram

RouterMostLinks=0
SwitchMostLinks=0

foreach (thisNode in NodesHT)

      if(thisNode is not a floor or server) then

      ThisRouterLinks=0
      ThisSwitchLinks=0

      foreach (thisLink in LinksHT)

      if(thisLink is not Internal) then
      if(thisNode is part of thisLink and thisNode is a router) then
      ThisRouterLinks = ThisRouterLinks + 1
      end if
      if(thisNode is part of thisLink and thisNode is a switch) then
      ThisSwitchLinks = ThisSwitchLinks + 1
      end if
      end if

      end foreach

      if(ThisRouterLinks>RouterMostLinks AND thisNode is a router)
      then
      RouterMostLinks=ThisRouterLinks
      RML=thisNode
      end if

      if(ThisSwitchLinks>SwitchMostLinks AND thisNode is a switch)
      then
      SwitchMostLinks=ThisSwitchLinks
      SML=thisNode
      end if
      end if

end foreach

if(RML is not empth) then
NetworkRoot=RML
else
NetworkRoot=SML
end if

DrawChildren(NetworkRoot, root UID)
```

Figure 4.15 Pseudo-code for root election

The Network diagram is represented as a binary tree; therefore the first step to drawing the diagram is electing a root for the tree. The previous algorithm will elect the router as the root. If there were two routers then the one with most links will be the root; and if there were no routers then the switch with most links will be elected as the root of the network diagram.

The reason why this election procedure was followed was because it is usually desired to have the router at the top of the network diagram followed by its directly connected switches as children and going down until reaching the end servers.

The previous algorithm will loop through the nodes HashTable except the floors. In every iteration, it will loop through the links HashTable counting the number of times the node occurred in all the links. If the node is a router then the program will increment ThisRouterLinks and if it was a switch ThisSwitchLinks will be increamented. Subsequently, the program will check if (ThisRouterLinks > RouterMostLinks) and if it is true then this router will be declared a RML (router with most links) and RouterMostLinks=ThisRouterLinks. The same will be also done for the switches. After looping through the entire nodes HashTable the program will check if there is an RML meaning the topology consists of at least one router, then it will elect the RML as the root of the tree; otherwise SML (switch with most links) will be elected as the root.

Once a root is elected the DrawChildren method is called. Initially the NetworkRoot and the RootUID are passed as parameters. The RootUID is a unique ID for every node in the diagram; this ID is needed for graphical purposes and will be used by Lithium.dll. The following algorithm explains how DrawChildren draws the children nodes and the children of the children and so on.

```
INPUT: elected root of the network diagram
OUTPUT: network diagram

DrawChildre(TheRoot,RootUID)
{
if(VisitedNodesHT does not containt (TheRoot)) then
      add TheRoot to VisitedNodesHT
end if

foreach (thisLink in LinksHT)

if(thisLink containts (TheRoot) AND thisLink is not [Internal]) then


      if(node1 of thisLink=TheRoot AND VisitedNodesHT does not
      containt (node2 of thisLink)) then

      Draw(node2)
      Color(node2)
      DrawChildren(node2,NodeUID)

      else if(node2 of thisLink=TheRoot AND VisitedNodesHT does not
      containt (node1 of thisLink)) then

      Draw(node1)
      Color(node1)
      DrawChildren(node1,NodeUID)

      end if

end foreach
}
```

Figure 4.16 Pseudo-code for drawing the network diagram

The previous algorithm takes two parameters; the TheRoot which is the network node that requires this method to draw its children and the RootUID. At the beginning, the method checks if TheRoot has been visited before by searching for it in the VisitedNodesHT HashTable. If the node is not found in the VisitedNodesHT then the method will loop through the LinksHT HashTable. In every iteration it will check if the non-internal link contains TheRoot; and if that is the case it will check on which end of the link is TheRoot. If node1 (the left end of the link) was TheRoot and node2 (the right

end of the link) has not been visited; then node2 will be drawn and colored and the method DrawChildren will be called recursively passing to it node2 as a parameter. Likewise, is node2 (the right end of the link) was TheRoot and node 1 (the left end of the link) has not been visited; then node1 will be drawn and colored and the method DrawChildren will be called recursively passing to it node1 as a parameter. This way the entire diagram will be drawn based on the content of the LinksHT and it will be colored based on the node type which is available in the NodesHT.

4.2.2.4 Throughput Analysis

VoIP is bounded by two important metrics. First is the throughput analysis. Second is the end-to-end delay. The actual number of VoIP calls that the network can sustain and support is bounded by the least number of calls obtained by such an available bandwidth or delay. [SAL06]

Throughput analysis is an important step to identify the network element, whether it is a node or a link, that puts a limit on how many VoIP calls can be supported by the existing network. For any path that has *N* network nodes and links, the bottleneck network element is the node or link that has the minimum available bandwidth.

In our program the delay and throughput analysis is done in multiple steps. The first step is done as soon as the user clicks on the analysis tab as shown in the following algorithm.

```
INPUT: background traffic on the links
OUTPUT: (background traffic, flow, ui, maximum calls supported and mu)
of every node

foreach (thisNode in NodesHT)
nodeCount = 0
BGTraffic of thisNode = 0
flow of thisNode = 0
if(thisNode is not a floor) then
foreach (thisLink in LinksHT)
      if(thisLink contains thisNode) then
        if(thisLink is not [Internal]) then
        N1 = left end of thisLink
        N2 = righ end of thisLink
        if (thisNode is N1) then
        increment BGTraffic of this node by BGTrafficpps21
        else if (thisNode is N2) then
        increment BGTraffic of this node by BGTrafficpps12
        end if
        end if
      end if
end foreach
foreach (thisPath in PathHT)
      nodeCount=0
      for i ← 0 to length of PathArray of thisPath do
      if(PathArray[i] of thisPath contains thisNode) then
      nodeCount = nodeCount + 1
      i = i + 1
      end if
      end for
      flow of thisNode = flow of thisNode + (percentage of thisPath x
      nodeCount)
end foreach
end if
if(thisNode is not a server and not a floor) then
```

$$u_i = \frac{BGTraffic_i}{C_i}$$

$$MaxCalls_i = \frac{(1-u_i)\times C_i \times (1-PercentageGrowth)}{VoipBWp}$$

$$TPATotalCallsSupported_i = \left\lfloor \frac{MaxCalls_i}{Flow_i} \right\rfloor$$

$$\mu_i = C_i - (C_i \times PercentageGrowth)$$

```
end if
end foreach
```

Figure 4.17 Pseudo-code for computing background traffic, flow, ui, maximum calls supported and mu of every node

The previous algorithm is composed of a large For loop that loops through all the nodes. Within the loop three major operations are performed to compute the attributes for every node in every iteration. The first operation computes the background traffic imposed on the node, recall that the user only inputs the background traffic imposed on the links and not the nodes therefore the program has to calculate that automatically. This is done by looping through all the links and if the node is found as an endpoint of a link then depending on which end it is on BGTrafficpps21 or BGTrafficpps12 will be added to BGTraffic.

The second part of the For loop calculates the flow for every node. That is done by looping through all the paths counting in every iteration the number of times the node was mentioned in the path. That is then multiplied by the percentage of calls expected for the path, which was entered by the user in step 3. Recall that a path is an array of links and an end point of one link is the starting point of the next. Therefore once the node is found in the array the program should not count the proceeding item to avoid counting the node twice. That is why the For loop that loops through the path array increments *i* twice, once in the For statement and another time inside the loop.

The third part of the For loop computes u_i, μ and TPATotalCallsSupported for every node. These computations are straightforward substitution to the formula as all the variables are available. Although μ is not needed for throughput analysis but it is calculated now to be used later on in the delay analysis.

Similarly another large For loop loops through all the links doing the same computations. The algorithm is similar to the previous algorithm that calculated the nodes parameters and therefore will not be discussed.

The final calculations for the throughput analysis report is done when the user click the throughput analysis report button. This is done using the following algorithm.

```
INPUT: maximum number of calls supported by ever node and link
OUTPUT: maximum number of calls supported by the network and the
bottleneck

IterCnt = 0

foreach (thisNode in NodesHT)

if(thisNode is not a floor AND thisNode is not a Server) then

IterCnt = IterCnt + 1

      if(IterCnt is 1) then
      MaxNCS = TPATotalCallsSupported for thisNode
      else if (TPATotalCallsSupported for thisNode <MaxNCS) then
      MaxNCS = TPATotalCallsSupported for thisNode
      Bottelneck = thidNode
      end if

end if

end foreach

foreach (thisLink in LinksHT)

if(thisLink is not internal)then

item1 = left end of the link
item2 = right end of the link

if (item1 and item2 are not servers) then
      if(TPATotalCallsSupported for thisLink < MaxNCS) then
      MaxNCS = TPATotalCallsSupported for thisLink
      Bottelneck = thisLink
      end if
end if
end if

end foreach
```

Figure 4.18 Pseudo-code for computing maximum number of calls supported and the bottel neck

As shown in the above algorithm the program first loops through all the nodes except the servers and floors. In every iteration it checks TPATotalCallsSupported (Throughput analysis total calls supported) for every node. The node with the least TPATotalCallsSupported is the bottleneck node and the TPATotalCallsSupported for that node is the maximum number of calls the nodes in the network can take. Similarly in the second for loop the program goes through the links checking TPATotalCallsSupported and if a link is found with a TPATotalCallsSupported less than the maximum number of calls supported by the nodes then this link becomes the bottleneck and the total calls it can support is declared to be the maximum calls the network can support.

4.2.2.5 Delay Analysis

As defined in Section 2.4 for the existing network, the maximum tolerable delay for a VoIP packet is 150 ms. The maximum number of VoIP calls that the network can sustain is bounded by this delay. We must always ascertain that the worst-case end-to-end delay for all the calls must be less than 150 ms. As described in Section 2.4, there are three sources of delay for a VoIP stream: sender, network, and receiver.

As mentioned in the previous section, in our tool some of the calculations for the delay and throughput analysis are done as soon as the user clicks on the analysis tab. Furthermore, when the user clicks the delay analysis report button the program will go through three functions where one of which will determine the maximum calls supported by the network based on throughput analysis. Before explaining the three functions the following Figure shows a higher view of the algorithm.

```
keeploop = true
while(keeploop is true)

      TotalCallsSupported = TotalCallsSupported + 1

      FUNCTION1 (when MM1 is negative)
      if (keeploop==true) then
            FUNCTION2 (when MD1 is negative)
      end if
      if (keeploop==true) then
            FUNCTION3 (when MM1 and MD1 are positive)
      end if

end while
```

Figure 4.19 High view of the delay analysis algorithm

In the previous algorithm, one call is added every iteration. Afterwards, one of the three functions determines if the network can support that number of calls. If function1 doesn't determine the number of calls supported then keeploop will remain true and function2 will be triggered. Only when function1 and 2 do not determine the number of calls supported function3 will be triggered and it will determine the number of calls supported by the network based on delay analysis.

The following algorithm explains function 1.

```
INPUT: nodes of the network
OUTPUT: lambda, MM1 and total calls supported

foreach (thisNode in NodesHT)
      if(thisNode is not a floor or server) then

       DATotalCallsSupported_i = TotalCallsSupported × flow_i
       λ_i = (VoipBW × DATotalCallsSupported_i) + BGTraffic_i
       MM1Delay_i = 1 / (μ_i − λ_i)

            if(MM1<0) then
                  TotalCallsSupported = TotalCallsSupported - 1
                  Keeploop = false
                  halt
            end if
      end if
end foreach
```

Figure 4.20 Pseudo-code (Function 1) for computing lambda, MM1 and total calls supported

As shown in the above algorithm the For Loop goes through all the nodes except the floors and servers. Using the TotalCallsSupported the function computes μ, λ and *MM1*. If *MM1* is less than zero this means that the network cannot take that number of calls therefore TotalCallsSupported is decremented and keeploop becomes false to prevent function2 and 3 from triggering. At this point the program breaks the For Loop because there is no need to further check the rest of the nodes.

Although *MM1* is the average delay of the node and delay cannot be in negative, in some scenarios it is. That is because *MM1* by definition is $\frac{1}{\mu - \lambda}$ and when $\mu < \lambda$ this means that the nodes capacity is less than the bandwidth generated by the total number of

calls. In such cases delay doesn't play a factor in determining the maximum number of calls supported by the network.

If the program goes through all the nodes without having the previous scenario then keeploop remains true and function2 is triggered as shown in the following algorithm.

```
INPUT: links of the network
OUTPUT: lambda, MD1 and total calls supported

foreach (thisLink in LinksHT)
      if(thisLink is not internal) then
      DATotalCallsSupported_i = TotalCallsSupported × flow_i
      λ_i = (VoipBW × DATotalCallsSupported_i) + Max(BGTraffic12_i, BGTraffic21_i)
                        1 − λ_i / (2 × μ_i)
      MD1Delay_i = ---------------------
                          μ_i − λ_i
            if(MD1<0) then
                  TotalCallsSupported = TotalCallsSupported - 1
                  keeploop =  false
                  halt
            end if
      end if
end foreach
```

Figure 4.21 Pseudo-code for (Function 2) computin lambda, MD1 and total calls supported

In the same way that function1 looped through the nodes, function2 loops through the links to compute *MD1*. If a link has an *MD1* value less than zero meaning that the link capacity is less than the bandwidth generated by the total number of calls then the program decrements the TotalCallsSupported and breaks the loop and keeploop becomes false to prevent from going to function3.

If all the nodes had an *MM1* value larger than zero and all the links had an *MD1* value larger than zero then function3 is triggered as shown in the following algorithm.

```
INPUT: paths of the network
OUTPUT: combined path delay, Total calls supported

lastpathDelay = 0
foreach (thisPath in PathHT)

      Calculate pathDelay
      if (pathDelay <= The Network Latency AND pathDelay >= 0)then
            if (pathDelay > lastpathDelay) then
            lastpathDelay = pathDelay
            end if
      else

            if (keeploop is true) then
                  TotalCallsSupported = TotalCallsSupported - 1
                  Keeploop = false
            end if
      end if

end foreach
```

Figure 4.22 Pseudo-code for computing path delay (function 3), total calls supported

In the previous algorithm the program will loop through the paths in the path HashTable and it will calculate the combined delay of the entire path. If the delay is less than the network latency then the network cannot take anymore calls. The program will furthermore search for the path with the highest delay and that path is marked as the bottleneck and it will determine the maximum calls supported by the network based on delay analysis.

4.3 Simulation Run

In this section we will use our tool to measure IP telephony readiness for a typical small enterprise.

4.3.1 Network Topology

As illustrated in Figure 4.23 the network presented in [SAL06] is Ethernet-based and has two Layer-2 Ethernet switches connected by a router and three LAN Layer-2 switches. The router is Cisco 2621, and the switches are 3Com Superstack 3300. Switch 1 connects to Floor 1 switch and Floor 2 switch and two servers; while Switch 2 connects to Floor 3 switch and four servers. The network makes use of VLAN's in order to isolate broadcast and multicast traffic. VLAN1 includes the database and file servers. VLAN2 includes Floor 1. VLAN3 includes Floor2. VLAN4 includes the servers for E-mail, HTTP, Web & cache proxy, and firewall. VLAN5 includes Floor 3. All the links are switched Ethernet 100Mbps full duplex.

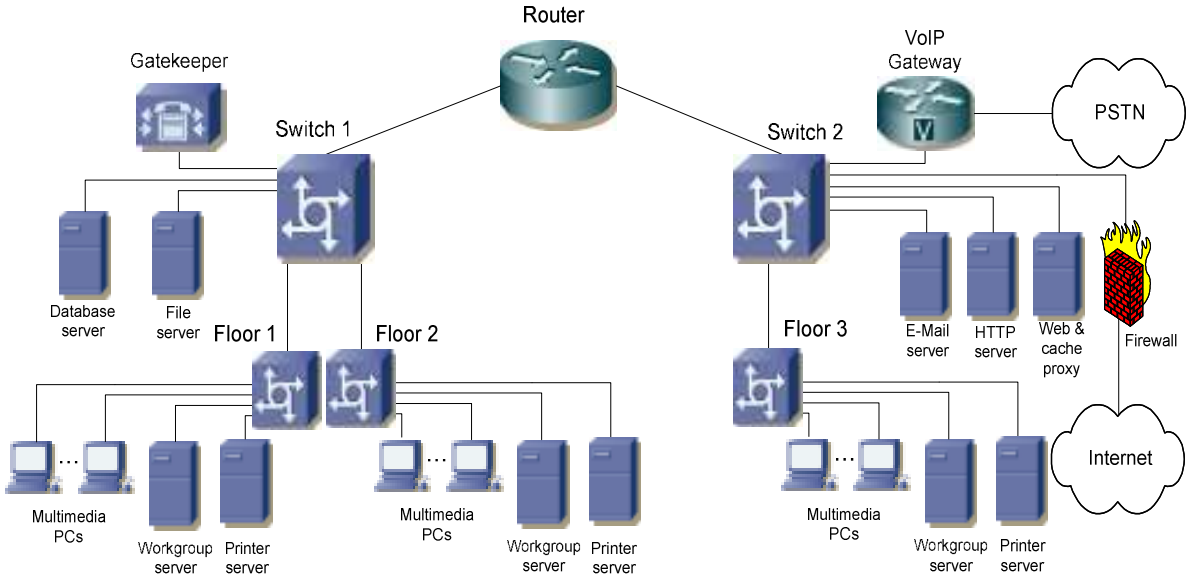

Figure 4.23 Enterprise Network Topology

4.3.2 Network Measurements

Before we can run our simulator some network measurements need to be performed. At first we need to identify the capacity of the network elements (switches and routers) in packets per second. This can be obtained from the datasheet of the device or the vendor's website. In our network we have a Cisco 2621 router with a capacity of 25,000 pps. All of the switches in the network are 3Com Superstack 3300 with a capacity of 1,300,000 pps.

We also have to identify the floors that users will be establish that VoIP calls. These could be physical floors of a building or logical groups or VLAN's. The floors

along with the nodes and their capacities are entered to the program in step 1 as mentioned in section 5.1.1.1.

A second major measurement is the links capacity and background traffic. The network management group should be well aware of all the links capacity. In our topology all the links have 100 Mbps capacity and are full duplex.

There are many ways to measure background traffic. Since all our network components are SNMP enabled we were able to use MRTG. The Multi Router Traffic Grapher (MRTG) is a freeware tool for monitoring and reporting traffic. It has advanced capabilities in generating HTML pages reporting results graphically.

We performed our measurement on all the links at peak hours and recorded the results in both Megabits per second and packets per second for incoming and outgoing traffic. The following table shows the peak utilization obtained for every link.

Link	Link utilization (Mbps)	Link utilization (pps)
Router ↔ Switch 1	9.44	812
Router ↔ Switch 2	9.99	869
Switch 1 ↔ F1 Switch	3.05	283
Switch 1 ↔ F2 Switch	3.19	268
Switch 1 ↔ File Server	1.89	153
Switch 1 ↔ Database Server	2.19	172
Switch 2 ↔ F3 Switch	3.73	312
Switch 2 ↔ Email Server	2.12	191
Switch 2 ↔ HTTP Server	1.86	161

Switch 2 ↔ Firewall	2.11	180
Switch 2 ↔ Proxy	1.97	176

Table 4-2 Links Utilization

These results along with the links capacities are entered to the simulator in step 2 as explained in section 5.1.1.2.

A third measurement that needs to be collected is the call distribution. This can be figured out from knowing the current number of telephone calls in the enterprise. Also call location and flow is important. This will include the percentage of calls made internally or externally. Call distribution must include percentage of calls within and outside of a floor, building, department, or organization. The following matrix represents the call distribution between the floors.

	Floor 1	**Floor 2**	**Floor 3**	**external**
Floor 1	4/27	2/27	2/27	1/9
Floor 2		4/27	2/27	1/9
Floor 3			4/27	1/9

Table 4-3 Call distribution

These numbers are entered in to the simulator in step 3 as explained in section 5.1.1.3.

A forth measurement is to determine the path every call will take in order to reach from one floor to another. This is done in step 5 as explained in section 5.1.1.5.

Finally we can run the analysis provided we know which VoIP standards will be used. In our topology we will use ITU G.711u. The attributes of this standard can be modified as shown in Figure 4.5.

Figures 5.24 and 5.25 show the output for throughput and delay analysis respectively. Based on the throughput analysis report the network can support 314 calls and the bottleneck is the router. Whereas based on the delay analysis report the network can support 307calls and the path with the highest delay is floor 2 to floor 3 with a total delay of 74.4 ms.

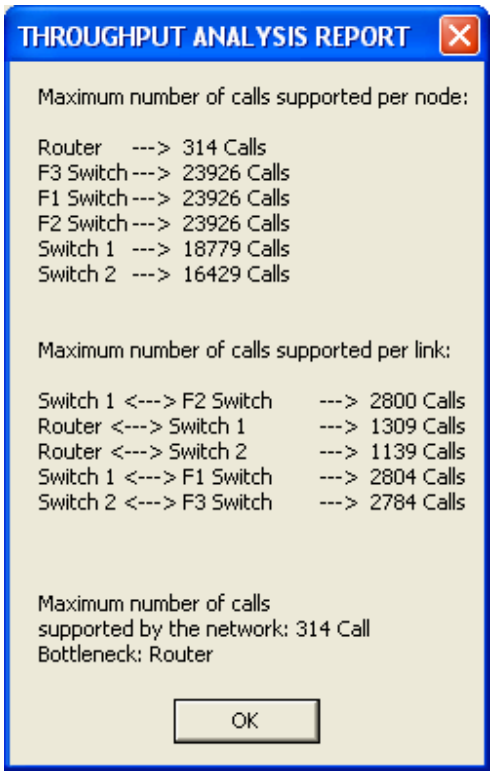

Figure 4.24 Throughput analysis report

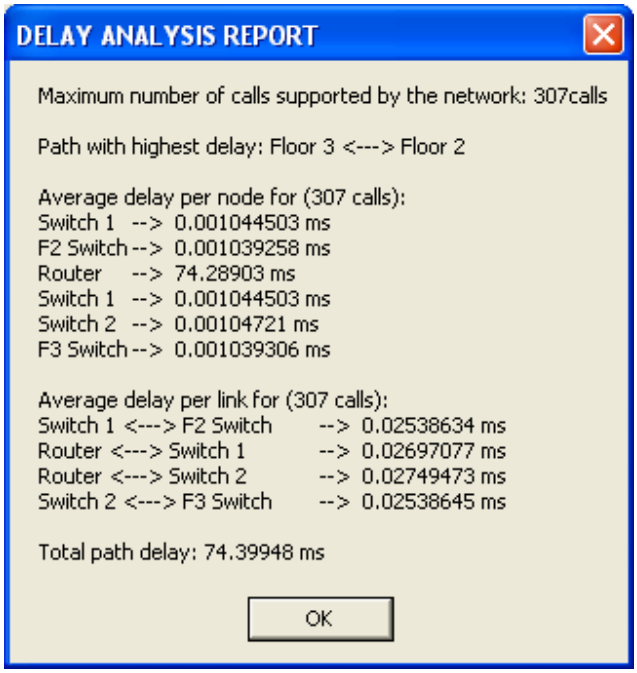

Figure 4.25 Delay analysis report

4.4 Validation and Comparison

To validate the results obtained using our tool, we will compare it with results obtained from a previous study performed using a simulation tool [SAL06]. The study used the popular MIL3's OPNET Modeler simulation package, Release 8.0.C.

The OPNET simulation was executed on a Sun E450 server. The Sun E450 server had Solaris 7 OS with a SPARCV9 400 MHz processor and 1G bytes of memory. The elapsed time for the 8 minute simulation run was approximately 15 hours and produced 722 million events, executed at average speed of 13878 event/second.

Using the same topology used in our tool, OPNET estimated that the network can support 315 calls. More details of this study could be found in [SAL06].

Using our analytical tool similar results were obtained instantaneously running on a regular workstation. As discussed earlier the total number of calls the network can support obtained from our analytical tool is 307 calls based on delay analysis and 314 calls based on throughput analysis. This is close to the 315 calls supported obtained from OPNET.

Due to the simplicity of our analytical tool sales staff can also make use of this tool to quickly show customers what needs to be done to successfully deploy telephony in their network. Network Designers can also make use of this tool to verify their design. Researchers can also take advantage of this tool to quickly verify results obtained using other approaches. Figure 4.26 shows the intended audience for some general purpose network design tools explained earlier and our tool including the level of skills required.

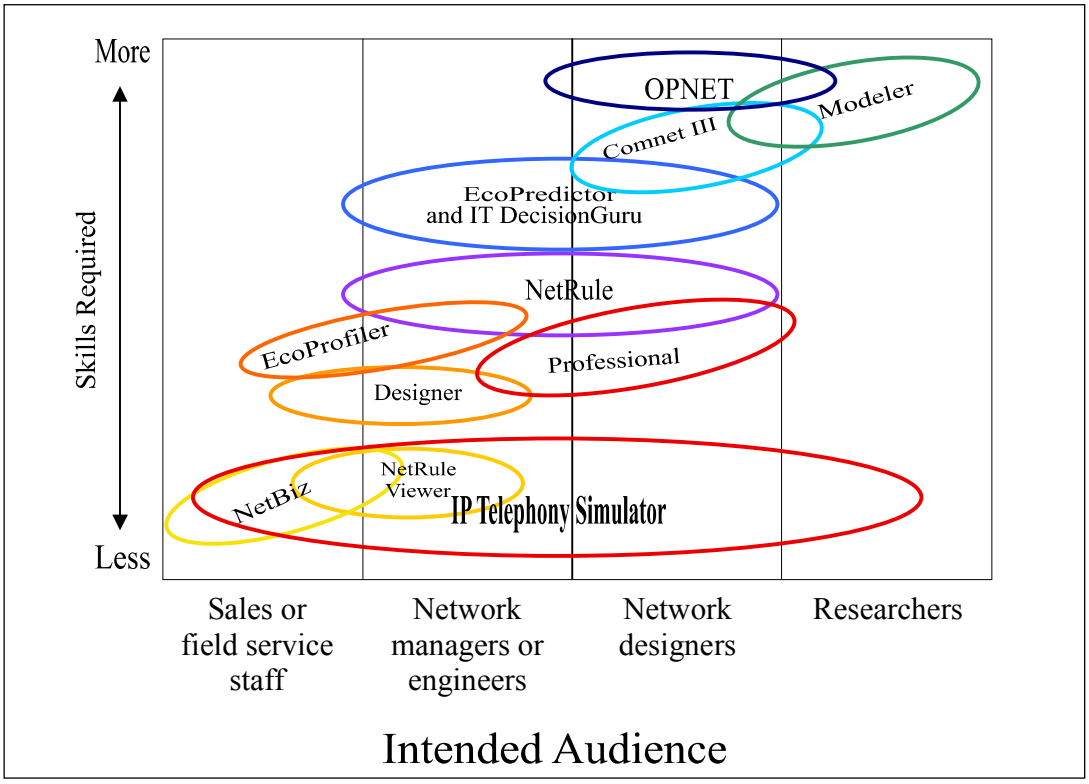

Figure 4.26 Intended Audience for some general purpose network design tools including our tool

CHAPTER 5

CONCLUSION

This chapter presents a summary of our major contributions in this thesis work. It also gives suggestions for future research directions.

In this thesis we have presented general purpose network design tools and VoIP design tools. We have categorized the general purpose tools into analytical and simulation tools and we categorized the VoIP design tools into tools performed online and tools performed offline we then further categorized the online tools into hardware-based and software-based. We have shown the skills required for some of these tools and the intended audience.

One of our major contributions in this thesis is designing and implementing an analytical tool to measure network readiness for IP Telephony. We have explained how to use this tool and how to generate throughput and delay analysis reports. Furthermore, we went through the algorithms used in our program. Finally, we conducted a simulation run on an enterprise topology and compared our results with results obtained using a simulator.

Our tool can help network researchers, designers, operators and salesmen to determine quickly and easily how well VoIP will perform on a network prior to deployment. Prior to the purchase and deployment of VoIP equipment, it is possible to predict the number of VoIP calls that can be sustained by the network while satisfying QoS requirements of all existing and new network services and leaving enough capacity for future growth.

The topics presented in this thesis open a new horizon for further research. The followings are some future directions:

- In our tool the user has to manually enter the network topology. Further enhancements could be made to the tool allowing it to automatically read the topology from the live network using SNMP.
- Our tool shows a network diagram of the topology but doesn't allow the user to modify the network through the graphical network diagram.
- Determining the path in our tool is entered by the user. This can be enhanced to be automatically fed to the tool once the tool knows the configuration of the nodes.
- Simple modifications could be done to the tool to make it measure network readiness for other applications such as video conferencing.
- In our tool, only peer-to-peer voice calls were considered. As a future work, one can consider implementing important VoIP options such as VoIP conferencing and messaging.
- In our tool the network topology is entered as a list of nodes and links. This can be further enhanced to be visualized as the user builds the network by having a drag-and-drop tool palette and also having the ability to import the diagram from Visio or HP OpenView

Bibliography

[ANAL05] Analytical Engines NetRule, http://www.netrule.com/, 2005.

[ARN00] Arnold B., Network Design: Which Network Design Tool Is Right for You? *IT Pro IEEE*, pages 23- 35, October 2000.

[BEA03] Bearden, M. & Denby, L. Avaya Labs Research: Assessing Network Readiness for IP Telephony, February 2003.

[BRIX05] Brix Networks, http://www.brixnet.com/products/voip_testsuite.html, 2005.

[EMPR05] Emprix Hammer Call Analyzer, http://www.empirix.com/, 2005.

[GOO02] Goode B, "Voice over Internet Protocol (VoIP)," *Proceedings of IEEE*, vol. 90, no. 9, Sept. 2002, pp. 1495-1517.

[KAR02] Karam M. and Tobagi F., "Analysis of delay and delay jitter of voice traffic in the Internet," *Computer Networks Magazine*, vol. 40, no. 6, December 2002, pp. 711-726.

[KAR04] Karacaly, B., Denby, L. and Meloche, J., Avaya Labs Research: Scalable Network Assessment for IP Telephony, Communications, 2004 *IEEE International Conference*, pages 1505- 1511, June 2004.

[LAN02] Lancaster T., "VoIP in the Enterprise: Preparing Your Network," January 31, 2002.

[MAR03] Markopoulou A., Tobagi F., Karam M., "Assessing the quality of voice communications over internet backbones", *IEEE/ACM Transaction on Networking*, vol. 11, no. 5, 2003, pp. 747-760.

[MET01] Mehta P. and Udani S., "Voice over IP", *IEEE Potentials Magaizine*, vol. 20, no. 4, October 2001, pp. 36-40.

[NETC] NetCracker, http://www.netcracker.com, 2005.

[NET05] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/, 2005

[OPNET05] Opnet Technologies, http://www.opnet.com, 2005.

[PAW02] Pawlikowski, K., Jeong, H.-D.J. Lee & J.-S.R. Canterbury Univ.,

Christchurch: On credibility of simulation studies of telecommunication networks, *Communications Magazine, IEEE*, pages 132-139, January 2002.

[PAW90] Pawlikowski K., Steady-State Simulation of Queuing Processes: A Survey of Problems and Solutions, *ACM Computing Surveys*, pages 123–170, February 1990.

[RAD05] Radcom VoIP Performer, http://www.radcom.com/radcom/test/VoIPPerformer.htm, 2005.

[REC96] Recommendation G.114, "One-Way Transmission Time," *ITU*, 1996.

[REC97] Recommendation H.323, "Packet-based Multimedia Communication Systems," *ITU*, 1997.

[REC98] Recommendation G.711, "Pulse Code Modulation (PCM) of Voice Frequencies," *ITU*, November 1998.

[ROB99] Robert C., Network World: "Test-drive Network Designs," May 1999.

[SAL06] Salah, K., "On the Deployment of VoIP in Ethernet Networks: Methodology and Case Study," *International Journal of Computer Communications*, Vol. 29, no. 8 Elsevier Science, May 15, 2006, pp. 1039 – 1054.

[SUN05] Sunrise Telecom Ghepardo, http://www.sunrisetelecom.com/ghepardo/voip.shtml, 2005.

[VIO05] Viola Networks, http://www.violanetworks.com/, 2005.

[VIVI05] Vivinet Assessor, http://www.netiq.com/products/va/default.asp, 2005.

[WIL00] William S. Queuing Analysis 2000.

Vita

- Meshal Abdulaziz Almashari.
- Born in Miami, Florida, August 28, 1978.
- Completed Bachelor of Science in Computer Science from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, in August 2001.
- Joined Saudi ARAMCO October 2001 as a Network Engineer
- Completed Master of Science in Computer Science from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, in May 2006.
- Email: meshal.mashari@aramco.com